Genero Studio User Guide
## Contents

**What's New in 3.10**.................................................................................................................12

**Welcome to Genero Enterprise**.............................................................................................16
  - What is Genero Enterprise........................................................................................................17
  - Tour Genero Studio..................................................................................................................22
  - Running the Demos..................................................................................................................42
  - Switching Genero Clients.........................................................................................................46
  - Creating with Quick Starts.......................................................................................................48
  - Finding more information..........................................................................................................49

**Welcome to Genero Studio**....................................................................................................49
  - Quick Start: Tour of Genero Studio........................................................................................49
  - Run the OfficeStore demo........................................................................................................50
  - Explore the Debugger..............................................................................................................54
  - vi emulator and diff tools in Code Editor..............................................................................63
  - Explore Source Code Management........................................................................................63
  - Explore database meta-schemas.............................................................................................70
  - Explore forms..........................................................................................................................71
  - Analyze code............................................................................................................................77
  - Generate code..........................................................................................................................83
  - The Welcome Page...................................................................................................................84
  - Modules.....................................................................................................................................85
  - Menus and Toolbars................................................................................................................96
  - Views.........................................................................................................................................116
  - Documents...............................................................................................................................121
  - Dialogs.......................................................................................................................................121
  - Genero Workplace Window....................................................................................................124
  - Updating Genero Studio and its documentation..................................................................124
  - Learning to use Genero Studio...............................................................................................125
  - Command line options.............................................................................................................125
  - Samples directory.....................................................................................................................126
  - Integrating existing applications.............................................................................................127
  - Setting Preferences..................................................................................................................128
  - Access Help..............................................................................................................................138

**Configuring Genero Studio**..................................................................................................138
  - Software configuration scenarios..........................................................................................139
  - Default configuration..............................................................................................................141
  - Change the active configuration.............................................................................................141
  - Configuring for BAM................................................................................................................142
  - Configure for Genero Mobile..................................................................................................142
    - Configure Genero Mobile for Android..............................................................................142
| Contents |
|------------------|------------------|------------------|
| Configure Genero Mobile for iOS | Display to the Genero Mobile Development Client | Setting up a local environment |
| Setting up a remote environment | Remote environment software requirements | Define local Genero installations, GAS configurations, and environment sets |
| Local environment software requirements | Define mount points to shared drives | Add remote host |
| Define remote Genero installations, GAS configurations, and environment sets | | |
| Access a database | | |
| Language support (text encoding) | Configuration reference | |
| Configure Genero Studio to use a different character set | GAS standalone dispatcher: httpdispatch | Configure the display clients |
| Add a text encoding plugin | Import Preferences dialog | Configure for the Desktop client |
| Character mapping table (encodingMap.xml) | Import Configuration dialog | Configure for the Web client |
| Configure keyboard and language on a Windows® client | GAS standalone dispatcher: httpdispatch | Configure for Mobile clients |
| Configure LANG on a Genero Studio Server | Configuration error messages | Configure for a TUI client |
| Test text encoding configuration | Microsoft® Windows® Security Blocking | |

**Business Application Modeling (BAM)**

| Quick Start: Generate an application | Create a BAM project | Add a meta-schema to the project (4dbx) |
| Create the Business Application diagram (4ba) | Implement the program and form | Generate and run the application |
| Add a detail list to the form | | Add a remote host |
| | | Define mount points to shared drives |
| | | Add remote host |

**Quick Start: Generating a mobile app**

| Create a BAM mobile project | Model the database | Model the app |
| Generate and run the app | Add phone functionality to the app | Customize the app |
| Package and Deploy | | |

**BAM Concepts**

| What is Business Application Modeling (BAM)? | How code is generated | |


| Contents |

<table>
<thead>
<tr>
<th>Meta-schema Manager</th>
<th>343</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is a database meta-schema? (4db)</td>
<td>343</td>
</tr>
<tr>
<td>Creating a meta-schema</td>
<td>344</td>
</tr>
<tr>
<td>Create a meta-schema</td>
<td>344</td>
</tr>
<tr>
<td>Extract meta-schema information from database</td>
<td>345</td>
</tr>
<tr>
<td>BDL schema file (sch)</td>
<td>349</td>
</tr>
<tr>
<td>Add a meta-schema to a project</td>
<td>349</td>
</tr>
<tr>
<td>Adding to a meta-schema</td>
<td>349</td>
</tr>
<tr>
<td>Add new tables and columns</td>
<td>349</td>
</tr>
<tr>
<td>Add constraints or indexes</td>
<td>350</td>
</tr>
<tr>
<td>Add foreign keys</td>
<td>351</td>
</tr>
<tr>
<td>Add a many-to-many relationship</td>
<td>354</td>
</tr>
<tr>
<td>Centralize field information (label, widget, default value)</td>
<td>355</td>
</tr>
<tr>
<td>Viewing a meta-schema</td>
<td>356</td>
</tr>
<tr>
<td>Comparing two meta-schemas</td>
<td>358</td>
</tr>
<tr>
<td>Update a meta-schema from database</td>
<td>359</td>
</tr>
<tr>
<td>Generate a database script from meta-schema</td>
<td>359</td>
</tr>
<tr>
<td>Generate meta-schema documentation</td>
<td>361</td>
</tr>
<tr>
<td>Copy a table or column definition as text</td>
<td>361</td>
</tr>
<tr>
<td>Meta-schema Manager Reference</td>
<td>361</td>
</tr>
<tr>
<td>Meta-schema properties</td>
<td>362</td>
</tr>
<tr>
<td>Data types</td>
<td>363</td>
</tr>
<tr>
<td>Database server/user information</td>
<td>365</td>
</tr>
<tr>
<td>Dialogs</td>
<td>367</td>
</tr>
<tr>
<td>Views</td>
<td>373</td>
</tr>
<tr>
<td>Meta-Schema Manager preferences</td>
<td>375</td>
</tr>
<tr>
<td>Meta-schema diagram context menu</td>
<td>376</td>
</tr>
<tr>
<td>Meta-schema Manager error messages</td>
<td>376</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB Explorer</th>
<th>392</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open DB Explorer</td>
<td>393</td>
</tr>
<tr>
<td>Using the Views menu</td>
<td>393</td>
</tr>
<tr>
<td>From the meta-schema or Data Model</td>
<td>394</td>
</tr>
<tr>
<td>Change connection details</td>
<td>394</td>
</tr>
<tr>
<td>Show data</td>
<td>395</td>
</tr>
<tr>
<td>Show table data (start with meta-schema diagram)</td>
<td>395</td>
</tr>
<tr>
<td>Show table data (start with DB Explorer)</td>
<td>395</td>
</tr>
<tr>
<td>Show data for select columns</td>
<td>395</td>
</tr>
<tr>
<td>Show data for a business record</td>
<td>396</td>
</tr>
<tr>
<td>Change the data</td>
<td>396</td>
</tr>
<tr>
<td>Show data from a table in edit mode</td>
<td>396</td>
</tr>
<tr>
<td>Show data from a different table in edit mode</td>
<td>396</td>
</tr>
<tr>
<td>Update data</td>
<td>396</td>
</tr>
<tr>
<td>Insert a row</td>
<td>397</td>
</tr>
<tr>
<td>Duplicate a row</td>
<td>397</td>
</tr>
<tr>
<td>Delete a row</td>
<td>397</td>
</tr>
<tr>
<td>Limit rows</td>
<td>397</td>
</tr>
<tr>
<td>Write a SQL query by hand</td>
<td>398</td>
</tr>
<tr>
<td>Execute a query</td>
<td>399</td>
</tr>
</tbody>
</table>

| File extensions and BAM | 324 |
| Business Application Modeling error messages | 325 |
| Business Records error messages | 336 |
| Business Application Diagram error messages | 340 |
| Contents |

**Project Manager** ................................................................................................... 399
- Genero project file (4pw) .............................................................................................. 400
- Structured projects ........................................................................................................ 400
- Quick Start: Create a project .......................................................................................... 404
- Creating new projects ...................................................................................................... 404
  - Create a new project .................................................................................................. 404
  - Import existing files as a new project ....................................................................... 409
  - mkproject - Convert a Makefile to a project .............................................................. 409
  - Connect to existing build systems .............................................................................. 409
- Organizing projects ....................................................................................................... 410
  - Groups, Applications, Libraries, and Packages ......................................................... 410
  - Using external libraries .............................................................................................. 411
  - Setting external dependencies .................................................................................. 411
- Application configuration files .................................................................................... 411
- Building and linking programs ..................................................................................... 412
  - Languages ................................................................................................................ 413
  - What are build rules .................................................................................................. 413
  - Add/Edit a build rule .................................................................................................. 414
  - Example: How build rules work ................................................................................ 415
  - Link rules .................................................................................................................. 416
  - Execution rules ......................................................................................................... 416
  - Command line options for build, link, execution rules ............................................. 416
  - Environment variables ............................................................................................... 418
  - Pre/Post compile ........................................................................................................ 420
  - Pre/Post link .............................................................................................................. 421
  - The gmake command ................................................................................................ 421
- Packaging ..................................................................................................................... 422
- Locate a file (starting at Project Manager) ................................................................. 423
- Project Manager Reference .......................................................................................... 423
  - Project Manager context menu .................................................................................. 423
  - Dialogs ....................................................................................................................... 424
  - Project Manager node properties ............................................................................. 434
  - Predefined node variables ....................................................................................... 436
- Project Manager error messages .................................................................................. 440

**Code Editor** ............................................................................................................ 445
- Editing code files........................................................................................................... 445
  - Code Editor basics .................................................................................................... 445
  - Smart editing - indenting, tabs, and backspace ....................................................... 446
  - Bookmarks ................................................................................................................. 447
  - Auto completion (Ctrl+Space) .................................................................................. 447
  - Code templates (Ctrl+T) .......................................................................................... 447
  - Split a document ...................................................................................................... 448
  - Search and replace .................................................................................................... 448
- Using the Diff tool ........................................................................................................ 451
- Code coverage ............................................................................................................... 454
- Highlight identifiers ...................................................................................................... 455
  - Define identifiers ..................................................................................................... 455
  - Set the color and font of identifiers ......................................................................... 456
- Printing files .................................................................................................................. 457
- XML Catalog files ........................................................................................................ 457
- Code Editor Reference ................................................................................................. 458
  - Code Editor preferences ......................................................................................... 459
Debugging an application in Studio ................................................................. 589
  Graphical Debugger ...................................................................................... 589
  Controlling program execution ....................................................................... 590
  Debugger output ............................................................................................... 595
  Examining data ................................................................................................ 595
  Examining execution flow ................................................................................ 596
  Record/replay a macro ..................................................................................... 596
  Profiler .............................................................................................................. 596

File Browser ........................................................................................................ 588
  Navigating files in File Browser ................................................................. 588
  Selecting files in File Browser ....................................................................... 588
  Managing files in File Browser ....................................................................... 589
  Locate a file (starting at File Browser) ........................................................ 589

GUI Testing ........................................................................................................ 480
  Quick Start: Regression testing ....................................................................... 480
  The testing framework ..................................................................................... 481
  Create and run a unit test ............................................................................... 481
  Viewing macro definitions ............................................................................... 482

Code Analyzer .................................................................................................... 482
  Sequence Diagrams ......................................................................................... 483
  Dependency Diagrams ...................................................................................... 484
  Dependency diagram context menu ............................................................... 486

Form Designer .................................................................................................... 486
  Forms in Genero applications ......................................................................... 486
  Quick Start: Creating a first form .................................................................... 487
  Creating the user interface .............................................................................. 488
    Forms .............................................................................................................. 488
    Business records (data sets) ......................................................................... 493
    Containers .................................................................................................... 496
    Widgets ......................................................................................................... 504
    Web components .......................................................................................... 515
    Action management (Toolbars, Topmenus) .................................................. 517
    Styles ............................................................................................................ 522
  Form Designer usage ....................................................................................... 523
    Drawing ......................................................................................................... 523
    Selecting, moving, resizing ........................................................................... 524
    Aligning ......................................................................................................... 524
    Transforming ................................................................................................ 525
    The gsform command .................................................................................. 525
    Localizing your form .................................................................................... 527
    Form Designer Reference ............................................................................. 528

Profiler ............................................................................................................... 596
  Record/replay a macro ..................................................................................... 596
  Examining data ................................................................................................. 595
  Debugger output ............................................................................................... 595
  Examining execution flow ............................................................................... 596
  Code Editor error messages ............................................................................. 479
Source Code Management (SCM) ........................................................................... 624

What is the SCM module?...................................................................................... 624
Using the SCM module......................................................................................... 625
Checkout files....................................................................................................... 625
Add files.............................................................................................................. 626
Commit / Review changes.................................................................................. 626
Locking................................................................................................................ 627
Revert changes / Un-add files............................................................................ 629
Delete files......................................................................................................... 630
Update / Update All.......................................................................................... 630
Cleanup............................................................................................................... 631
Copy working files and directories.................................................................... 631
Revert from a single revision............................................................................. 631
Merge and revert.............................................................................................. 631
Move a working copy (Switch).......................................................................... 632
Create patch....................................................................................................... 632
Apply patch......................................................................................................... 632
Browse repository.............................................................................................. 633
View log information......................................................................................... 633
Specify the revision range for logs................................................................... 633
Blame.................................................................................................................. 634
Diff with revised file.......................................................................................... 635
Integrate bug tracking....................................................................................... 635
SCM module reference...................................................................................... 636
Views................................................................................................................. 637
Dialogs.............................................................................................................. 640
Specify a Subversion client............................................................................... 645
SVN error messages......................................................................................... 645

Report Writer.................................................................................................... 647

What is Genero Report Writer (GRW)?.............................................................. 647
GRW overview.................................................................................................. 647
GRW runtime architecture................................................................................ 649
Get Started with Reports.................................................................................... 649
Steps to a Report............................................................................................... 650
The Reports demo............................................................................................ 651
Configure printers.............................................................................................. 656
Configure fonts.................................................................................................. 657
Options for data operations in reports.............................................................. 670
Create a report program.................................................................................... 671
The runtime architecture (GRW for Genero BDL).......................................... 672
BDL programs.................................................................................................. 672
Web Services ................................................................. 1075
  Create a Web Services program ........................................ 1076
  Add Web Service .......................................................... 1076
  The Web Services wizard ................................................ 1076
  Build the application ....................................................... 1079
  Update the WSDL ........................................................ 1080

Mobile applications ....................................................... 1080
  Mobile development environment .................................... 1080
  Genero mobile app demos .............................................. 1082
  Debugging a mobile app .................................................. 1083
    Debug version of a deployed app ..................................... 1083
    Debug tools for apps in developer mode ......................... 1085
  View the AUI tree ......................................................... 1086
  View the program logs .................................................. 1088
  View the device logs ..................................................... 1089
  Debugging a Web Component ........................................ 1090
  Localize a mobile app .................................................... 1091
BAM Template Developer Guide ...........................................................................1094
  Quick Start: Customizing templates.................................................................1095
  Example 1: Adding a new property.................................................................1095
  Example 2: Adding a File > >> New item........................................................1099
  Example 3: Adding an entity to the BA Diagram.............................................1101
  Example 4: Configuring a template to use an old FGL version.......................1105
  How code is generated....................................................................................1107
  The code generation template set.................................................................1110
    Interpreting settings.agconf.......................................................................1111
    Interpreting creatables.conf......................................................................1116
  Tcl basics and samples..................................................................................1117
    Add POINT and BLOCK sections to template..............................................1117
    Example: Using XSLT instead of Tcl............................................................1118
  Image directory structure.............................................................................1121
  Template Reference.......................................................................................1121
    XML reference.........................................................................................1121
    POINT and BLOCK reference....................................................................1168

Packaging, deploying, and distributing apps....................................................1168
  Packaging for the application server..............................................................1169
    Steps for packaging to the application server..............................................1169
    Determine the files to include in the application.........................................1169
    Create an application configuration file.......................................................1169
    Determine the directory structure for the packaged application................1170
    Create a Genero package node....................................................................1171
    Create directory nodes...............................................................................1171
    Build the package......................................................................................1172
    Deploy the package....................................................................................1172
  Packaging for a mobile device.....................................................................1173
    What is packaging?..................................................................................1173
    Steps for packaging to a mobile device......................................................1175
    Package a mobile app...............................................................................1175
    Deploy a mobile app for testing..................................................................1178
    Package and Directory nodes and properties..............................................1180
    Distribute your app...................................................................................1184
    Manage App updates.................................................................................1187
  Deploy a packaged file................................................................................1189
  Deploying a BAM application as a standalone application............................1189
  Advanced packaging: Create your own packaging rules..............................1189
    Package and deploy rules (desktop).............................................................1189
    Package and deploy rules (mobile)...............................................................1192

Upgrading GST..................................................................................................1193
  New Features of GST....................................................................................1194
    GST 3.10 new features.................................................................................1194
    GST 3.00 new features.................................................................................1198
    GST 2.51 new features.................................................................................1202
    GST 2.50 new features.................................................................................1202
    GST 2.41 new features.................................................................................1208
    GST 2.40 new features.................................................................................1209
  Upgrade Guides for GST...............................................................................1210
Migrating to a new BAM template set................................................................. 1215
  Migrate from dbapp3.2 to dbapp4.0................................................................. 1216
  Migrate from dbapp3.1 to dbapp3.2................................................................. 1216
  Migrate from dbapp3.0 to dbapp3.1................................................................. 1216
  Migrate from dbapp2.0 to dbapp3.0................................................................. 1216
  Migrate from dbapp1.0 to dbapp2.0................................................................. 1217
  Migrate from 2.3x to dbapp1.0........................................................................ 1217
  Migrate customized template sets................................................................. 1218
Using templates with another version of compiler......................................... 1218

Framework error messages............................................................................. 1218

Genero Glossary............................................................................................... 1227
  A.................................................................................................................. 1227
  B.................................................................................................................. 1228
  C.................................................................................................................. 1228
  D.................................................................................................................. 1228
  E.................................................................................................................. 1230
  F.................................................................................................................. 1230
  G.................................................................................................................. 1230
  L.................................................................................................................. 1232
  M.................................................................................................................. 1232
  N.................................................................................................................. 1233
  O.................................................................................................................. 1233
  P.................................................................................................................. 1233
  R.................................................................................................................. 1234
  S.................................................................................................................. 1236
  T.................................................................................................................. 1237
  U.................................................................................................................. 1237
  W.................................................................................................................. 1237

Legal Notices.................................................................................................... 1238

Copyrights and Trademarks............................................................................ 1238

Index.............................................................................................................. 1239
# GST 3.10 new features

This publication includes information about new features and changes in existing functionality. These changes and enhancements are relevant to this publication.

## Table 1: General

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual view allows you to view two documents simultaneously, side-by-side.</td>
<td>See <a href="#">Documents</a> on page 121.</td>
</tr>
<tr>
<td>Create and run an automated unit test of your GUI application using Genero Ghost Client.</td>
<td>See <a href="#">Create and run a unit test</a> on page 481.</td>
</tr>
<tr>
<td>Support for Hi-DPI (High Dots per Inch) screen displays.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>Use the <code>-convert</code> option of the <code>gsmake</code> command to migrate projects to newer versions.</td>
<td>See <a href="#">The gsmake command</a> on page 421.</td>
</tr>
<tr>
<td>Use the <code>gsform</code> command to import <code>.per</code> files and migrate form files to newer versions.</td>
<td>See <a href="#">The gsform command</a> on page 525.</td>
</tr>
<tr>
<td>User actions can be associated with Project Manager nodes; remote user actions can execute on the server side; <code>$(open)</code> command now available; and when executing a user action, the previous execution parameters are remembered.</td>
<td>See <a href="#">User actions</a> on page 132.</td>
</tr>
<tr>
<td>The Genero Desktop Client and Genero Application Server can now be stopped, as they appear as tasks in the <a href="#">Tasks</a> view.</td>
<td>See <a href="#">Tasks view</a> on page 120.</td>
</tr>
<tr>
<td>Projects remember the last configuration used, and prompt you to use that configuration when you re-open a project.</td>
<td>See <a href="#">Genero project file (4pw)</a> on page 400.</td>
</tr>
<tr>
<td>The Welcome Page is now based on Chromium.</td>
<td>See <a href="#">The Welcome Page</a> on page 84.</td>
</tr>
<tr>
<td>Genero Studio now supports TUI (text mode) applications.</td>
<td>See <a href="#">Configure for a TUI client</a> on page 179.</td>
</tr>
<tr>
<td>Genero Studio now manages opening files that have previously been saved using a newer version of Genero Studio.</td>
<td>See <a href="#">Open from a different version</a> on page 101.</td>
</tr>
<tr>
<td>Restore the layout of the default workspaces.</td>
<td>See <a href="#">Workspaces Configuration</a> on page 131.</td>
</tr>
<tr>
<td>When using a remote configuration, the Genero Workplace Window opens on the remote host.</td>
<td>See <a href="#">Tools menu</a> on page 113.</td>
</tr>
<tr>
<td>Genero Archive files can be deployed from the Files view (File Browser).</td>
<td>See <a href="#">Deploy a packaged file</a> on page 1189.</td>
</tr>
</tbody>
</table>

**Open as Text...** and **Open with Encoding...** menu options can be found under the **File** menu, and within the contextual menus in Project Manager and File Browser.

Genero Studio Help includes a button that opens a feedback page for documentation comments. The URL of the current topic is passed to the feedback page.

No additional reference.
## Overview

Genero Studio prompts when a software or documentation update is available. You have the option of completing the install or turning off update notifications.

### Table 2: Project Manager

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>A new project now contains a <strong>Test</strong> application node, for the generated unit test scenario.</td>
<td>See Groups, Applications, Libraries, and Packages on page 410.</td>
</tr>
<tr>
<td>Options added to the <code>$ (4fdcomp)</code> build command.</td>
<td>See Command line options for build, link, execution rules on page 416.</td>
</tr>
<tr>
<td>Filter results to better find your project items.</td>
<td>See Projects view on page 119.</td>
</tr>
<tr>
<td>You can now select an application as a dependency.</td>
<td>See Setting external dependencies on page 411.</td>
</tr>
<tr>
<td>Preview of text (.per) forms supported from both Project Manager and File Browser.</td>
<td>See Build menu on page 108.</td>
</tr>
<tr>
<td>Structured projects are designed to manage the mapping and deployment of custom .ttf fonts.</td>
<td>See Structured projects on page 400.</td>
</tr>
<tr>
<td>Dependencies can be created between application nodes. This is required for generating unit tests for the Genero Ghost Client.</td>
<td>See GUI Testing on page 480.</td>
</tr>
<tr>
<td>At the top of the opened document, the color of the line indicates the existence of errors or warnings. If the document contains an error, the line is red. If the document contains a warning, the line is orange. If there are no errors or warnings, the line is blue.</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>

### Table 3: Business Application Modeler

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM now supports Genero Report Viewer for HTML5.</td>
<td>See CRUD Form entity on page 237.</td>
</tr>
<tr>
<td>A Custom Form entity provides a form in which the code remains free.</td>
<td>See Custom Form entity on page 243.</td>
</tr>
<tr>
<td>For BAM template developers! New RecordField attributes have been added for use with templates: qualifier1, qualifier2, precision, scale, and length.</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>

### Table 4: Meta-Schema Manager

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <strong>Category</strong> property provides a new method for grouping and filtering tables; the <strong>Description</strong> property allows developers to document schema tables and columns.</td>
<td>See Meta-schema properties.</td>
</tr>
<tr>
<td>Copy a table or column from the meta-schema diagram and paste the variable definition into a Code Editor document.</td>
<td>See Copy a table or column definition as text on page 361.</td>
</tr>
<tr>
<td>Support added for index and foreign key extraction for PostgreSQL.</td>
<td>See Extract meta-schema information from database on page 277.</td>
</tr>
</tbody>
</table>
### Overview

A database schema supports views.

#### Table 5: Code Editor

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Code Editor now supports the Swift language, as well as the Web customization languages CSS, SCSS, Sass, JavaScript and JSON.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>The Code Editor supports right-to-left languages, such as Arabic. Set the Text Encoding preference to ISO-8859-6.</td>
<td>See Language support (text encoding) on page 189.</td>
</tr>
<tr>
<td>You can now find a function or variable declaration using CRTL+CLICK.</td>
<td>See Finding declarations on page 446.</td>
</tr>
<tr>
<td>The Structure view synchronizes with cursor movements to show the current function.</td>
<td>See Language Specific preferences tab on page 461.</td>
</tr>
<tr>
<td>In the Code Structure view, private functions and variables are displayed with a lock icon.</td>
<td>See Code Structure view on page 466.</td>
</tr>
<tr>
<td>You can find the callers of a function using the right-click menu.</td>
<td>See Finding callers of a function on page 446.</td>
</tr>
<tr>
<td>You can turn off warning messages.</td>
<td>See Messages configuration on page 134.</td>
</tr>
<tr>
<td>Full file names now appear in the document tabs.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>The selection range now appears in the status bar.</td>
<td>See Code Editor basics on page 445.</td>
</tr>
<tr>
<td>You have the option to copy the entire line if nothing is already selected.</td>
<td>See Behavior &amp; Display preferences on page 460.</td>
</tr>
<tr>
<td>Code Editor can auto-detect UTF-16 and UTF-32 files, even when the byte order mark (BOM) is missing from the file.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>You can select the file encoding when opening a file.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>The file encoding displays in the Status bar.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>Bookmarks are now stored by project. Project-specific bookmarks are restored when a project opens.</td>
<td>See Bookmarks on page 447.</td>
</tr>
<tr>
<td>The Diff tool has been enhanced to quickly include or exclude case sensitivity and white space differences; the number of differences displays; and you can use the navigation toolbar to move between differences.</td>
<td>See Using the Diff tool on page 451.</td>
</tr>
<tr>
<td>Quick run allows you to run the current Genero .4gl file.</td>
<td>See Debug menu on page 109.</td>
</tr>
<tr>
<td>The Code Coverage view provides statistics about source code line usage.</td>
<td>See Code coverage on page 454.</td>
</tr>
<tr>
<td>Highlight identifiers and define the colors to use.</td>
<td>See Highlight identifiers on page 455.</td>
</tr>
<tr>
<td>Set a preference where you can double-click on a word to highlight all occurrences of the word in the document.</td>
<td>See Behavior &amp; Display preferences on page 460.</td>
</tr>
</tbody>
</table>
### Table 6: Form Designer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic generation of <code>.wcsettings</code> file for WebComponents.</td>
<td>See <a href="#">Form Designer preferences</a> on page 528.</td>
</tr>
<tr>
<td>Form item property <code>initialPageSize</code> for <code>ScrollGrid</code> container.</td>
<td>See <code>initialPageSize</code> on page 558.</td>
</tr>
<tr>
<td>Form item property <code>keys</code> for formfields.</td>
<td>See <code>keys</code> on page 560.</td>
</tr>
<tr>
<td>Form item property <code>placeholder</code> defines a hint for the user when the field contains no value.</td>
<td>See <code>placeholder</code> on page 563.</td>
</tr>
</tbody>
</table>

### Table 7: Genero Report Writer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can set rendering hints for optimizing the printing of reports.</td>
<td>See <a href="#">Optimize the rendering process</a> on page 715.</td>
</tr>
<tr>
<td>Driverless, server side, silent printing is available using PJL.</td>
<td>See <a href="#">Printing without a driver using PJL</a> on page 704.</td>
</tr>
<tr>
<td>New functions <code>oddPhysicalPage()</code> and <code>evenPhysicalPage()</code> reveal whether an object starts on an odd or an even page.</td>
<td>See <a href="#">Start on an odd or even page</a> on page 805.</td>
</tr>
<tr>
<td>An SAP® Connector for the Genero Report Writer can be provided on request. Please contact your support center.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>A Pentaho PDI plug-in can be provided on request. Please contact your support center.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>The scripts residing in <code>GREDIR/bin</code> now use the default Java version of the operating system.</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>

### Table 8: Genero Report Designer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original data produced by an application can now be transformed within the report designer without coding. Transformations include selecting, duplicating, moving, re-ordering, pivotizing and computing aggregations.</td>
<td>See <a href="#">Report schema transformations</a> on page 775.</td>
</tr>
<tr>
<td>The Image Box can now embed the first page of a PDF file.</td>
<td>See <a href="#">Image Box</a> on page 865.</td>
</tr>
<tr>
<td>The new Toolbox object, PDFBOX, can embed an entire PDF.</td>
<td>See <a href="#">PDF Box</a> on page 864.</td>
</tr>
<tr>
<td>You can create a Spider Web chart.</td>
<td>See <a href="#">Category charts</a> and <code>drawAs</code> property.</td>
</tr>
<tr>
<td>The business graphs now use a new &quot;flat&quot; design by default, although you can still specify the &quot;classic&quot; style.</td>
<td>See <a href="#">Specifying classic or default style</a> on page 832.</td>
</tr>
<tr>
<td>If the parent object is a propagating container, the child object does not fit in the remaining space for the parent object, and you set the <code>Y-Size</code> property to <code>rest</code>, the child now expands to the maximum extent of the parent rather than just the remainder of the parent.</td>
<td>See <code>Y-size</code> property.</td>
</tr>
<tr>
<td>For <code>X-Size Adjustment</code> and <code>Y-Size Adjustment</code>, a value of <code>expandToParent</code> now causes the box to stretch as much as possible without intersecting the borders of a parent or sibling.</td>
<td>See <code>X-Size Adjustment</code> and <code>Y-Size Adjustment</code> properties.</td>
</tr>
</tbody>
</table>
Welcome to Genero Enterprise

Welcome to Genero, the software infrastructure for enterprise business application development and deployment.

When you first install Genero Enterprise and open Genero Studio, you see the Welcome page, which provides access to the projects, tutorials and samples. You can start by running the demos or by creating applications using the Quick Starts.
## What is Genero Enterprise

Genero Enterprise provides an integrated development environment that you can use for the creation of applications to deploy to desktop, web, cloud, and mobile clients.

### Genero Enterprise Components

Genero Enterprise is made up of visual tools, business and program logic components that effectively work together to quickly create applications and reports.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Genero Business Development Language (BDL)</strong></td>
<td>Genero BDL is a simple, easy-to-learn programming language for data-intensive business applications. See <a href="#">Genero Business Development Language</a>.</td>
</tr>
<tr>
<td><strong>Genero Studio</strong></td>
<td>Genero Studio is an intuitive suite of visual tools (or modules) for creating application interfaces and developing and debugging the underlying program logic. See <a href="#">Genero Studio</a>.</td>
</tr>
<tr>
<td><strong>Genero Report Writer</strong></td>
<td>Genero Report Writer is a set of programming APIs and a graphical user interface (GUI) that provide a drag-and-drop interface for the design of business reports, such as corporate or accounting documents, pre-printed forms, labels, and business graphs. See <a href="#">Genero Report Writer</a>.</td>
</tr>
<tr>
<td><strong>XML-based Abstract Presentation Layer</strong></td>
<td>Genero provides an abstract definition of the user interface as an XML tree of objects that can be manipulated at runtime by the front end client to enable GUI independence for your Genero application. See <a href="#">XML-based Abstract Presentation Layer</a>.</td>
</tr>
<tr>
<td><strong>Dynamic Virtual Machine (DVM)</strong></td>
<td>The DVM executes your application and manages database interaction and communications with client platforms. See <a href="#">DVM</a>.</td>
</tr>
<tr>
<td><strong>Open Database Interface (ODI)</strong></td>
<td>The ODI refers to the component of the Genero DVM that maps database-vendor-independent high-level code to provide native access for a variety of vendor databases. See <a href="#">ODI</a>.</td>
</tr>
<tr>
<td><strong>Genero Web Services (GWS)</strong></td>
<td>Web services enables applications built with different technologies to integrate and exchange data across varied systems and enterprises using web-based protocols. See <a href="#">GWS</a>.</td>
</tr>
<tr>
<td><strong>Genero Desktop Client (GDC)</strong></td>
<td>The GDC provides a graphical front end that displays your Genero application on Windows®, Linux™, or Apple™ desktops. See <a href="#">GDC</a>.</td>
</tr>
<tr>
<td><strong>Genero Application Server (GAS)</strong></td>
<td>The GAS provides remote access to Genero applications deployed for the GDC. The GAS includes a Web client component - the Genero Browser Client (GBC) - to support the development of database-oriented web applications. The GAS also manages Genero Web Services. See <a href="#">GAS</a>.</td>
</tr>
</tbody>
</table>

**Note:** This installation of the GAS does not include the web server component of the software, however the products are fully functional standalone and can be used for developing applications. For production systems,
you can install the web server component later using the installation packages of the full GAS products.

These components are discussed in more detail below.

**Genero Business Development Language (BDL)**

Genero BDL is a simple, easy-to-learn programming language for data-intensive business applications. Use Genero BDL to build an interactive database application, a program that handles the interaction between a user and a database.

The programming language is English-like and easy-to-learn. Executable statements enable:

- Program flow control
- Conditional logic
- Error handling
- Structured Query Language (SQL) statement support

High-level program instructions substitute for the many lines of code usually needed to handle user interaction and database manipulation. For example, the `INPUT BY NAME` instruction turns program control over to the user, and allows the user to move around the application form, entering or modifying data.

SQL statements to communicate with database servers are a part of BDL. Dynamic SQL management allows you to execute any SQL statement that is valid for your database version, in addition to those included as part of the BDL.

Built-in classes and predefined functions support BDL features and enhance rapid application development. Using the Genero classes and functions, you can manipulate your application's user interface at runtime and perform other common application tasks.

Data types supported by the language include user types, which you can be defined as synonyms of existing data types, or as shortcuts for records and array structures.

**XML-based Abstract Presentation Layer**

A Genero application presents its user interface as windows and forms. Interactive elements on the forms (such as buttons, toolbars, and menus) allow the user to trigger actions within the application.

Forms can be designed using the graphical Genero Studio Form Designer, directly in text files, or built dynamically by the application.

The XML-based Abstract Presentation Layer allows a single Genero source code stream to support different user interfaces. Genero client software, such as the Genero Desktop Client or the Genero Browser Client, display an application's interface on the client machines.

**Figure 1: How the AUI tree functions** on page 19 shows how the abstract user interface (AUI) tree is shared by the runtime system and the front end client. Their interaction during the running of the application is explained here.
Figure 1: How the AUI tree functions

- The form definition files are translated into XML documents when they are compiled into runtime form files.
- The Genero runtime system creates the AUI tree from the XML documents, and sends this information to the Genero client software.
- The Genero client software uses the AUI tree to display the application's interface on the client machine.
- When a user triggers an action, the event is transmitted to the runtime system for interpretation.
- Any changes to the user interface resulting from this action are applied to the AUI tree.
- The runtime system automatically synchronizes the copies of the AUI trees on the application server and the client machine.

**Important**: Genero BDL contains built-in classes that allow an application to modify the application's interface at runtime, dynamically changing the appearance of the application.

**Dynamic Virtual Machine**

The Dynamic Virtual Machine is the software or runtime system (fglrun) where an application's business logic is processed. It serves as a highly efficient application server that:

- Manages communications with clients.
- Supports a wide variety of architectures, including the Web and Web services.
- Optimizes performance across multiple platforms and databases.

The DVM's n-tier architecture enables the distribution of applications and databases through firewalls and across distributed networks.

Ported to multiple UNIX, Windows, and Linux® platforms, the Dynamic Virtual Machine executes the portable byte code (P-code) of Genero applications.
Open Database Interface

Using the Genero Open Database Interface architecture a Genero application can connect to database servers from different database vendors, including Oracle™, IBM™ DB2™, PostgreSQL™, SQL Server™, MySQL™, IBM® Informix™, SQLite™, and Sybase™.

Compatibility guides for writing portable SQL are provided in the *Genero Business Development Language User Guide*.

- Database drivers specific to each supported database vendor are provided as pre-linked shared libraries.
- At runtime, the DVM generates the appropriate SQL commands to be used with the target database server.

A Genero application can connect to different database servers simultaneously by issuing simple `CONNECT TO` and `SET CONNECT` instructions.

Genero Web Services

Genero support for Web Services is built in to Genero BDL.

Genero developers do not need to learn intricate programming to use Web Services; they can use simple embedded commands in BDL to create and use Web services.

Web Services work by answering requests for information and returning data in structured XML documents. As XML is simple text and Web Services can be invoked via the hypertext transfer protocol (HTTP), it does not matter what platform runs the Web Service.

Typically, web services use the Simple Object Access Protocol (SOAP) or Representational State Transfer (REST) protocols to define the communication and structure of messages.

Genero Desktop Client

The Genero Desktop Client is a front end client that displays application screens natively on your Windows, Linux, or MAC OS. The GDC also allows you to run your application through the GAS, yet deliver it locally using the GDC. Shortcuts can store the information necessary to start an application.
Figure 3: Genero Desktop Client

Genero Application Server

The Genero Application Server (GAS) is an engine that allows you to develop Genero applications for delivery to Genero front-end clients for both the desktop and the web. No changes to your Genero application source code and form program files are required; the same Genero application can be displayed in a browser or as a desktop application.

The GAS is embedded with a Web Server; it includes dispatcher and proxy processes to enable the GAS to be interfaced with a Web Server to handle requests from the Internet. For development cycle you can install the GAS
locally and use the GAS's Standalone HTTP capabilities to serve the web page. But a Web server (e.g. Apache™ httpd, lighttpd, nginx, or Microsoft™ ISAPI) must be included for deployment.

The Genero Browser Client component of the GAS supports development of both highly scalable database-oriented web applications, and enterprise applications that provide web and desktop interfaces to their functionality. You can customize and control how the Web Client renders the application to provide a more web-like interface when the application is displayed in a browser. This is done by the front end client software rendering the Genero applications for display in standard HTML browsers.

The GAS establishes and manages the communication between Genero front end client software (GDC or GBC) and the DVM. Communication between the front-end client and the GAS is handled by the Genero dispatcher, which routes requests from the Web Server to the correct GAS daemon. Several GAS daemons can be configured to load balance the requests.

The GAS also manages a pool of DVMs for Web Services applications.

**Tour Genero Studio**

Genero Studio accelerates the development and management of Genero business applications by providing tools for designing interfaces, writing and debugging programs, and teaching new developers the basics of Genero Business Development Language (BDL). Take a short tour to discover what it can do.

[Launch Genero Enterprise](#) to discover the following features and tools Genero Studio provides:
Launch Genero Enterprise

Windows® users: Select Start > Four Js Genero Evaluation Program 3.00.xx > Genero Enterprise

Linux® users: Select Desktop Menu > Four Js Genero Evaluation Program 3.00.xx > Genero Enterprise

When Genero Enterprise is launched, the Welcome Page displays a list of the available Genero projects. In this tour, you will explore Genero Enterprise by working with the Reports project. Additional demo programs are also available for you to try, see Running the Demos on page 42. At any time you can select the help icon.
in the Genero Enterprise toolbar to get additional help from the documentation.

**Project Manager**

The Project Manager helps you manage projects and their associated files. Genero Studio supports Source Code Management (SVN).

*Reports* is a pre-defined project that contains multiple applications and the corresponding source code files.

1. Choose the project *Reports* from the listing in the **Welcome Page**. A project (*Reports.4pw*) maps the relationships between the nodes in the Reports project. The structure of the project is displayed in the **Projects** view.

2. Expand the *Reports* structure tree in Projects view by clicking the + sign. The *Reports* project consists of application and library nodes that contain the files associated with the various applications in the project.
3. Expand the application **OrderReport**. Within this project, expand the listings. The project contains two virtual folders:

- **Designs** - contains the report design documents associated with the application
- **src** - contains the Genero BDL files and other files associated with the application

**Code Editor**

Code Editor is a structured, graphical editor with real time Genero syntax checking, code completion, code template management, and more. It is a programming-oriented editor designed primarily for editing Genero BDL source code, but it can handle any kind of text or languages.
1. From your expanded Reports project, double-click the file OrderReport.4gl. The Code Editor displays the file for editing, and the toolbar now has additional icons specific to Code Editor. If you wish, you can choose Window > Workspaces > Document to switch your view to one designed for document editing.

The Code Structure view on the right displays the structure of the file, listing the functions and other components. Syntax errors are marked in the editing window and listed in the Output panel (Document Errors). Select a function in the Structure panel to display the corresponding code in the editing window.
Figure 6: Genero Studio Code Editor

```plaintext
-- In this case you will need an office store database up

CONSTANT runFromFile INTEGER = FALSE
CONSTANT dataFile STRING = "OrderReport.unl"

DEFINE controlBlock PivotControlBlock

MAIN

DEFINE
    r_output STRING,
    r_filename STRING

LET r_filename="OrderReport"
LET r_output="Genero Report Viewer"
OPEN FORM f_configuration FROM "Configuration"
DISPLAY FORM f_configuration

INPUT BY NAME r_filename, r_output ATTRIBUTES (UNBUFFERED)
BEFORE INPUT
    CALL dialog.setActionActive("saveOnDisk",r_output)
ON ACTION preview
    CALL runReport(r_filename, r_output,"preview")
ON ACTION saveOnDisk
    CALL runReport(r_filename, r_output,"saveOnDisk")
```
2. Select File > Close OrderReport.4gl to close the Code Editor.

   Tip: If you changed your Workspace view, use the Studio main menu option Window > Workspaces > Normal to change it back.

Graphical Debugger

The Graphical Debugger is used to monitor the execution of an application, stopping at chosen points to examine the application's behavior, or to test different scenarios.

Breakpoints are set on program lines in the .4gl program file in Studio Code Editor, using the right-click menu:
Figure 7: Genero Studio Graphical Debugger
The **Debug** menu has commands to control the execution of the program during debugging. The program variables and their current values are displayed in the **Data** tab of the **Output** view. Additional tabs allow you to manually enter commands and use watchpoints, for example.

**Form Designer**

Form Designer is a drag-and-drop visual editor that supports the creation, editing, and layout of Genero user interfaces. The Form Designer is integrated with a **Meta-schema Manager** to simplify the creation and modification of interfaces that are database-aware.

1. Double-click *Configuration.4fd* in the *OrderReport src* folder to open the form definition file in Form Designer.

   The form definition consists of containers and widgets that are displayed in the form and listed in the **Form Structure** view. The application displays the form to the user. High-level BDL statements in the Genero BDL file *OrderReport.4gl* handle the user's input on the form. This form allows the **OrderReport** application user to select a report and its output format.

2. Select **Build > Preview** from the Studio menu to see how the form will display to the user. Select **Yes to All** if the **New connection** window is displayed. Close your previewed form.

3. The properties of a selected form object is listed in the Properties view. **Select the RadioGroup** (as shown) to see its properties in the **Properties** view.
4. Form Designer is a drag-and-drop visual editor. Select the grid around the form items and drag the bottom to make it larger. Then, re-arrange the form objects as you wish.

5. Select Build > Preview to see how your changes appear to the user.

6. Select File > Close from the Studio menu to close the form and exit Form Designer. Do not save the changed form.
Note: To examine a more complex form, select the OfficeStore project from the Welcome Page. Expand the project nodes Office Store Demo, Entities and double-click OrderForm.4fdm. Close the Form Designer before continuing with the next exercise.

Meta-schema Manager

A meta-schema is an XML file that is a central repository of the Database Metadata; the information about the tables, columns, and relations of a relational database. It also provides the appropriate default values when using the tables, columns, or relations in Genero Studio Forms or Studio-generated applications. The officestore.4db schema is used in the Reports project. Double-click on officestore.4db to explore it in the Meta-schema Manager.
Figure 9: The Meta-schema Manager

Report Writer
The Genero Report Writer is a powerful tool for the creation of business reports for mission-critical applications. The Report Writer includes:
- Genero Report Designer (GRD) for graphical report layouts.
- Genero Report Engine (GRE) to process the raw data.
- Genero Report Viewer (GRV) to render the report.

1. Double-click the **OrderReport.4rp** in the **Designs** folder in the **Projects** view to open one of the report designs used by the Order Report application.

![Image of Genero Report Designer](image-url)

*Figure 10: Genero Report Designer*
2. The **Tool Box** view allows you to drag containers and other objects onto the report design, and the **Data View** view allows you to add the report data. The **Properties** view lists the property values for a selected report object. The report's tree structure is listed in the **Report Structure** view.

3. Close the **OrderReport.4rp** document.

4. Compile and link the **OrderReport** application - right-click the node in the **Projects** view and select **Build** from the context menu. The output from the **Build** operation displays in the **Output** view.

**Figure 11: Building Order Report**

5. Run the program - right-click the **Order Report** application node and select **Execute** from the context menu.

6. The Genero Desktop Client is called automatically to display the **Configuration** form, the application's user interface:
   - Select **Order Report** from the **Report File** dropdown box
   - Select **SVG** as the **Output**
• Select **Preview**
  The Order Report is displayed in the Genero Report Viewer.

**Business Application Modeling**

Business Application Modeling is the process of modeling applications. The Business Application Modeler (BAM) is the set of modeling tools that allow you to graphically model your application. The BAM works with Studio components, such as Meta Schema Manager, Business Application Diagram, Form Designer, Report Designer, Report Data designer and Web Services data designer to:

• Create a design model (.4ba) of your application.
• Generate forms (application screens) based on the database meta-schema.
• Generate the source code for a Genero application, using the design model and generated forms as input.
The Business Application Modeler permits you to outline the behavior of your application in diagrams. It manages the Genero BDL language code for you. The code is updated as you declare new behavior in the diagrams.

The generated program:

- Allows the user to search a database table, add a new database row, update a database row, or delete a database row.
- Allows the user to retrieve a value for a form field from a list displayed in a pop-up window.

Figure 12: The Business Application Modeler (BAM)
• Populates the items for a ComboBox using values from a database table.
• Handles the master-detail relationships between tables, such as between orders and order items as shown in the example.
Figure 13: Example Order Detail Application Screen
Code Analyzer

The Code Analyzer reverse engineers your existing applications for component relationships and function call sequence. It generates diagrams, providing you with an overview of your application. Select the link between components to display a list of function calls.

Dependency Diagram - displays the complex relationships between application and its components, the dependencies between the various objects. To display the diagram, right-click the Application node or the Project node, and select Open Dependency Diagram from the menu of options:

![Example Dependency Diagram of Officestore Application](image)

Figure 14: Example Dependency Diagram of Officestore Application

Sequence Diagram - displays the application's BDL functions and how they interact with each other, which functions call and/or are called by other functions. To display the diagram for a function from an open Genero .4g1 source code file - right-click the function name and select Open Sequence Diagram from the context menu:
Integrated Help

Online help is available from within the **Genero Enterprise Program > Help** menu.

**Tip:** You can also get context help for a feature or tool you are working with by pressing the **F1** key, or selecting the **help icon**

You can also access Genero Studio Help from the **Four Js Genero Evaluation** program list:
- **Windows®** users: Select **Start > Four Js Genero Evaluation Program 3.xx.xx > Genero Studio Help**
- **Linux®** users: Select **Desktop Menu > Four Js Genero Evaluation Program 3.xx.xx > Genero Studio Help**

The complete Genero documentation sets is displayed in the **Contents** tree in the left panel of the help window as shown, which you can browse or search using the **Index** or **Search** tools.
Running the Demos

Genero Studio includes demo programs that illustrate some typical business applications.

To run a demo:

1. Launch Genero Studio.
2. From the Welcome Page, select the Tutorials & Samples tab and the demo that you want to run.
3. Right-click the application node, and select Execute.

Continue reading for a brief explanation of two of the demos.

**Office Store demo**

The Office Store demo displays orders and accounts from the included Office Store database.

Click on the link to open the OfficeStore demo. Under the Officestore Model group, execute the Orders application.

---

**Figure 17: Execute the Orders application**

You can then interact with the Office Store database by using the Orders application.
Figure 18: Using the Orders application

Reports demo

The Reports demo provides a sample reporting application with various reports design documents. For example, you can view the report data in a list or a chart.

Open the Reports demo. Under the Reports > Applications group, execute the OrderReport application.
Figure 19: Execute the OrderReport application

From the application, you can select the report file and the type of output. To start, we recommend you use the default values of *OrderReport* and *Genero Report Viewer*. Click *Preview* to see the report.

1. Select Reports demo
2. Right-click OrderReport node
Switching Genero Clients

Genero Studio installs with a set of configurations that launch your application using various Genero front ends.

Front-end options

Your default options include:

- **Desktop**: When you select the Desktop configuration, the application displays in the Genero Desktop Client. This configuration works out-of-the-box.

- **Web**: When you select the Web configuration, the application displays in the Genero Browser Client. This configuration works out-of-the-box.

- **3.00 Web (deprecated)**: When you select the Web (deprecated) configuration, the application displays in the Genero Web Client for HTML5. This configuration works out-of-the-box.
Tip: The Genero Web Client for HTML5 is provided for legacy Genero applications, and is provided for backwards compatibility only. We recommend you use the Web configuration - and the Genero Browser Client - for your web applications.

Android™

When you select the Android™ configuration, the application displays on an Android™ device.

Important: This configuration will NOT work out-of-the-box. You must complete additional configuration steps and provide an Android™ device. See Configure Genero Mobile for Android™ on page 142.

iOS Dev Client

When you select the iOS Dev Client configuration, the application displays on an iOS device.

Important: This configuration will NOT work out-of-the-box. You must complete additional configuration steps and install the Genero Mobile Development Client on your iOS device. See Display to the Genero Mobile Development Client on page 162.

Select your configuration

Select a configuration from the combobox located in the lower right corner of Genero Studio, then run your application.
Creating with Quick Starts

A quick start provides you with simple step-by-step instructions for completing specific tasks. Quick starts are located in sections where they have supporting topics. Here is a list of all the quick starts that can be found in this guide:

- **Getting Started**
  - Quick Start: Tour of Genero Studio on page 49

- **Business Application Modeling (BAM)**
  - Quick Start: Generate an application on page 206
  - Quick Start: Generating a mobile app on page 214

- **Project Manager**
  - Quick Start: Create a project on page 404

- **Form Designer**
  - Quick Start: Creating a first form on page 487

- **BAM Template Developer Guide**
  - Quick Start: Customizing templates on page 1095
Finding more information

Genero Studio includes a comprehensive documentation suite to help you understand and use the product.

All Genero documentation is installed with Genero Studio. Documentation is also available on our web site.

The Four Js Training Portal provides access to our instructor-led training calendar and our online training videos. The training videos are targeted to programmers who are new to Genero, as well as to more experienced programmers who need to catch up with Genero and its latest features.

Welcome to Genero Studio

**Genero Studio** provides a graphical integrated development environment for developing and managing Genero applications.

Genero Studio is dedicated to Genero users, to help you quickly and easily:

- Work locally or across remote resources, with multiple defined configurations. See Configuring Genero Studio on page 138.
- Design, create and maintain database meta-schemas. See Meta-schema Manager on page 343.
- Design a user interface that interacts with a database. See Creating the user interface on page 488.
- Create application forms based on database tables. See Form Designer on page 486.
- Create application code that uses the correct syntax. See Code Editor on page 445.
- Create graphical reports. See Report Writer on page 647.
- Organize projects. See Project Manager on page 399.
- Compile, run, profile, and debug Genero applications. See Building and linking programs on page 412 and Graphical Debugger on page 589.
- Use version control to manage files. See Source Code Management (SCM) on page 624.
- Automatically generate the logic and source code for a database application. See Business Application Modeling (BAM) on page 205.
- Analyze the code and generate function diagrams for existing applications. See Code Analyzer on page 482.

Getting Started

When you have installed and configured Genero Studio, you can use the quick starts to start your exploration.

- Quick Start: Tour of Genero Studio on page 49
- The Genero Studio framework on page 83
- Learning to use Genero Studio on page 125

Quick Start: Tour of Genero Studio

Use this tour to quickly become familiar with Genero Studio.

This tour assumes that you have recently installed Genero Studio and have not changed the default configurations. If configurations have been changed, the results of some steps may be different than documented.

**Note:** If you are using Genero Mobile, see the Configuring Genero Mobile section in the Genero Studio User Guide first. This will assist you in configuring a display client (iOS or Android). You can then run the OfficeStoreMobile project instead of OfficeStore to explore Genero Studio.

To begin, launch Genero Studio. From the Welcome Page, Projects tab, select the OfficeStore project. This opens the OfficeStore project file in the project view.
Run the OfficeStore demo

Follow these steps to run the OfficeStore demo using the Genero Desktop Client (GDC) and the Genero Browser Client (GBC).

1. Select the Desktop configuration.

2. Expand the **OfficeStore** project. Right-click on the **Orders** node and select **Execute**.

![Figure 22: OfficeStore sample project](image)

![Figure 23: Selecting the Desktop configuration.](image)
Figure 24: Executing a program

3. Navigate the Orders program to browse records, add a new item to an order, and modify the shipping address.
   a) Use the Next and Previous options in the Toolbar to browse the records.
   b) Click Modify to edit a record.
c) Add a new item to the order. Select an item in the item list and click **Append**. In the **Number** field enter 100.

d) Tab to the **Item Id** field and click ... to bring up a list of options. Click the **Product Name** column header to sort the rows. Find the product name *Basketball* and double-click to add this item to the order.

e) Tab to the **Quantity** field and enter 5.

f) Click on a different item. You are prompted to save your changes: select **Yes**.

Your item is added to the order.

g) Select the **Order Date** field and use the calendar to select a new date.

h) Select the **Shipping** tab. In the **Address** field enter 4, Rue Beethoven.

---

**Figure 25: Program running in GDC**

4. Select **File > Exit** to exit the **Orders** program.

Run the same **Orders** program using the Genero Browser Client (GBC).

5. In the lower right corner, change the display configuration to **Web**.
Figure 26: Change display to Web

6. Right-click on the Orders node and select Execute.
Figure 27: An application running in a browser with the default configuration for Genero Web Client

7. Navigate the Orders program in the Genero Browser Client.
8. Select File > Exit to exit the Orders program.

Explore the Debugger

Follow these steps to explore Debugger features.

1. Expand the Intermediate Files folder in the Accounts node and double-click Account.4gl to open the source in Code Editor.
2. Right-click on line 33, a function call to `AccountForm_ui_uiOpenForm()` and select **Add/Delete Breakpoint**.

A red dot, the breakpoint icon, appears in the gutter adjacent to line 33.
Figure 29: Adding a breakpoint

3. Right-click the Accounts application node and select Debug to start the application in Debug mode. When the application runs, it stops at the breakpoint, and waits.

Figure 30: Waiting at a breakpoint
4. Select **Debug > Step in** to execute the function call and open the function module, `AccountForm_ui.4gl`, in Code Editor.

Execution stops inside the called function, waiting for the next Debugger instruction.
5. Use the Code Structure view to quickly locate functions in `AccountForm_ui.4gl` for additional breakpoints:
   a) In the Code Structure view, select the `AccountForm_ui_uiDisplay()` function to display the function source in Code Editor.
   b) In Code Editor, right-click on the line containing the function header and select **Add/Delete Breakpoint**.
   c) Repeat Sub-steps a and b to create a breakpoint for the `AccountForm_ui_uiInput()` function.
6. In the Data view examine the values of local, module and global variables.

7. In the Data view, expand the **Local variables** folder, right-click on the variable `p_openMode`, and select **Add to watch**.

   Setting a watchpoint on the `p_openMode` variable will stop program execution each time the variable value changes.

   ![Image of a software interface showing a watchpoint added to a variable](image)

   **Figure 33: Adding a Watchpoint**

8. In the command line area of the Command view, type `continue` to resume program execution until the `p_openMode` watchpoint is triggered.

   Entering the `continue` command resumes program execution until a breakpoint is reached, a watchpoint is triggered, or the program terminates. The `continue` command is also available in the Debug menu and Toolbar.
Figure 34: Command view showing the results of the continue command


An application screen displays and program execution stops at the breakpoint on function AccountForm_ui_uiDisplay().

10. Use the Backtrace view to see which functions in the program have been called.
11. Select Debug > Continue.

Program execution continues until an interactive Dialog statement (DISPLAY ARRAY) switches control to the application screen, which appears exactly as if it were running outside the Debugger.
12. Interact with the program:
   a) Select Search in the Accounts window
   b) Type miller into the User Id field and select Accept.
      Control switches back to the Debugger and program execution continues until it reaches the breakpoint at function AccountForm_ui_uiDisplay().
   c) Select Debug > Next to step through the application one instruction at a time until the record for userid miller displays and control is switched back to the application screen.
   d) Select File > Exit to close the application screen and return control back to the Debugger.

13. Select Debug > Continue to terminate the application and complete the tour.
    Terminating the application also terminates the Debugger session.
vi emulator and diff tools in Code Editor

Easily switch to the vi code editor while working with text files.

1. From the OfficeStore project, expand the Orders node and Intermediate Files folder. Double-click on Order.4gl to open it in Code Editor.
2. Add a new comment anywhere in the file by typing --My comment.
3. Select Edit > VI Editing Mode. Use vi commands to navigate the file and remove the comment you added in the prior step.
4. Notice that the file is automatically in diff mode.

Explore Source Code Management

Maintain and share project files with Genero Source Code Management (SCM)

Before you begin, a Subversion client must be installed on your local machine as described in What is the SCM module? on page 624. In addition, you must have access to a Subversion repository containing the OfficeStore sample to perform the steps below.

The steps in this tour assume that the files for the OfficeStore sample project have been stored in an SVN Repository.

1. Use the Checkout option from the SCM menu to checkout the OfficeStore project files from the repository to your checkout directory.

The SCM Checkout wizard steps you through the process of checking out project files and prompts you to select a project file (4pw) to open in Project Manager.

Figure 37: The SCM Checkout wizard
2. Select File > New > Genero Files > Style (.4st), and save the new styles file as account.4st in the Entities node.

Adding a file to the project automatically performs an SVN add, which uploads and adds the file to the repository on your next SVN commit. The Entities folder displays a red exclamation mark to indicate uncommitted changes to the directory.

Figure 38: Adding a file to a project managed by SCM

3. Select Window > Views > SVN Status, navigate to your checkout directory and select Add mode.

Selecting Add mode lists the newly added styles file as well as any other unversioned files present in the directory.
4. Ensure the Select option is checked and select the Commit... button at the top of the view to invoke the SVN Commit dialog.

5. Enter New styles file in the Commit Comments area and select the Commit button to commit the new file to the repository.

6. Enable the SVN needs-lock property on the account.4st file to prevent commit conflicts.
   Setting the needs-lock property requires a user to lock the file before modifying it.
   a) Right-click on the account.4st file in the Projects view and select SCM > Properties.
   b) Select the + at the top of the Property view to launch the Add SVN Property dialog.
   c) Select svn:needs-lock from the selection list, select OK and close the SVN Properties dialog.
7. Lock and edit the account.4st file.
   
a) In Project Manager, right-click on the account.4st file and select SCM > lock.

b) Enter testing locks in the Lock files comment area and select Lock.

c) Select the SVN Locks button in the SVN view at the bottom of the screen to view information about all locked files in the checkout directory.

d) Open the account.4st file in Code Editor, delete lines two through nine and save the changes.
   Deleting the lines will create many errors as indicated by the red icons in the Code Editor gutter.

Figure 40: Setting the needs-lock property
8. Select the `account.4st` file in the SVN Status view and commit the changes (use style edits for the Commit Comment). Note that committing the file releases the lock. You may be prompted to reload the modified file when you commit. Respond by selecting **Reload**.

9. Right-click the `account.4st` file in the Project view and select **SCM > Show Log** to add and open the SVN Log view at the bottom of the screen.

10. Right-click on the entry for Revision 4 in the SVN Log and select **Diff** to call the Diff utility to compare the selected revision with the previous revision.

   You can also use ctrl-click to select specific revisions you wish to compare with the Diff utility.
11. Right-click on the entry for Revision 4 in the SVN Log view and select **Blame** to see inline annotations for each change, including the author and revision number.

---

**Figure 42: Performing a Diff**
12. Revert back to an error-free version of the `account.4st` file.
   
a) Right-click on the entry for Revision 3 and select **Revert to this Revision**... to launch the SVN Merge/Revert dialog.

b) Accept the defaults in the SVN Merge/Revert dialog and select **Merge/Revert**, then **Finish** in the Merge/Revert result window.

The `account.4st` file reverts back to the selected revision and displays in Code Editor with the deleted lines from Revision 4 replaced and highlighted.
Figure 44: Reverting to a previous revision

13. Delete the account.4st file to complete the tour.
   a) Close the account.4st file in the Code Editor view.
   b) Right-click the account.4st file in Project Manager and select Delete from Disk.
   c) Select the account.4st file in the SVN Status view and select Commit... (enter SVN test for the Commit Comment).

Explore database meta-schemas

Follow these steps to interact with a database meta-schema for developing forms, reports, and services.

1. From the Welcome Page, Projects tab, select the OfficeStore project. This opens the OfficeStore project file in the project view.
2. Expand the project and find the Databases node.
4. Note the layout of the tables and columns and the defined join relationships between them. Zoom in and out of the diagram by holding down the Ctrl key while scrolling with the mouse.
5. Select File > Print Preview. Note that you can print any diagram in a variety of ways.
6. Find the account table. Select the lastname column and modify its column length from 80 to 120.
7. Save your changes. Note that the diagram reflects that the schema is different from the database by flagging the modified column.

8. **Generate a script** to update the database with your change to the meta-schema. Select **Database > Generate Database Update Script**. Select OK and note that a 4gl file is created for you to use or expand upon as needed. Note the additional options in the Database menu allow you to also update a schema from the database as well as compare two schemas.

**Explore forms**

Follow these steps to explore the visual tools for user interface development.

Explore a form, then create a new form

1. Expand the **OfficeStore** project and find the **Entities** node.
2. Expand the **Entities** node and double-click on the **OrderForm.4fdm** file.
   
   The **OrderForm.4fdm** file is a form and opens in the Form Designer.

![Figure 45: The OrderForm.4fdm file](image)

3. Customize your working environment to best suit you.
   
   All panels can be re-sized with the mouse or closed and reopened with **Window > Views**. Try pressing `Alt-F11` to toggle to Document editing. Select `Alt-F11` again to return to normal view. Double-click on the title of the Form Structure view to undock it. Double-click again to re-dock it into its last position.

4. Scroll down in the Form Structure view to the Form section and select a form item.
   
   The item is also selected in the form design.
5. Use the **Form from Database** wizard to create a new form:
   a) Select **File > New > Genero Files > Form from Database (.4fd)**.
b) Select the **officestore** database and the **account** table (use the double arrow to select all columns in the account table for the form).
c) Select Finish.

The form is now ready to use or modify in Form Designer. Additional table columns, container, and widgets can be added.
Figure 49: Form created from the Form from Database wizard

6. Select the Records tab at the bottom of the window.
   The Records tab represents the data set for your form. Additional records can be added here or in the design.

7. Right-click on the record and select Add Field.
   A field is added to the record with default values.

Figure 50: Adding a new field to a form record
8. Switch to the Form tab.
   Note that the new field has been placed on the form in the top left. This field’s properties and placement can be modified.

9. Right-click on the new field and select **Convert Widget > FFLabel** to change the field to a form field label.

![Figure 51: The Convert Widget menu](image)

**Figure 51: The Convert Widget menu**

10. In the properties for the new field, change its color value from Black to Blue.
    The color changes on the form design.
Figure 52: The color property

11. Select **Build > Preview** to get an idea about how the form will render in the Desktop client.

12. Close the form preview and the untitled form (do not save it) to complete the tour.

**Analyze code**

Follow these steps to learn how to reverse engineer an application with Dependency and Sequence diagrams, resources provided by Genero Code Analyzer.

1. Right-click the **OfficeStore Group node** and select **Open Dependency Diagram**.

   Dependency diagrams display a graphical view of the complex relationships between components of a project and can be opened at the group or application level.
2. Right-click the **Entities node** in the Dependency diagram and select **Expand** from the context menu.

The Entities node expands to display all the sub-components and the relationships between them. Use Ctrl-mouse wheel to zoom in and out.
3. Use a Dependency diagram filter to focus on the dependencies between components of the Accounts application:
   a) Right-click anywhere in the Dependency diagram margins and select **Filter Items...**
   b) In the **Select items to filter** dialog, deselect **Orders**.
   c) Press **OK**.
4. Select the link between `AccountForm_ui` and `AccountForm_uidata` to display associated function calls in the Function calls view.

Details about function calls between the selected modules are shown in the Function calls view and the project structure displays as a tree in the Dependency Diagram Structure view.
5. Right-click on the `AccountForm_ui_uiInput` function in the Function Calls or Dependency Diagram Structure view and select Open Sequence Diagram.

The diagram shows the logic flow of `AccountForm_ui_uiInput`, with the starting point indicated by the stick-figure representing the user who interacts with the application. The boxes represent functions in the `AccountForm_ui.4g1` module, and the sequence is indicated by the order in which the boxes are listed. Plus/minus signs on each box allow you to display or hide sub calls.
6. Right-click the box for the `AccountForm_ui_recAccount_setDefaultValues` function and select `Show Sub Calls`.

The box is expanded to show the `AccountForm_ui_recAccount_clearCascade()` subcall.
7. Close the AccountForm_ui.uiInput Sequence diagram tab and OfficeStore.4pw Dependence diagram tab in the Code Editor view to complete the tour.

**Generate code**

Genero Studio includes the Business Application Modeler (BAM) for generating business applications.

See Quick Start: Generate an application on page 206 or Quick Start: Generating a mobile app on page 214.

**The Genero Studio framework**

When a Genero Studio module is launched, the framework is displayed and all other windows and views, menus, Toolbars, and icons are contained within.

- The Welcome Page on page 84
- Modules on page 85
- Menus and Toolbars on page 96
- Views on page 116
Tip: In addition to using the Close icon (a red "X"), you can close a tab in the Central Workspace by clicking the mouse wheel on the tab label.

The Welcome Page

The Welcome Page is displayed when Genero Studio is launched. From the Welcome Page, open a recent or sample project or configure the environment or database connection.

Open the Welcome Page by selecting **Tools > Welcome Page**. The Welcome Page is the first tab in the document workspace.

Figure 59: The Genero Studio framework

Figure 60: Welcome Page tab
Tabs

Tabs organize the Welcome Page into sections.

Projects
Open a recently opened Project (4pw) or document. See Project Manager on page 399.

Tutorials and Samples
Sample files distributed with Genero Studio; contains links to open the sample projects. See Samples directory on page 126.

News
RSS feed with news about Genero.

Support
Access to support and training videos.

Customizing the Welcome Page

Select Tools > Preferences > Welcome Page to change what is on the Welcome Page.

Show Welcome Page at start up
Uncheck if you do not wish this page to display.

Template
Select the template for the Welcome Page.

Clear
Clear all stored cookies.

Visible Sections
Check the sections that you wish to be displayed. Enable RSS feeds with the RSS Section checkbox.

Parameters
Use the edit button to edit the feeds used on the page and to add or edit RSS locations.

Modules

Genero Studio is made up of modules managing the design, development, and execution of Genero applications.

A module is also commonly referred to as a plug-in.

- BAM - Application Modeling and Code Generation on page 86
- Code Analyzer on page 86
- Code Editor on page 89
- Debugger on page 90
- Diff on page 91
- File Browser on page 92
- Form Designer on page 93
- Meta-schema Manager on page 94
- Project Manager on page 94
- Report Writer on page 95
- Source Code Management on page 95
BAM - Application Modeling and Code Generation
Genero Studio Business Application Modeling (BAM) is a graphical design tool to generate the code that extracts data and creates data models in BDL. It automatically generates the logic and source code for a database application to query, add, update and delete rows in database tables. BAM can generate desktop, web, and mobile applications.

Figure 61: Business Application Modeling (BAM)
Launch by creating or opening a BAM Project from the File menu or by opening a new or existing Business Application Model diagram from the File menu or Project view.

See Business Application Modeling (BAM) on page 205 for more details.

Code Analyzer
The Code Analyzer reverse engineers existing applications and can generate diagrams to provide an overview of the application.

The Sequence Diagram visually displays the flow of your application logic. It shows how the functions of the application call and/or are called by other functions.
Figure 62: Sequence Diagram

The Dependency Diagram displays a graphical view of the complex relationships between the various pieces of a project. It shows the components that depend on other components, and/or have components that depend on them.
Figure 63: Dependency Diagram

See Code Analyzer on page 482 for more details.
**Code Editor**

*Code Editor* is a programming-oriented editor. In addition to editing source code, it can handle any kind of text as well as languages such as 4GL and XML. Smart editing features like auto-completion, code templates, and robust search make coding easier and more efficient.

![Figure 64: Code Editor](image)

Use **File > New** or **File > Open** to open a text file.

See *Code Editor* on page 445 for more details.
**Debugger**

The *Graphical Debugger* provides a graphical interface to test and control the behavior of a Genero application.

![Image of the Graphical Debugger](image)

**Figure 65: Debugger**

Launch with the *Debug* menu options.

See *Graphical Debugger* on page 589 for more details.
Diff

The Diff tool compares two files: a read-only base copy of the file and a working copy. It is integrated into Code Editor.

![Figure 66: Vertical Dual Diff View display mode](image)

Launch with Tools > Diff.

See Using the Diff tool on page 451 for more details.
**File Browser**

*File Browser* is a tool to navigate, open, delete or rename files, and to create new folders or files on a file system.

**Figure 67: The Files view**

Launch with **Tools > File Browser** or from the Project Manager **Files** tab.

See **File Browser** on page 588 for more details.
Form Designer

*Form Designer* is a visual editor that supports the creation, editing, and layout of Genero forms in Genero Studio.

![Figure 68: Form Designer](image)

Use **File > New** or **File > Open** to create or open a Genero form file.

See [Form Designer](#) on page 486 for more details.
Meta-schema Manager

The *Meta-schema Manager* is a visual tool used to design, create and maintain database meta-schema files.

![Image of Meta-schema Manager](image1)

**Figure 69: Meta-schema Manager**

Use *Database > New Schema* or *File > Open* to open an existing schema.

See *Meta-schema Manager* on page 343 for more details.

Project Manager

*Project Manager* is a tool to manage the organization and build of executables from the program's source files.

![Image of Project Manager](image2)

**Figure 70: The Projects view**
Use **File > New** or **File > Open** to open project file.

See [Project Manager](#) on page 399 for more details.

**Report Writer**

![Figure 71: Genero Report Designer](image)

Launch by opening a new or existing Genero Report Design document from the **File** menu or by opening a Genero Report Design document in the Project view.

See [Report Writer](#) on page 647 for more details.

**Source Code Management**
Genero Source Code Management (SCM) enables collaborative sharing and maintaining of the files in Genero projects.

![Figure 72: Project view with status icons](image)
See Source Code Management (SCM) on page 624.

**Web Services**
Web services can be called from a Genero application. The Web Service Wizard guides you through the process of adding a Web service. Web services can also be generated from the Business Application Modeling tools.

See Web Services on page 1075 and Add Web services on page 250 for more details.

**Menus and Toolbars**
Menus and Toolbars are constructed dynamically depending on the context and the currently active module.

- File menu on page 96
- Edit menu on page 101
- Search menu on page 104
- View menu on page 104
- Projects menu on page 105
- Diff menu on page 106
- Build menu on page 108
- Debug menu on page 109
- Database menu on page 110
- SCM menu on page 111
- Tools menu on page 113
- Windows menu on page 115
- Help menu on page 116

**Customize menus and toolbars**
The menus and toolbars displayed in Genero Studio are fully customizable.

You can add, modify, and delete toolbars, menus and corresponding accelerators using Tools > Preferences > User Interface.

Reorganize the layout of the Toolbar by dragging and dropping a Toolbar to a new location within the Toolbar region or to float on the Genero Studio framework. Right-click to display a context menu for a selected item.

**File menu**
The icon, menu name, shortcut key, and description of each option in the File menu.

- The File New menu on page 98
- Save / Save As / Save All on page 100
- Open from a different version on page 101

**Table 11: The File menu**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator (Shortcut)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![File icon]</td>
<td>Open</td>
<td>Ctrl+O</td>
<td>Open a file. See also Open from a different version on page 101.</td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator (Shortcut)</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><img src="image" alt="Open as Text..." /></td>
<td>Open as Text...</td>
<td></td>
<td>Opens the selected file as a text document.</td>
</tr>
<tr>
<td><img src="image" alt="Open with Encoding..." /></td>
<td>Open with Encoding...</td>
<td></td>
<td>Allows you to open a file as a text document with a specific encoding. The chosen encoding only applies when opening plain text files. The encoding declared in XML files always takes precedence.</td>
</tr>
<tr>
<td><img src="image" alt="Close file" /></td>
<td>Close filename</td>
<td>Ctrl+F4</td>
<td>Close the open file identified by <code>filename</code>.</td>
</tr>
<tr>
<td><img src="image" alt="Close Welcome Page" /></td>
<td>Close Welcome Page</td>
<td>Ctrl+F4</td>
<td>Close the Welcome Page.</td>
</tr>
<tr>
<td><img src="image" alt="Close Project" /></td>
<td>Close Project</td>
<td></td>
<td>Close project.</td>
</tr>
<tr>
<td><img src="image" alt="Checkout..." /></td>
<td>Checkout...</td>
<td></td>
<td>Checkout from SVN repository. See Checkout files on page 625.</td>
</tr>
<tr>
<td><img src="image" alt="Save file" /></td>
<td>Save filename</td>
<td>Ctrl+S</td>
<td>Save current file. See Save / Save As / Save All on page 100.</td>
</tr>
<tr>
<td><img src="image" alt="Save file as..." /></td>
<td>Save filename as...</td>
<td>Ctrl+Alt+S</td>
<td>See Save / Save As / Save All on page 100.</td>
</tr>
<tr>
<td><img src="image" alt="Save all" /></td>
<td>Save all</td>
<td>Ctrl+Shift+S</td>
<td>Save all unsaved files. See Save / Save As / Save All on page 100.</td>
</tr>
<tr>
<td><img src="image" alt="Import text form (.per)" /></td>
<td>Import text form (.per)</td>
<td>Ctrl+I</td>
<td>Import a text form to Form Designer format. See Importing .per files on page 493.</td>
</tr>
<tr>
<td><img src="image" alt="Import Project Files" /></td>
<td>Import Project Files</td>
<td></td>
<td>See Import files into the diagram from the project on page 275.</td>
</tr>
<tr>
<td><img src="image" alt="Export as Image..." /></td>
<td>Export as Image...</td>
<td></td>
<td>Exports a diagram to an image file. This option opens a dialog for configuring the image file path, format (such as png and jpg), and size.</td>
</tr>
<tr>
<td><img src="image" alt="Print..." /></td>
<td>Print...</td>
<td>Ctrl+P</td>
<td>Print a file. Opens the Print dialog, to print the file using the options available to your operating system printers.</td>
</tr>
<tr>
<td><img src="image" alt="Print Preview..." /></td>
<td>Print Preview...</td>
<td></td>
<td>Preview file. Opens the Print preview dialog. Displays a preview of the printed file in a preview window. For diagrams, scaling and number of pages to use to print can be configured. See Print preview dialog on page 122.</td>
</tr>
</tbody>
</table>
### Getting Started

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator (Shortcut)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Poster Printing Setup...</td>
<td></td>
<td>Used to print diagrams, it opens a dialog for configuring the scaling and number of pages to use to print. This action is available from Code Analyzer, Report Writer, Business Application Modeling, and Meta-schema Manager.</td>
</tr>
<tr>
<td></td>
<td>Recent documents</td>
<td></td>
<td>List of recent opened documents.</td>
</tr>
<tr>
<td></td>
<td>Recent projects</td>
<td></td>
<td>List of recently opened projects</td>
</tr>
<tr>
<td></td>
<td>Quit</td>
<td></td>
<td>Quit Genero Studio</td>
</tr>
</tbody>
</table>

### The File > New menu

**File > New** creates a new file. When created, the corresponding Genero Studio component is opened. For example, Code Editor is launched for new 4gl files.

### Table 12: File > New options

<table>
<thead>
<tr>
<th>Category</th>
<th>Section</th>
<th>Item Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genero BAM Desktop</td>
<td>Project</td>
<td>• BAM Desktop Project (4pw) - A BAM project for Desktop applications with an underlying file structure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Simple BAM Desktop Project (4pw) - A BAM project for Desktop applications, without underlying file structure.</td>
</tr>
<tr>
<td></td>
<td>Application Modeling</td>
<td>• Business Application diagram (4ba) - application design diagram, used in application generation</td>
</tr>
<tr>
<td></td>
<td>Database</td>
<td>• DB Schema (4dbx) - database schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DB Schema from Database (4dbx) - extracts schema from database</td>
</tr>
<tr>
<td></td>
<td>Sources</td>
<td>Forms and programs for generated code.</td>
</tr>
<tr>
<td></td>
<td>Resources</td>
<td>• Action defaults (4fd) - a Genero action defaults file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Toolbar (4tb) - a Genero Toolbar definition file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Top Menu (4tm) - a Genero top menu definition file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Style (4st) - a Genero Style definition file</td>
</tr>
<tr>
<td>Genero BAM Mobile</td>
<td>Project</td>
<td>• BAM Mobile Project (4pw) - A BAM project for Mobile applications with an underlying file structure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Simple BAM Mobile Project (4pw) - A BAM project for Desktop applications, without underlying file structure.</td>
</tr>
<tr>
<td>Category</td>
<td>Section</td>
<td>Item Type</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Application Modeling</td>
<td></td>
<td>• Business Application diagram (4ba) - application design diagram, used in application generation</td>
</tr>
<tr>
<td>Database</td>
<td></td>
<td>• DB Schema (4dbx) - database schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DB Schema from Database (4dbx) - extracts schema from database</td>
</tr>
<tr>
<td>Sources</td>
<td></td>
<td>Forms and programs for managed code</td>
</tr>
<tr>
<td>Resources</td>
<td></td>
<td>• Action defaults (4fd) - a Genero action defaults file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Style (4st) - a Genero Style definition file</td>
</tr>
<tr>
<td>Genero</td>
<td>Project</td>
<td>• Desktop Project (.4pw) - project for Desktop applications with an underlying file structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mobile Project (.4pw) - project for Mobile applications with an underlying file structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Simple project (.4pw)</td>
</tr>
<tr>
<td>Database</td>
<td></td>
<td>• DB Schema (4db) - database schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DB Schema from Database (4db) - extracts schema from database</td>
</tr>
<tr>
<td>Sources</td>
<td></td>
<td>• Source (4gl) - a Genero BDL source code module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Form (4fd) - a Genero Studio form definition file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Form from database (4fd) - calls the Form from Table wizard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Form as text (per) - a Genero BDL form definition file in text format</td>
</tr>
<tr>
<td>Resources</td>
<td></td>
<td>• Action defaults (4fd) - a Genero action defaults file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Toolbar (4tb) - a Genero Toolbar definition file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Topmenu (4tm) - a Genero Topmenu definition file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Startmenu (4sm) - A Genero Start Menu definition file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Localized String (str) - Genero Localized String file</td>
</tr>
<tr>
<td>Report from Template</td>
<td>Templates</td>
<td>A list of predefined report templates that you can use to quickly define a report. For further information, see Create and manage report templates on page 986.</td>
</tr>
<tr>
<td>Reports</td>
<td>Report Data</td>
<td>• Report from database (4gl) - to generate BDL code to extract data from database</td>
</tr>
</tbody>
</table>
## Getting Started

<table>
<thead>
<tr>
<th>Category</th>
<th>Section</th>
<th>Item Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report Designs</td>
<td></td>
<td>• Empty report <em>(4rp)</em> - a blank report design document</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• List report <em>(4rp)</em> - template of a report design document formatted for a list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For further information, see <a href="#">Create a report design document</a> on page 777.</td>
</tr>
<tr>
<td>Report Templates</td>
<td></td>
<td>• Empty report template <em>(.4rt)</em> - See <a href="#">Create a new report template</a> on page 992.</td>
</tr>
<tr>
<td>Report Schema</td>
<td></td>
<td>• Template Schema Definition <em>(.rsd)</em> - a file used to manually add the structure of the fields and groups to a report template. For further information, see <a href="#">Report template schema definition file <em>(.rsd)</em></a> on page 993.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Schema Transformation <em>(.rst)</em> - a schema file that provides access to data from pre-defined sources. For further information, see <a href="#">Report schema transformations</a> on page 775.</td>
</tr>
<tr>
<td>Web / AS Sources</td>
<td></td>
<td>• HTML <em>(html)</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• XHTML <em>(xhtml)</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CSS <em>(css)</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SCSS <em>(.scss)</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• JavaScript™ <em>(js)</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• JSON <em>(.json)</em></td>
</tr>
<tr>
<td>Configuration</td>
<td></td>
<td>• Application Configuration <em>(.xcf)</em> - for further information, see <a href="#">Application configuration files and Project Manager</a> on page 411</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Web Service Configuration <em>(.xcf)</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Genero Archive MANIFEST - for further information, see <a href="#">Create a Genero package node</a> on page 1171.</td>
</tr>
<tr>
<td>Tip:</td>
<td></td>
<td>To have the MANIFEST file be recognized as XML, use the View as XML option.</td>
</tr>
<tr>
<td>SOA Services</td>
<td></td>
<td>• Soap Server <em>(4gl)</em> - create a 4gl application and server stub from <em>wsdl</em> file</td>
</tr>
<tr>
<td>Other files</td>
<td>New Files</td>
<td>• With no extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text <em>(.txt)</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• XML <em>(.xml)</em></td>
</tr>
</tbody>
</table>

### Save / Save As / Save All

When you save a file for the first time, or select **File > Save As**, the **Save as** dialog opens.

If a project is open, you can also choose to create a link to the file in a group node or virtual folder.
Path

Specify where the file should be saved in the file system. Click the magnifying glass icon to browse for the path and enter the file name.

Insert the file in the project

If checked, you can choose the application or library node, or a virtual folder, from the list that is displayed in the dialog. Use the plus icons to expand the list.

If a project is not open, the file can be saved in the file system only. No link can be created in a group node.

Save in

Select the path where the file should be saved in the file system.

Filename

Enter the name of the file.

Save as type

Change the type of the file, if desired.

If you have previously saved the file, choosing File > Save saves it again with the same parameters. No dialog opens. The Save All option saves the project and its contents.

Open from a different version

If you open a file that was saved with a different version of Genero Studio, you can decide how to handle the file.

If the file was saved with an older version, you can open it in the older version, or convert and open it in the current version of Genero Studio.

If the file was saved with a newer version, you cannot open it in the current version of Genero Studio.

The dialog options include:

Open in new instance

Displays a selector to find the version of Genero Studio associated with the file version.

Important: Open in New Instance is not available on macOS™.

Open

Converts the file in memory and opens it. If saved, it will be saved to the latest file format.

Note: Not available if the file was saved with a newer version of Genero Studio.

Cancel

Cancels opening the file.

Edit menu

The icon, menu name, shortcut key, and description of each option in the Edit menu.

Table 13: The Edit menu

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator (Shortcut)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="icon" alt="Edit" /></td>
<td>Edit</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="icon" alt="Undo" /></td>
<td>Undo</td>
<td>Ctrl+Z</td>
<td>Undo the last action(s).</td>
</tr>
<tr>
<td><img src="icon" alt="Redo" /></td>
<td>Redo</td>
<td>Ctrl+Y</td>
<td>Redo the last action(s).</td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator (Shortcut)</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td><img src="cut.png" alt="" /></td>
<td>Cut</td>
<td>Ctrl+X</td>
<td>Cut to clipboard.</td>
</tr>
<tr>
<td><img src="copy.png" alt="" /></td>
<td>Copy</td>
<td>Ctrl+C</td>
<td>Copy to clipboard.</td>
</tr>
<tr>
<td><img src="paste.png" alt="" /></td>
<td>Paste</td>
<td>Ctrl+V</td>
<td>Paste from clipboard.</td>
</tr>
<tr>
<td><img src="delete.png" alt="" /></td>
<td>Delete</td>
<td>Del</td>
<td>Delete item.</td>
</tr>
<tr>
<td><img src="selectall.png" alt="" /></td>
<td>Select All</td>
<td>Ctrl+A</td>
<td>Select all on area with focus.</td>
</tr>
<tr>
<td><img src="search.png" alt="" /></td>
<td>Search/Replace</td>
<td>See <a href="#">Search and replace</a> on page 448.</td>
<td></td>
</tr>
<tr>
<td><img src="find.png" alt="" /></td>
<td>Find</td>
<td>CTRL+F</td>
<td>Find in current file.</td>
</tr>
<tr>
<td><img src="findinfiles.png" alt="" /></td>
<td>Find in Files</td>
<td>CTRL +SHIFT+F</td>
<td>Find in selected files.</td>
</tr>
<tr>
<td><img src="findprevious.png" alt="" /></td>
<td>Find Previous</td>
<td>SHIFT+F3</td>
<td>Find previous occurrence.</td>
</tr>
<tr>
<td><img src="findnext.png" alt="" /></td>
<td>Find Next</td>
<td>F3</td>
<td>Find next occurrence.</td>
</tr>
<tr>
<td><img src="replaceprevious.png" alt="" /></td>
<td>Replace previous</td>
<td></td>
<td>Replace previous occurrence.</td>
</tr>
<tr>
<td><img src="replacenext.png" alt="" /></td>
<td>Replace next</td>
<td>F3</td>
<td>Replace next occurrence.</td>
</tr>
<tr>
<td><img src="replaceall.png" alt="" /></td>
<td>Replace all</td>
<td></td>
<td>Replace all occurrences.</td>
</tr>
<tr>
<td><img src="stop.png" alt="" /></td>
<td>Stop search</td>
<td></td>
<td>Stop current search</td>
</tr>
<tr>
<td><img src="clear.png" alt="" /></td>
<td>Clear all Search Results</td>
<td></td>
<td>Clear all search results.</td>
</tr>
<tr>
<td><img src="bookmarks.png" alt="" /></td>
<td>Bookmarks</td>
<td></td>
<td>See <a href="#">Bookmarks</a> on page 447.</td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator (Shortcut)</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------------</td>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Toggle Bookmark</td>
<td>CTRL+B, CTRL+B</td>
<td>Toggle bookmark at location. See Bookmarks on page 447.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Previous Bookmark</td>
<td>CTRL+B, CTRL+P</td>
<td>Go to previous bookmark</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Next Bookmark</td>
<td>CTRL+B, CTRL+N</td>
<td>Go to next bookmark</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Delete all Bookmarks</td>
<td>CTRL+B, CTRL+L</td>
<td>Go to next bookmark</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Open Line</td>
<td>Ctrl+Shift +L</td>
<td>Open a new line</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Delete Line/Word</td>
<td></td>
<td>Delete current word or line</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Selection</td>
<td></td>
<td>Select current word or line. See Selection keymap on page 473.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Go To</td>
<td>Ctrl+G</td>
<td>Go to location in file</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Comment</td>
<td>Ctrl+K</td>
<td></td>
</tr>
<tr>
<td>![Icon]</td>
<td>Toggle Line Comment</td>
<td>Ctrl+K, Ctrl +K</td>
<td>Toggle line comment</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Block Comment</td>
<td>Ctrl+K, Ctrl +C</td>
<td>Comment section of code</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Block Uncomment</td>
<td>Ctrl+K, Ctrl +U</td>
<td>Uncomment section of code</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Completion</td>
<td>Ctrl+Space</td>
<td>Completes a line of code or prompts for a valid keyword in the syntax. See Auto completion (Ctrl +Space) on page 447.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Auto Complete</td>
<td>Ctrl+Space</td>
<td>Completes a line of code or prompts for a valid keyword in the syntax. See Auto completion (Ctrl +Space) on page 447.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Use Template</td>
<td>Ctrl+T</td>
<td>Select template from list. Code templates (Ctrl+T) on page 447.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Case</td>
<td>Alt+Shift +U</td>
<td>Change selection to upper case</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Upper Case</td>
<td>Alt+Shift +U</td>
<td>Change selection to upper case</td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator (Shortcut)</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------</td>
<td>------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><img src="image" alt="Lower Case" /></td>
<td>Lower Case</td>
<td>Alt+Shift+L</td>
<td>Change selection to lower case.</td>
</tr>
<tr>
<td><img src="image" alt="Uppercase Word" /></td>
<td>Uppercase Word</td>
<td>Ctrl+Alt+Shift+W</td>
<td>Change word to upper case.</td>
</tr>
<tr>
<td><img src="image" alt="Lowercase Word" /></td>
<td>Lowercase Word</td>
<td>Ctrl+Alt+W</td>
<td>Change word to lower case.</td>
</tr>
<tr>
<td><img src="image" alt="Toggle Case" /></td>
<td>Toggle Case</td>
<td>Ctrl+U</td>
<td>Toggle case.</td>
</tr>
<tr>
<td><img src="image" alt="Convert Keywords to Uppercase" /></td>
<td>Convert Keywords to Uppercase</td>
<td>Ctrl+Shift+T</td>
<td>Convert keywords to upper case.</td>
</tr>
<tr>
<td><img src="image" alt="Strip Trailing Spaces" /></td>
<td>Strip Trailing Spaces</td>
<td>Ctrl+Shift+T</td>
<td>Delete all spaces at the end of lines.</td>
</tr>
<tr>
<td><img src="image" alt="Format and Indent" /></td>
<td>Format and Indent</td>
<td>Ctrl+Shift+I</td>
<td>Format the document by indenting XML elements and removing extra spaces. This option is enabled only for XML documents.</td>
</tr>
<tr>
<td><img src="image" alt="Convert to" /></td>
<td>Convert to</td>
<td></td>
<td>Convert text to Windows, UNIX, or Mac® format.</td>
</tr>
<tr>
<td><img src="image" alt="VI Editing Mode" /></td>
<td>VI Editing Mode</td>
<td>Ctrl+Shift+E</td>
<td>Use VI commands to edit the file. See vi emulator and diff tools in Code Editor on page 63.</td>
</tr>
</tbody>
</table>

**Search menu**

The icon, menu name, shortcut key, and description of each option in the Search menu.

**Table 14: The Search menu (Genero Report Designer specific)**

**View menu**

The icon, menu name, shortcut key, and description of each option in the View menu.

**Table 15: The View menu**
### Table 16: The Projects menu

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator (Shortcut)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="" alt="Zoom out icon" /></td>
<td>Zoom out</td>
<td>-</td>
<td>Zoom out.</td>
</tr>
<tr>
<td><img src="" alt="Zoom in icon" /></td>
<td>Zoom in</td>
<td>+</td>
<td>Zoom in.</td>
</tr>
<tr>
<td><img src="" alt="Actual size icon" /></td>
<td>Actual size</td>
<td></td>
<td>Restore default zoom.</td>
</tr>
<tr>
<td><img src="" alt="Split Horizontal icon" /></td>
<td>Split Horizontal</td>
<td>Ctrl+Alt+H</td>
<td>Split document horizontally. See <em>Split a document</em> on page 448</td>
</tr>
<tr>
<td><img src="" alt="Split Vertical icon" /></td>
<td>Split Vertical</td>
<td>Ctrl+Alt+V</td>
<td>Split document vertically. See <em>Split a document</em> on page 448</td>
</tr>
<tr>
<td><img src="" alt="Unsplit icon" /></td>
<td>Unsplit</td>
<td>Ctrl+Alt +Shift+N</td>
<td>Unsplit.</td>
</tr>
<tr>
<td><img src="" alt="Unsplit all icon" /></td>
<td>Unsplit all</td>
<td>Ctrl+Alt +Shift+A</td>
<td>Unsplit all.</td>
</tr>
<tr>
<td><img src="" alt="Fold icon" /></td>
<td>Fold</td>
<td>Ctrl+-</td>
<td>Fold section of text. See <em>Folding text</em> on page 446.</td>
</tr>
<tr>
<td><img src="" alt="Unfold icon" /></td>
<td>Unfold</td>
<td>Ctrl++</td>
<td>Unfold section of text.</td>
</tr>
<tr>
<td><img src="" alt="Fold all icon" /></td>
<td>Fold all</td>
<td>Ctrl+Shift+-</td>
<td>Fold all text. See <em>Folding text</em> on page 446.</td>
</tr>
<tr>
<td><img src="" alt="Unfold all icon" /></td>
<td>Unfold all</td>
<td>Ctrl+Shift++</td>
<td>Unfold all text.</td>
</tr>
<tr>
<td><img src="" alt="Languages icon" /></td>
<td>Languages</td>
<td></td>
<td>Specify the language that Code Editor uses for formatting and error marking.</td>
</tr>
</tbody>
</table>

**Projects menu**

The table shows the icon, menu name, shortcut key, and description of each option in the Projects menu.

The Projects menu is visible when an item in the Projects view has focus.

---

**Table 16: The Projects menu**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator Description (Shortcut)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="" alt="Help icon" /></td>
<td>Help</td>
<td></td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator Description (Shortcut)</td>
</tr>
<tr>
<td>------</td>
<td>----------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td><img src="image1.png" alt="Icon" /></td>
<td>New Group</td>
<td>Adds a group node to the selected group or project.</td>
</tr>
<tr>
<td><img src="image2.png" alt="Icon" /></td>
<td>New Application</td>
<td>Adds an application node to the selected group.</td>
</tr>
<tr>
<td><img src="image3.png" alt="Icon" /></td>
<td>New Library</td>
<td>Adds a library node to the selected group.</td>
</tr>
<tr>
<td><img src="image4.png" alt="Icon" /></td>
<td>New Virtual Folder</td>
<td>Adds a virtual folder to the selected application, library or virtual folder.</td>
</tr>
<tr>
<td><img src="image5.png" alt="Icon" /></td>
<td>New Directory</td>
<td>Adds a new directory to the selected package.</td>
</tr>
<tr>
<td><img src="image6.png" alt="Icon" /></td>
<td>New Package</td>
<td>Adds a new package to the selected group.</td>
</tr>
<tr>
<td><img src="image7.png" alt="Icon" /></td>
<td>Add Files...</td>
<td>Add files to an application, group or virtual folder. Selecting this option opens a file selector dialog.</td>
</tr>
<tr>
<td><img src="image8.png" alt="Icon" /></td>
<td>Import Project...</td>
<td>Import a project under the selected project or group. Selecting this option opens the Import project dialog.</td>
</tr>
<tr>
<td><img src="image9.png" alt="Icon" /></td>
<td>Compute Dependencies</td>
<td>Removes and recomputes all dependencies between all applications and libraries in the project.</td>
</tr>
<tr>
<td><img src="image10.png" alt="Icon" /></td>
<td>Edit Build Rules</td>
<td>Opens the Build Rules Configuration dialog.</td>
</tr>
<tr>
<td><img src="image11.png" alt="Icon" /></td>
<td>Edit Package Rules</td>
<td>Opens the Edit Package Rules dialog.</td>
</tr>
</tbody>
</table>

**Diff menu**

The icon, menu name, shortcut key, and description of each option in the Diff menu.

When two files are selected for comparison using the Tools > Diff menu option, the files display under a single tab in the central work area. As long as this tab has the focus, the Diff menu is visible.

**Table 17: The Diff menu**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator Description (Shortcut)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image12.png" alt="Icon" /></td>
<td>Diff</td>
<td></td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator (Shortcut)</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td><img src="image1.png" alt="Icon" /></td>
<td>Normal View</td>
<td></td>
</tr>
<tr>
<td><img src="image2.png" alt="Icon" /></td>
<td>Diff View</td>
<td></td>
</tr>
<tr>
<td><img src="image3.png" alt="Icon" /></td>
<td>Diff View with Deleted Blocks</td>
<td></td>
</tr>
<tr>
<td><img src="image4.png" alt="Icon" /></td>
<td>Vertical Dual Diff View</td>
<td></td>
</tr>
<tr>
<td><img src="image5.png" alt="Icon" /></td>
<td>Horizontal Dual Diff View</td>
<td></td>
</tr>
<tr>
<td><img src="image6.png" alt="Icon" /></td>
<td>Code Coverage View</td>
<td></td>
</tr>
<tr>
<td><img src="image7.png" alt="Icon" /></td>
<td>Base File</td>
<td></td>
</tr>
<tr>
<td><img src="image8.png" alt="Icon" /></td>
<td>Flip Sides</td>
<td></td>
</tr>
<tr>
<td><img src="image9.png" alt="Icon" /></td>
<td>First Difference</td>
<td>Alt+Home</td>
</tr>
<tr>
<td><img src="image10.png" alt="Icon" /></td>
<td>Previous Difference</td>
<td>Alt+Up</td>
</tr>
<tr>
<td><img src="image11.png" alt="Icon" /></td>
<td>Next Difference</td>
<td>Alt+Down</td>
</tr>
<tr>
<td><img src="image12.png" alt="Icon" /></td>
<td>Last Difference</td>
<td>Alt+End</td>
</tr>
<tr>
<td><img src="image13.png" alt="Icon" /></td>
<td>Copy to Right</td>
<td>Alt+Right</td>
</tr>
<tr>
<td><img src="image14.png" alt="Icon" /></td>
<td>Copy to Left</td>
<td>Alt+Left</td>
</tr>
<tr>
<td><img src="image15.png" alt="Icon" /></td>
<td>Copy All to Right</td>
<td></td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator (Shortcut)</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td>Copy All to Left</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create Patch</td>
<td></td>
</tr>
</tbody>
</table>

**Build menu**
The icon, menu name, shortcut key, and description of each option in the Build menu.

**Table 18: The Build menu**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator (Shortcut)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compile File</td>
<td>F6</td>
<td>Compile selected file.</td>
</tr>
<tr>
<td></td>
<td>Preview</td>
<td>Ctrl+Shift+P</td>
<td>Preview the selected form.</td>
</tr>
<tr>
<td></td>
<td>Build all</td>
<td>F8</td>
<td>Build all of project.</td>
</tr>
<tr>
<td></td>
<td>Rebuild all</td>
<td>Shift+F8</td>
<td>Rebuild all of project. Clean and then build files that are not up-to-date.</td>
</tr>
<tr>
<td></td>
<td>Clean all</td>
<td></td>
<td>Clean all of selected. Erase all output files defined in the Build and Link rules.</td>
</tr>
<tr>
<td></td>
<td>Build</td>
<td>F7</td>
<td>Build default application. Compile and link files in the default application.</td>
</tr>
<tr>
<td></td>
<td>Rebuild</td>
<td>Shift+F7</td>
<td>Rebuild selected or default application.</td>
</tr>
<tr>
<td></td>
<td>Clean</td>
<td></td>
<td>Clean selected or default application.</td>
</tr>
<tr>
<td></td>
<td>Abort Last Task</td>
<td>Ctrl+Shift+F5</td>
<td>Abort last started task.</td>
</tr>
</tbody>
</table>
**Debug menu**
The icon, menu name, shortcut key, and description of each option in the **Debug** menu.

**Table 19: The Debug menu**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator (Shortcut)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Execute</td>
<td>Ctrl+F5</td>
<td>Execute selected or default application.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Execute with Profiler</td>
<td></td>
<td>Execute selected or default application with Profiler. See <strong>Profiler</strong> on page 596.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Execute with Trace</td>
<td></td>
<td>Execute the selected or default application and print all functions calls for your program, with the values of parameters passed to and returned from functions to the Output view. This option implements the --trace option for the fglrun command. For more information about the execution trace of a program, refer to <strong>Execution trace</strong> in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>F5</td>
<td></td>
<td>Debug selected or default application. See <strong>Graphical Debugger</strong> on page 589.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Attach to Process...</td>
<td></td>
<td>Debug a running application. See <strong>Start the Debugger on a running program</strong> on page 590.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Attach to Mobile Process</td>
<td></td>
<td>Debug a mobile application. See <strong>Attach the Graphical Debugger to a mobile process</strong> on page 617.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Quick Run</td>
<td></td>
<td>Compiles the current Genero .4gl file using the current configuration and launches the DVM to run the file. If the file is found in a project, it uses the current project environment. If the code opens forms, those forms will not be automatically compiled. If the code references database schemas, those schemas should be made available by setting the correct environment.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Create test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>![Icon]</td>
<td>Add/Delete Breakpoint</td>
<td>F9</td>
<td>Add or delete a breakpoint at current location.</td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator (Shortcut)</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------</td>
<td>------------------------</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Enable/Disable Breakpoint</td>
<td>Ctrl+F9</td>
<td>Enable or disable breakpoint at current location.</td>
</tr>
<tr>
<td></td>
<td>Abort Last Task</td>
<td></td>
<td>Abort last started task.</td>
</tr>
</tbody>
</table>

### Database menu

The icon, menu name, shortcut key, and description of each option in the **Database** menu.

**Table 20: The Database menu**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator (Shortcut)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Database</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extract Schema...</td>
<td>Ctrl+M</td>
<td>Extract schema from database.</td>
</tr>
<tr>
<td></td>
<td>Import SCH File...</td>
<td></td>
<td>Import schema from sch file.</td>
</tr>
<tr>
<td></td>
<td>Update Schema</td>
<td></td>
<td>Update database meta-schema file from database. See <strong>Update a meta-schema from database</strong> on page 359.</td>
</tr>
<tr>
<td></td>
<td>Generate SCH File</td>
<td></td>
<td>Generate a BDL schema file (<strong>sch</strong>) from the meta-schema file. This file is automatically created by Genero Studio when a meta-schema is compiled. See <strong>BDL schema file (sch)</strong> on page 349.</td>
</tr>
<tr>
<td></td>
<td>Generate Database Creation</td>
<td></td>
<td>Used to generate a source 4gl file that can be used to create a new database and tables according to the meta-schema file. See <strong>Generate a database script from meta-schema</strong> on page 359.</td>
</tr>
<tr>
<td></td>
<td>Generate Database Update Script</td>
<td></td>
<td>Used to generate a source 4gl file that can be used to update an existing database based on the meta-schema file. See <strong>Generate a database script from meta-schema</strong> on page 359.</td>
</tr>
<tr>
<td></td>
<td>Generate Schema Documentation</td>
<td></td>
<td>Generates an HTML page with details on the meta-schema. See <strong>Generate meta-schema documentation</strong> on page 361.</td>
</tr>
<tr>
<td></td>
<td>Diff Schema...</td>
<td></td>
<td>See <strong>Comparing two meta-schemas</strong> on page 358.</td>
</tr>
<tr>
<td></td>
<td>Add Table</td>
<td></td>
<td>See <strong>Add new tables and columns</strong> on page 349.</td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator (Shortcut)</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------</td>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Add Column</td>
<td></td>
<td>See <em>Add new tables and columns</em> on page 349.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Add Constraint or Index</td>
<td></td>
<td>See <em>Add constraints or indexes</em> on page 350.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Add Foreign Key</td>
<td></td>
<td>See <em>Add foreign keys</em> on page 351.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Select</td>
<td>Esc</td>
<td>Selection tool.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Edit</td>
<td>F2</td>
<td>Edit properties of the item selected.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Insert Column Before</td>
<td>Ctrl+Shift+Ins</td>
<td>See <em>Add new tables and columns</em> on page 349.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Insert Column After</td>
<td>Ins</td>
<td>See <em>Add new tables and columns</em> on page 349.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Revert</td>
<td></td>
<td>See <em>Revert schema changes dialog</em> on page 373.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Layout</td>
<td></td>
<td>Rearrange the items in the diagram.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Filter Items</td>
<td></td>
<td>The Filter View dialog allows you to hide and show items on a diagram.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Locate in Diagram</td>
<td></td>
<td>This action brings focus in the diagram to the selected item. If the selected object is not visible in the current view, the Meta-schema Manager will try to find another view where the object is visible. If no view is found, you are prompted to make the object visible in the current view or to create a new view.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Advanced Properties</td>
<td></td>
<td>Specify the extraction and/or generation options. See <em>Extract meta-schema information from database</em> on page 277 and <em>Generate a database script from meta-schema</em> on page 359.</td>
</tr>
</tbody>
</table>

**SCM menu**
The icon, menu name, shortcut key, and description of each option in the **SCM** menu.

**Table 21: The SCM menu**
<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator (Shortcut)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Icon" /></td>
<td>Checkout...</td>
<td></td>
<td>Checkout files from a repository. See Checkout files on page 625.</td>
</tr>
<tr>
<td><img src="image2" alt="Icon" /></td>
<td>Show Log</td>
<td></td>
<td>Display the SVN Log view. See SVN Log view on page 637.</td>
</tr>
<tr>
<td><img src="image3" alt="Icon" /></td>
<td>Review Changes</td>
<td></td>
<td>Display the SVN Status view. See Commit / Review changes on page 626.</td>
</tr>
<tr>
<td><img src="image4" alt="Icon" /></td>
<td>Show locks</td>
<td></td>
<td>Display SVN Locks view. See SVN Locks view on page 638.</td>
</tr>
<tr>
<td><img src="image5" alt="Icon" /></td>
<td>Merge/Revert...</td>
<td></td>
<td>Display the SVN Merge/Revert dialog. See Merge and revert on page 631.</td>
</tr>
<tr>
<td><img src="image6" alt="Icon" /></td>
<td>Copy...</td>
<td></td>
<td>Display the SVN Copy dialog. See Copy working files and directories on page 631.</td>
</tr>
<tr>
<td><img src="image7" alt="Icon" /></td>
<td>Switch...</td>
<td></td>
<td>Display the SVN Switch dialog. See Move a working copy (Switch) on page 632.</td>
</tr>
<tr>
<td><img src="image8" alt="Icon" /></td>
<td>Apply patch...</td>
<td></td>
<td>The Apply patch command applies a patch file to files in the repository. See Apply patch on page 632.</td>
</tr>
<tr>
<td><img src="image9" alt="Icon" /></td>
<td>Browse Repository</td>
<td></td>
<td>Display the SVN Repository Browser. See Browse repository on page 633.</td>
</tr>
<tr>
<td><img src="image10" alt="Icon" /></td>
<td>Update</td>
<td></td>
<td>Display the SVN Update dialog. See Update / Update All on page 630.</td>
</tr>
<tr>
<td><img src="image11" alt="Icon" /></td>
<td>Lock</td>
<td></td>
<td>Locking a file provides exclusive rights to a user for changing that file in the repository. See Locking on page 627.</td>
</tr>
<tr>
<td><img src="image12" alt="Icon" /></td>
<td>Cleanup</td>
<td></td>
<td>The Cleanup command cleans up the working copy, removing stale locks.</td>
</tr>
<tr>
<td><img src="image13" alt="Icon" /></td>
<td>Blame</td>
<td></td>
<td>The Blame command shows author and revision information inline for specified files or URLs.</td>
</tr>
<tr>
<td><img src="image14" alt="Icon" /></td>
<td>Blame...</td>
<td></td>
<td>Display the SVN Blame dialog.</td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator (Shortcut)</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------</td>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>![Properties]</td>
<td>SVN Properties</td>
<td></td>
<td>Display the SVN Properties dialog. See SVN Properties dialog on page 644.</td>
</tr>
</tbody>
</table>

**Tools menu**
The icon, menu name, shortcut key, and description of each option in the Tools menu.

**Table 22: The Tools menu**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator (Shortcut)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Dependency Diagram]</td>
<td>Dependency Diagram</td>
<td></td>
<td>Open Dependency Diagram. See Dependency Diagrams on page 484.</td>
</tr>
<tr>
<td>![Genero Tools]</td>
<td>Genero Tools</td>
<td></td>
<td>Access to Genero tools such as the BDL Licenser and the Genero Workplace Window.</td>
</tr>
<tr>
<td>![Android™ Tools]</td>
<td>Auto-configure Android™ SDK</td>
<td></td>
<td>Run the scripts to auto-configure the Android™ SDK for use in Genero mobile development.</td>
</tr>
<tr>
<td>![Create Android™ Virtual Device (x86)]</td>
<td>Create Android™ Virtual Device (x86)</td>
<td></td>
<td>Create the default Android™ emulator.</td>
</tr>
<tr>
<td>![Launch Android™ Emulator (x86)]</td>
<td>Launch Android™ Emulator (x86)</td>
<td></td>
<td>Launch the default Android™ emulator.</td>
</tr>
<tr>
<td>![Deploy Genero Mobile for Android™]</td>
<td>Deploy Genero Mobile for Android™</td>
<td></td>
<td>Deploy the GMA client to the connected device or emulator.</td>
</tr>
<tr>
<td>![Open Android™ Debug Monitor]</td>
<td>Open Android™ Debug Monitor</td>
<td></td>
<td>Launch Android™ Debug Monitor tool which is part of the Android™ SDK.</td>
</tr>
<tr>
<td>![List Devices]</td>
<td>List Devices</td>
<td></td>
<td>Lists each attached device and its unique id.</td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator (Shortcut)</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Show Android™ Device Logs</td>
<td></td>
<td>Displays the Android™ Device Logs to the Output view. See View device logs (Android) on page 622.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Display Standard output and error / Stop display Standard output and error</td>
<td></td>
<td>Display or stop display of the Standard output and errors to the Output view.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Show AUI Tree</td>
<td></td>
<td>Display AUI tree in browser. See View the AUI tree (Android) on page 619.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>iOS Tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>![Icon]</td>
<td>Launch iOS Simulator</td>
<td></td>
<td>Launch the default iOS Simulator.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Deploy Genero Mobile for iOS</td>
<td></td>
<td>Deploy the GMI client to the connected device or simulator.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>iOS App Store deployment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>![Icon]</td>
<td>Open iTunesConnect</td>
<td></td>
<td>Open the <a href="https://itunesconnect.apple.com">https://itunesconnect.apple.com</a> webpage.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Launch Application Loader</td>
<td></td>
<td>Opens Xcode's &quot;Application Loader&quot; application.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>List Devices</td>
<td></td>
<td>Lists each attached device and its unique id.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Show AUI Tree</td>
<td></td>
<td>Display AUI tree in browser. See View the AUI tree (iOS) on page 620.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Current Config</td>
<td></td>
<td></td>
</tr>
<tr>
<td>![Icon]</td>
<td>Launch GDC</td>
<td>Ctrl+Shift+G</td>
<td>Launch the Genero Desktop Client (GDC).</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Launch GDC...</td>
<td></td>
<td>Prompt for GDC options before launching GDC.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Open Application Server Monitor</td>
<td></td>
<td>Open Application Server Monitor.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Genero BDL Licenser</td>
<td></td>
<td>Open Genero BDL Licenser program.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Genero Workplace Window</td>
<td></td>
<td>Open the Genero Workplace Window. See Genero Workplace Window on page 124.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Global Setup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>![Icon]</td>
<td>Edit File Associations</td>
<td></td>
<td>See File associations configuration on page 129.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Edit Build Rules</td>
<td></td>
<td>Edit global build rules. See What are build rules on page 413.</td>
</tr>
<tr>
<td>Icon</td>
<td>Menu Item</td>
<td>Accelerator (Shortcut)</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------</td>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Edit Package Rules</td>
<td></td>
<td>Edit global package rules. See Package and deploy rules (mobile) on page 1192.</td>
</tr>
<tr>
<td></td>
<td>Specific Setup</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Edit File Associations</td>
<td></td>
<td>See File associations configuration on page 129.</td>
</tr>
<tr>
<td></td>
<td>Edit Build Rules</td>
<td></td>
<td>See What are build rules on page 413.</td>
</tr>
<tr>
<td></td>
<td>Reload</td>
<td>Ctrl+Shift+R</td>
<td>Reload Application Generator settings.</td>
</tr>
<tr>
<td></td>
<td>Clean orphan properties</td>
<td></td>
<td>Opening a file generated with a template version different than the one set in Application Generator preferences may produce a warning indicating that some properties are not found in the current template definition.</td>
</tr>
<tr>
<td></td>
<td>Server Connections</td>
<td></td>
<td>Connect to a remote server. See Setting up a remote environment on page 179</td>
</tr>
<tr>
<td></td>
<td>Genero Configurations</td>
<td>Ctrl+Shift+C</td>
<td>Open Configuration Management to modify configurations. See Genero Configuration Management dialog on page 197.</td>
</tr>
<tr>
<td></td>
<td>Preferences</td>
<td>Alt+Shift+F12</td>
<td>Open Preferences. See Setting Preferences on page 128.</td>
</tr>
<tr>
<td></td>
<td>Translation</td>
<td></td>
<td>Change the language of the labels in Genero Studio.</td>
</tr>
</tbody>
</table>

**Windows menu**
The icon, menu name, shortcut key, and description of each option in the **Windows** menu.

**Table 23: The Windows menu**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator (Shortcut)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Close all other Documents</td>
<td></td>
<td>Close all documents except the current document.</td>
</tr>
<tr>
<td></td>
<td>Close all Documents</td>
<td>Ctrl+Shift+F4</td>
<td>Close all open documents.</td>
</tr>
</tbody>
</table>
### Table 24: The Help menu

<table>
<thead>
<tr>
<th>Icon</th>
<th>Menu Item</th>
<th>Accelerator (Shortcut)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Help icon]</td>
<td>Help</td>
<td>F1</td>
<td>Open Help.</td>
</tr>
</tbody>
</table>

### Views

Views are the panels in the Genero Studio framework that display information about the current document or project. To view a list of views, select **Window > Views**.

- **Show, dock, or move a view** on page 116
- **Views Listing** on page 117

### Show, dock, or move a view

Views (panels) can be hidden and shown, docked or undocked, and moved within the framework.

### Show or hide a view

Show or hide views by right-clicking in the Genero Studio window title, or use **Window > Views**.

**Note:** The Data View and Tool Box views are only listed when you have a report design document open in the Document view.

For a list of hot keys to show and hide the views, look at **Window > Views**. Where available, the hot key is listed next to the view name.

### Select visible views

To open the **Manage Views** dialog, select **Window > Views > Manage Views**. A dialog opens listing all views. Use the checkbox to set each view as visible or hidden.

- If the view has a check in its checkbox, then the view is shown in Genero Studio.
- If the view does not have a check in its checkbox, then the view is hidden.

To make all views visible, select the **Select all** checkbox located at the bottom of the dialog.
Dock or undock a view

Undock a view by double-clicking on its title bar. Re-dock a view to its last position by double-clicking on its title bar.

Move views

Move a view to by selecting its title bar and dragging it to float or to a new position in the framework. As you move the view, shaded areas appear showing you valid locations to place the view.

Views Listing

This is a list of views available in Genero Studio.

Some views are available at all times, while others are only available when the appropriate module is in use.

- Backtrace - See Backtrace view on page 603.
- Bookmarks - See Bookmarks view on page 117.
- Breakpoints - See Breakpoints view on page 602.
- Command - See Command view on page 599.
- Data View (Report Designer) - See Adding report data (Data view) on page 796. This view is only available when using Genero Report Designer.
- Data (Debugger) - See Data view on page 600.
- DB Explorer - See DB Explorer on page 392.
- DB Schemas - See DB Schemas tab on page 374.
- Document - The Document view is the default view, also known as the central work area. You cannot hide the Document view.
- Document Errors - See Document Errors view on page 118.
- Files - See File Browser on page 588.
- Function Callers - See Code Editor basics on page 445
- Function Search - see Function search on page 451.
- Output - See Output view on page 118.
- Projects - See Projects view on page 119.
- Properties - See Properties view on page 119.
- SVN Locks - See SVN Locks view on page 638.
- SVN Log - See SVN Log view on page 637.
- SVN Repository - See Browse repository on page 633.
- SVN Status - See SVN Status view on page 639.
- Search Results - See Search Results view on page 471.
- Search/Replace - See Search/Replace view on page 467.
- Structure - See Structure view on page 120.
- Tasks - See Tasks view on page 120.
- Tool Box - See The Tool Box view on page 781. This view is only available when using Genero Report Designer.
- Watchpoints - See Watchpoints view on page 601

Bookmarks view

The Bookmarks view lists the bookmarks for the current project.

The Bookmarks view includes these options:

- Navigate previous / next bookmark
  Activates the previous / next bookmark in the view.
- Delete bookmark / Delete all bookmarks
  Removes current or all defined bookmarks.

Related concepts

Bookmarks on page 447
Bookmarks provide quick access to areas of a document.

**Document view**
The Document view is the area designated for working on the document of a Genero Studio module, a form in Form Designer, for example. It is commonly called the central work area.

Each document has its own tabbed page. The Document view may contain as many tabbed documents as needed.

- Click a tab to **bring a document to the front**.
- Click the Select a desired document icon
  ![Select a desired document icon]
  at the top right corner to **select a desired document** from the drop-down list.
- Click the close icon
  ![Close current document icon]
  to close the current document.
- Right-click a tab to display this menu:
  - Close the **current document**
  - Close all **other documents**
  - Close all **documents**

The Window menu allows you to display various views, providing additional information.

**Document Errors view**
The Document Errors view displays errors related to a document.

- Select the error number and press the F1 key to display additional information about the error.
- Use Tools >> Preferences, Genero Studio Preferences, Messages to hide a specific information or warning message.
- View **BUG** or **TODO** notations found in your code. Enter the notation into your code files with ```--KEYWORD <message>``` where **KEYWORD** is **BUG** or **TODO**. After compiling, put focus in the Project Manager view to see these messages in the Document Errors tab.

**Output view**
The Output view displays messages related to the output and errors specific to the process being performed.

Click **Show as List** to toggle between text display and list display.

With a list display, you can filter the items:

- The **Channel** drop-down list filters by process.
- Selecting icons filters by severity. Each icon acts as a toggle switch. You can have more than one message type selected.

**Table 25: Message icons**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Error Icon]</td>
<td>Select this icon to show/hide error messages.</td>
</tr>
<tr>
<td>![Warning Icon]</td>
<td>Select this icon to show/hide warning messages.</td>
</tr>
<tr>
<td>![Information Icon]</td>
<td>Select this icon to show/hide information messages.</td>
</tr>
</tbody>
</table>
Properties view
Select an item in the structure view or central work area to display its properties in the Properties view; the property values can be viewed and/or changed.

Related concepts
Changing a property value (The Properties view) on page 795
Select a report element in the report page (work area) or Structure View to display and edit the property values in the Properties View.

Meta-schema properties on page 362
Properties can be set for each element in a meta-schema.

Form item properties on page 492
Properties can be set on form items such as containers and widgets to provide information on how the runtime system should display or handle the item.

Projects view
The Projects view displays a project and its components.

Right-click on a node to display a context menu of available actions.
Select a node to set its properties in the Project Manager node properties on page 434.
You can find files more easily by using the Filter items text box at the bottom of the Projects view.

Table 26: Projects View Nodes

<table>
<thead>
<tr>
<th>Node</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>projectName.4pw</td>
<td>The active project (4pw) is the root node.</td>
</tr>
<tr>
<td>Group</td>
<td>Contains the Application and Library nodes that make up the project. Properties defined for the Group node can be inherited by the Application and Library nodes.</td>
</tr>
<tr>
<td>Node</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Application</td>
<td>Generates an executable program (42x). It can contain both files and virtual folders. Only one of the files may have a MAIN statement: One application node = one executable. The name of the application node is used as the name of the 42x file, so it must be unique and can only contain characters allowed by the file system. The default application is shown in boldface. Use the Toolbar icon to set a different application as the default. The options on the Build menu execute for the default application.</td>
</tr>
<tr>
<td>Library</td>
<td>Used to group binary files into a single library and generate a library file (42x). It can contain both files and virtual folders. The name of the library node is used as the name of the 42x file, so it must be unique and can only contain characters allowed by the file system. Libraries should be used when creating a set of features having a common goal, like the logic of an application, a library of mathematical functions, etc. A library can also be used to group other project files together (images, styles or other resources). If a library node contains no 4gl file, no 42x is built. A library from a different project can be added to a project using the right-click menu option Add external dependencies. <strong>Important:</strong> A library must be linked to any application in which it will be used by right-clicking the application node and selecting Advanced Properties, dependencies. The checkbox for any required library must be checked.</td>
</tr>
<tr>
<td>Folder</td>
<td>Folders are virtual folders only, providing a way to group the source files within an application or library node. Folders can contain both files and other folders.</td>
</tr>
<tr>
<td>File</td>
<td>A link to the file in the file system that has the same name as the node. Renaming a File node also renames the file that is stored on the disk. Project Manager will accept any type of file. Opening a file opens the corresponding Genero Studio module, such as Code Editor. If there is no corresponding Genero Studio module, it asks the operating system to open it.</td>
</tr>
</tbody>
</table>

**Structure view**
The Structure View displays a tree showing the structure of the current document, a form or source code file, for example.

**Related concepts**
- Organizing the report structure (the Report Structure view) on page 790
- You use containers to make sure that the report elements print correctly.
- Viewing a meta-schema on page 356
  Once a database meta-schema file has been created, it can be viewed and enriched with information that is not present in the database. Opening a meta-schema file displays it as a diagram for viewing and editing.
- Code Structure view on page 466
  If the file being edited is a Genero source code file (4gl), its structure is displayed in a tree.

**Tasks view**
The Tasks view displays a list of the current tasks started by Genero Studio. From this view, you can stop a task.

In addition to listing running applications, the Tasks view displays a task for the Genero Desktop Client or the Genero Application Server when they are started by Genero Studio.

To stop a running task:
1. Select the task row.
2. Click the Abort toolbar icon.

All running tasks are stopped when you exit Genero Studio.
**Documents**

A document refers to a file that you view in the central work area.

Examples of documents include text-based source files, graphical form design files, application flow diagrams, report design documents, and more. When you open a document, it opens in the central work area; it is not in a view that you can re-locate or undock.

**View two documents simultaneously**

By default, when you open two or more documents, one document has the focus and sits on top of the others. You switch between documents by clicking the appropriate tab.

If you wish to view two documents side-by-side, right-click on the tab and select **Move document to other view**. You can repeat this process to relocate additional documents to the new location.

To move the document back to its original location, right-click on the tab and (once again) select **Move document to other view**. The document will move back to its original location. When you have done this to all documents that you may have moved to the second location, you again are presented with the default, where a single document has the focus and sits on top.

**Dialogs**

Dialogs display as the result of a selected action. In most cases, you must complete the dialog to continue.

- **Filter View dialog** on page 121
- **Print preview dialog** on page 122

**Filter View dialog**

The Filter View dialog allows you to hide and show items on a diagram.

Right-click the diagram or an item on the diagram and select the **Filter Items** option. Check boxes on the Filter View dialog allow you to specify which items to display. Type a value in the search field to reduce the number of check boxes shown.

To access the Filter view, right-click on a diagram in:

- Meta-schema Manager
- Dependency Diagram
- Business Application Modeling
Figure 74: Filter View

Print preview dialog
The Print preview dialog allows you to preview a document or diagram, before sending it to the printer. Various toolbar icons allow you to change how the document or diagram prints.

The Print preview dialog appears when you select File > Print Preview....

Figure 75: Print preview dialog
Actions available from the Print preview dialog are presented as icons in the toolbar.

**Table 27: Print preview toolbar icons**

<table>
<thead>
<tr>
<th>Icon Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print</td>
<td>Opens the print dialog.</td>
</tr>
<tr>
<td>Portrait printing</td>
<td>Formats layout to be a portrait page.</td>
</tr>
<tr>
<td>Landscape printing</td>
<td>Formats layout to be a landscape page.</td>
</tr>
<tr>
<td>Page setup</td>
<td>Opens the page setup dialog.</td>
</tr>
<tr>
<td>Fit to width</td>
<td>Fits the content to the width of the page.</td>
</tr>
<tr>
<td>Fit to page</td>
<td>Fits the content to the size of the page.</td>
</tr>
<tr>
<td>Percentage combobox</td>
<td>Sets the display size in proportion of the final print size.</td>
</tr>
<tr>
<td>Zoom in</td>
<td>Enlarges the display percentage.</td>
</tr>
<tr>
<td>Zoom out</td>
<td>Reduces the display percentage.</td>
</tr>
<tr>
<td>Single page</td>
<td>View a single page at a time.</td>
</tr>
<tr>
<td>Facing pages</td>
<td>View pages in book style, with even and odd pages facing each other.</td>
</tr>
<tr>
<td>Overview of pages</td>
<td>View all pages in two columns.</td>
</tr>
<tr>
<td>Page navigator</td>
<td>Navigate to the front of the print document, to the previous page, to a specific page, to the next page, or to the last page of the print document.</td>
</tr>
<tr>
<td>Help</td>
<td>Access the help documentation.</td>
</tr>
</tbody>
</table>
Genero Workplace Window

The Genero Workplace Window provides a command-line interface, using the environment of your current configuration or project. This allows you, for example, to run the Genero Studio commands (such as gsmake and gsmake) within the correct environment.

If you are using a remote configuration, the Genero Workplace Window opens on the remote host via ssh with the environment set. If you open the Genero Workplace Window using the context menu in Project Manager, it uses the environment of the current node. If you open it from the Tools > Genero tools > Genero Workplace Window menu, it uses the current Genero configuration.

Related concepts
Command line options on page 125
Genero Studio can be launched at the command line with the command generostudio at a console, terminal, or using the Genero Workplace Window.

Updating Genero Studio and its documentation

Genero Studio software and documentation can be updated via the internet.

Genero Studio needs permission to provide software update notifications. Complete the Software Update dialog to activate notifications. This dialog first appears as part of the installation of Genero Studio. When activated, you receive a notice when a software or documentation update is available. If you do not activate software update notifications during the installation, you can always activate at a later time.

Tip: Update notices are sent weekly. To check for updates between notices, you can use the Help > Check for Updates menu option. If any updates are available, the proper dialog displays.

Software updates

If updates are available, you are notified and redirected to the Four Js web site. The update is not automatic, you need to download the update from the Web site and manage the installation yourself.

Documentation updates

If you have notification activated and a documentation update is available:

A dialog displays asking if you want to download the documentation update. Answer Yes and the documentation bundle is downloaded. A message is written to the Output view telling you that the download finished, and a second dialog displays.

The second dialog asks you whether you want to install the update you have downloaded. Answer Yes to install the updated documentation. This overwrites the previous documentation with the new documentation.

Tip: From the documentation launched from Genero Studio (either by clicking the Help icon or by pressing the F1 key), select Help > About Genero Studio Help to see the version and build number of the current documentation set.

The Software Update notification dialog

When you first start Genero Studio after installation, the Software Update notification dialog displays. This dialog allows you to activate your update notifications.

To activate the update notifications:

1. Check the box signifying that you agree to the terms and wish to activate software updates.
2. Enter the email address you use to log into the Four Js web portal.
3. Click OK.
Check your update status

To see whether updates are activated, select Tools > Preferences.

Note: On a Mac, the Preferences dialog is found under the product name menu option.

A message under the Software Update section tells you if updates are activated, and the name of the associated account.

Activate updates

Select Help > Check for Updates. Complete the Software Update dialog

De-activate updates

Select Tools > Preferences to access the Preferences dialog.

Note: On a Mac, the Preferences dialog is found under the product name menu option.

On the General page, under the Software Update section, click Reset.

Learning to use Genero Studio

Reference topics for the samples directory, setting preferences, accessing help and more.

- Command line options on page 125
- Samples directory on page 126
- Integrating existing applications on page 127
- Setting Preferences on page 128
- Access Help on page 138

Command line options

Genero Studio can be launched at the command line with the command generostudio at a console, terminal, or using the Genero Workplace Window.

Syntax

generostudio [options]

Table 28: Genero Studio command line options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td>Display help instead of the standard behavior.</td>
</tr>
<tr>
<td>-V</td>
<td>Display program name and version information.</td>
</tr>
<tr>
<td>-height HEIGHT</td>
<td>Set window height in pixels.</td>
</tr>
<tr>
<td>-width WIDTH</td>
<td>Set window width in pixels</td>
</tr>
<tr>
<td>-translate LOCALE</td>
<td>Set translation file where LOCALE is zh_TW, the locale for Taiwanese (using traditional Chinese characters) or zh_CN, the locale for Chinese (using simplified Chinese characters).</td>
</tr>
<tr>
<td>-log [all, req, &lt;id&gt;]</td>
<td>Set the log level. Use all to turn on all logs, req to turn on server request logs, and &lt;id&gt; to turn on logs for a given module.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-diff FILE1 FILE2</td>
<td>Open the given files in diff mode. See Using the Diff tool on page 451.</td>
</tr>
</tbody>
</table>

**Related concepts**

**Genero Workplace Window** on page 124

The Genero Workplace Window provides a command-line interface, using the environment of your current configuration or project. This allows you, for example, to run the Genero Studio commands (such as gsreport and gsmake) within the correct environment.

**Samples directory**

The samples directory contains demo files, programs, and databases.

The default location for the sample files is My Documents/My Genero Files/samples.

Genero Mobile samples (installed with the standalone Genero Mobile product) are located in My Documents/My Genero Mobile Files/samples.

- **HelloWorld**
  Contains the HelloWorld.4pw project, which has the source code for the very simple Hello World application.

- **OfficeStore**
  Contains the OfficeStore.4pw project which is an application generated using Business Application Modeling (BAM). It uses the officestore sample database.

- **WebComponentChart**
  Contains the WebComponentChart.4pw project, which has the source code for including a charting Web Component within a Genero application.

- **HTML5ClientTutorial**
  Contains the HTML5ClientTutorial project files, which demonstrate customization of a web application.

- **Reports**
  Contains the Reports.4pw project, which has a set of example programs that create reports using Genero Report Writer.

- **DSConfig**
  Contains the DSConfig.4pw project, which has the program source code to complete an FGLPROFILE configuration file with the necessary information for a database selected by the user. An FGLPROFILE file already exists for the sample databases, but you must create one if you choose to use your own database.

- **databases**
  Contains directories for the sample SQLite databases custdemo and officestore. The directories also contain schema files and a program file with the necessary SQL commands to recreate the databases for your own database software, if desired. These files are grouped in project files, OfficeStoreSetup.4pw and CustDemoSetup.4pw.

- **BDL tutorial**
  Contains the BDLTutorial.4pw project, which has a set of tutorial programs that illustrate the use of the Genero Business Development Language (BDL), to be used in conjunction with the BDL Tutorial.

The necessary data for the sample programs is stored in the provided SQLite databases.
Open the project for one of the samples to examine and execute the sample programs. From the Genero Studio Welcome Page, select the Tutorials and Samples tab. Links to open the sample projects are displayed under Samples and Demos.

**Integrating existing applications**

Considerations for integrating existing applications into Genero Studio.

<table>
<thead>
<tr>
<th>Consideration</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine whether you will use a local or remote configuration of Genero Studio.</td>
<td>See Software configuration scenarios on page 139.</td>
</tr>
<tr>
<td>Incorporate a schema.</td>
<td>See Create a meta-schema on page 277.</td>
</tr>
<tr>
<td>Import your files.</td>
<td>See Import existing files as a new project on page 409.</td>
</tr>
<tr>
<td>Reorganize your project nodes and check dependencies</td>
<td>See Default organization of imported files.</td>
</tr>
<tr>
<td>Start correcting errors - syntax, link errors, language errors.</td>
<td>See Code Editor basics on page 445.</td>
</tr>
<tr>
<td>Recompile and link. Adapt makefiles.</td>
<td>Adapt makefiles or create build rule that calls a makefile. See Building and linking programs on page 412.</td>
</tr>
<tr>
<td>Run your application in traditional mode.</td>
<td>You can use the traditional GUI mode to ease migration from TUI based applications to GUI mode.</td>
</tr>
<tr>
<td></td>
<td>With the traditional mode, application windows bound to forms using a SCREEN section will be displayed as simple boxes in a main front end window. Other windows bound to forms defined with the LAYOUT section will be displayed a new GUI windows.</td>
</tr>
<tr>
<td></td>
<td>The traditional GUI mode can be enabled with an FGLPROFILE entry:</td>
</tr>
<tr>
<td></td>
<td>gui.uiMode = &quot;traditional&quot;</td>
</tr>
<tr>
<td>Consider form migration from .per to .4fd and begin using Form Designer.</td>
<td>See Importing .per files on page 493, Creating the user interface on page 488, Form Designer usage on page 523.</td>
</tr>
<tr>
<td>Modernize forms.</td>
<td>See Creating the user interface on page 488.</td>
</tr>
<tr>
<td>Analyze code.</td>
<td>See Code Analyzer on page 482.</td>
</tr>
<tr>
<td>Debug.</td>
<td>See Graphical Debugger on page 589.</td>
</tr>
<tr>
<td>Switch client configurations from desktop to web.</td>
<td>See Change the active configuration on page 141.</td>
</tr>
<tr>
<td>Improve the meta-schema.</td>
<td>See Adding to a meta-schema on page 349</td>
</tr>
<tr>
<td>Consider modernizing code.</td>
<td>See <a href="http://www.4js.com">www.4js.com</a></td>
</tr>
<tr>
<td>Consider report strategy.</td>
<td>See Report Writer on page 647.</td>
</tr>
<tr>
<td>Customize studio environment.</td>
<td>See Integrate your tools into Genero Studio on page 138, General Preferences on page 128.</td>
</tr>
</tbody>
</table>
Setting Preferences

Customize Genero Studio to meet your needs.

- **General Preferences** on page 128
- **User interface preferences** on page 135
- **Compiler and Runtime preferences** on page 137
- **Integrate your tools into Genero Studio** on page 138

Select **Tools > Preferences** to modify the behavior of Genero Studio modules.

Save, test, or cancel changes you have made.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load from default</td>
<td>Reloads the initial default configuration.</td>
</tr>
<tr>
<td>OK</td>
<td>Save and apply all modifications, then exit Preferences window.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Undo all modifications and exit the Preferences window. The last saved values are restored.</td>
</tr>
<tr>
<td>Apply</td>
<td>Confirms your updates, allowing you to test the new configuration. If you want this configuration to become permanent, save it by pressing the OK button.</td>
</tr>
</tbody>
</table>

**Related concepts**

- **Meta-Schema Manager preferences** on page 375
  Set default preferences for Meta-Schema Manager and DB Explorer.

- **Form Designer preferences** on page 528
  The **Form Designer Preferences** page sets the preferences for form elements in Form Designer.

- **Code Editor preferences** on page 459
  Set preferences for the Code Editor. For example, you can set the default diff mode or the color and font preferences.

- **Customize Diff tool preferences** on page 465
  Set preferences for the Diff tool. For example, you can set the tool to ignore case or white space, or specify the color used.

- **Report Writer preferences** on page 982

**Related tasks**

- **Specify a Subversion client** on page 645
  To specify the location of your SVN client, select **Tools > Preferences > Source Code Management > Subversion** from the main menu.

**General Preferences**

Set general preferences, such as text file encoding and proxy settings.

Select **Tools > Preferences > General** to access these preferences.

**Table 30: General preferences**

<table>
<thead>
<tr>
<th>Preference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text File Encoding</td>
<td>See <strong>Language support (text encoding)</strong> on page 189.</td>
</tr>
<tr>
<td>Proxy Setting</td>
<td>If you need to access a web service using a proxy, enter:</td>
</tr>
<tr>
<td></td>
<td>• Host - the hostname or host IP address</td>
</tr>
<tr>
<td></td>
<td>• Port - The port number on which the service is listening</td>
</tr>
</tbody>
</table>
### Preference | Description
--- | ---
Browser Setting | You can specify the browser to be used with the Genero Browser Client:
  - Use default web browser
  - Use specific web browser - enter the filename of the executable, including path, of the browser that you wish to use, or select the file in your file system using the Browse icon.
Slow Networks | Select to optimize for slow networks.
Users and Passwords | Press the Clear button to clear the passwords stored in Genero Studio.

**File associations configuration**

Select **Tools > Global setup > Edit File Associations** or **Tools > Specific setup > Edit File Associations** to edit the standard or specific file associations, such as for a template.

The template file associations are available if the `GSTSETUPDIR` environment variable is set and refers to an BAM template directory. The dialog lists the file associations set in the file-types.xml file found in the template directory.

You can associate file types handled by Genero Studio modules with Genero Studio predefined actions, or with **User Actions** that you have defined.
Figure 76: File Associations

Use the integrated Toolbar to add, modify or delete a new file type.

**File Types**

- **Search**: Search for the file type you wish to view or edit.
- **Mime**: Predefined file type/identifier
- **Description**: Optional description.
- **Editor Content Type**: Type of editor used with this file type.
- **Extensions**: File extensions matching the file type.
Filenames
Optional file names matching the file type.

Associated Actions
When you select Add from the icons at the top of the Actions list, the available actions are displayed in a list for your selection. User Actions must be defined using Preferences, User Actions before they will appear in the list.

Action
Predefined action names.
Default
Indicates the action is the default; use the icon to change the default.
Open
Indicates the action is used to open the file; use the icon to change.
Command
Predefined command to be executed; may be [Internal].

Appearance
Icon
Search for the icon on your file system. The path is specified for icons that are not in the default Genero Studio icons directory. See Image directory structure on page 1121.
Preview
Sample is displayed.

Workspaces Configuration
The Workspaces Configuration page sets preferences for the workspace and views arrangement of Genero Studio.
To access the preferences page, select Tools > Preferences, then select Workspaces from under the General node.

Documents on Startup
Select the Open last session documents on startup checkbox to restore the last documents you had open when you start the next Genero Studio session.

Workspaces
A workspace refers to the organization and visibility of views when opened in the local Genero Studio.
The Default category provides three pre-defined workspaces, each serving a different purpose:

Normal
Displays the most common views needed for day-to-day application development: Projects, Structure, Properties, Output and Tasks.
Debug
Removes some of the common views to focus on the Debugger; automatically switches the workspace when starting and stopping a debug session. Keeps track of the current workspace and the one used while debugging, and switches between them when a debug session is started or stopped.
Document
Displays the current document and the Project Manager to focus on document editing.

The User category is for any user-created workspaces. A user-created workspace is first created by duplicating an existing workspace. Only workspaces in the User category can be renamed or deleted.
The Restore default workspaces button restores the layout of the workspaces in the Default category.
**User actions**
User actions define commands that can be invoked from a menu or toolbar icon within Genero Studio.

**Important:** If you are using a remote configuration, the user action is executed on the remote server unless you specify otherwise.

To view user actions, select **Tools > Preferences > General > User Actions**.

![User Actions Configuration page](image)

**Figure 77: User Actions Configuration page**

To add or edit user actions, use the toolbar icons on the **User Actions Configuration** page.
Figure 78: Add a User Action dialog

Table 31: Adding user action detail

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the user action.</td>
</tr>
<tr>
<td>Label</td>
<td>Label for the user action as displayed in Genero Studio action listings, menus, or toolbars.</td>
</tr>
<tr>
<td>Description</td>
<td>Optional.</td>
</tr>
<tr>
<td>Icon</td>
<td>Optional.</td>
</tr>
<tr>
<td>Target nodes</td>
<td>Project nodes that display the user action in their contextual menus. To attach the user action to another project node, click the ellipse at the end of this field.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If the user action is not attached to any nodes, the message &quot;Check File Associations&quot; displays.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Command line          | Command to be executed. If the command includes spaces, it must be in quotes. Arguments can be included on the command line as needed. Use the **Insert Variable** button to select from predefined node variables. Create a variable to prompt the user to input an argument. • This example executes the program **stest** using the selected file: "C:\myprograms\stest $(FileName)" • This example executes the program **stest**, prompting the user to input the file name for the user defined variable $(p1): "C:\myprograms\stest $(p1)" • This example opens the associated file (a .per or .4fd file) as a compiled .42f file: 
  "$(open) $(FileDir)/$(FileBaseName).42f)" |
| Display output        | If selected, the output from the executed commands are displayed in the Output view.                                                        |
| Verbose               | If selected, the user action commands are displayed in the Output view.                                                                      |
| Always execute locally| If selected, the user action is executed locally even if the current configuration is remote.                                                |

**Associating the User Action**

You can associate the user action with:

- Target nodes (see above)
- Files of a specific mime type
- Toolbar icons
- Menus

**Messages configuration**

**Message dialogs**

A list of messages is displayed as a tree, grouped by Genero Studio module. Expand a module listing to display its messages.

You can prevent a message from displaying while you are using Genero Studio by unchecking the **Show** checkbox for the message.

Use the **Search** box to locate a specific message.

**Document Error Filters**

This list specifies the message id of information or warning messages that you wish to be hidden in the Document Errors tab of the Output View. Use the icons to add and remove message ids from the list.

**History configuration**

In the work area of many of the Genero Studio modules, the names of previously used items are stored and displayed as history. Genero Studio allows you to manage these History lists.

**History list**

The name of each list that you can manage is displayed. These are the only lists that you can alter.
History Details

Table 32: History Details

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>History Size</td>
<td>Specify the maximum number of items displayed in a list.</td>
</tr>
<tr>
<td>Allow Duplicates</td>
<td>If this box is checked, duplicate names can be added to the list of items. If the list contains duplicate items, unchecking the box will cause a dialog box to be displayed, asking you to confirm the removal of the duplicate items from the list.</td>
</tr>
</tbody>
</table>

History items

Icons allow you to:

- **Add** - add a new item
- **Delete** - remove the selected item
- **Edit** - change the name of the selected item
- **Up/Down arrow** - Rearranges the items in the list by moving a selected item up and down.
- **Default** - specifies the default item for the selected list.

User interface preferences

Set preferences for toolbars, menu bars, and accelerators.

- Toolbars configuration
- Menu configuration
- Accelerators configuration

Toolbar configuration

Manage toolbars and set visibility preferences using the Toolbars Configuration page.

To access the page, select **Tools > Preferences > User Interface > Toolbars.** Each toolbar displays as a folder.

Modifying a toolbar

Expand a toolbar folder to display its actions.

Select a toolbar or action, and use the icons or right-click to display the menu options:

- Add a new toolbar
- Add an action to a toolbar
- Remove or rename a toolbar
- Remove or rename an action
- Add or remove a separator

Item order within a toolbar may be changed using "drag and drop".

To return all toolbars to their original settings, click the **Load from Default** button.

Toolbar visibility

By default, toolbar visibility is automatically calculated by Genero Studio. When a toolbar contains an action associated with the active module or framework, that toolbar will be made visible.

A toolbar can have one of three visibility settings:

- **Automatic**
  - The toolbar displays when any action assigned to the toolbar is within the scope of the current module or by
Always visible

When set to "Always visible", Genero Studio displays the toolbar. The scope of any toolbar actions are not considered.

Always hidden

When set to "Always hidden", Genero Studio hides the toolbar. The scope of any toolbar actions are not considered.

Changing the toolbar's visibility setting

To change the visibility setting for a toolbar, locate the toolbar in the Visibility section. Click on the visibility value next to the toolbar name and a combobox appears. Select the desired visibility setting from the combobox.

Once you have made your changes, click the Apply button to apply your changes, or click the OK button to apply changes and close the Preferences dialog.

Menus configuration

Set preferences for menu configuration.

Expand a menu listed in the tree to display its actions.

Select a menu or action, and use the icons or right-click to display the menu options to:

• Add a new menu at the same level, or a submenu
• Add an action to a menu
• Remove or rename a menu
• Remove or rename an action
• Add or remove a separator

Item order may be changed using "drag and drop". This applies to menu order within the tool bar tree, and to icon order within one menu.

Accelerators (Shortcuts) configuration

Accelerators and their associated Genero Studio actions are defined in a default set called a profile.

Profile

Profiles correspond to a set of accelerators for each action. They are saved in Accelerator Profile files having an extension of apr. A default profile file is preinstalled in the Genero Studio installation directory. Initially, this is the only profile available, and it cannot be modified. You can create your own profile by duplicating the default profile and modifying the accelerators associated with an action. All the accelerators (except menu accelerators) are disabled during editing, and are re-enabled when editing is completed or you leave the Accelerators configuration window.

• current - This combobox displays the currently active profile, with the names of other available profiles displayed in a dropdown list. The default profile, which cannot be modified, has a lock icon. To change profiles, select a profile from the dropdown list, and click OK.
• Duplicate - Creates a new profile. The Duplicate Profile dialog allows you to enter the name of a new profile, which will be a copy of the currently active profile. The file is saved in this user directory: Documents and Settings\<username>\Application Data\<companyname>\Genero Studio\version>.

Modifications in the accelerators can be made in your new user profile. All the user profiles that you create will have a user icon associated with their name in the list in the current combobox.

• Remove - Removes the currently active profile. This removes the associated apr file. The default profile provided by Genero Studio cannot be removed.

Accelerator profiles can be shared with other users:

• Import - Imports an .apr file. An Open dialog allows you to browse for the profile file. When you click OK, the selected file is copied into the user directory and appears in the current combobox.
• **Export** - Exports an `.apr` file. A Save As dialog allows you to save your profile file.

**Actions**

The actions and the associated accelerators for a particular profile are displayed in a table, grouped by application by default. Expand the application group to display all its actions. The **Search** box allows you to locate a specific action in the tree.

You can sort the table to make the resolution of conflicts easier. Use the icons to switch between viewing the tree grouped by application, or in alphabetical order by action. To sort the tree by the accelerator column, click the column title.

**Accelerators**

The accelerators that are defined for an action display in a list. Use the icons or right-click for a menu of options to:

- **Add** a new, or additional, accelerator to an action using the Add New Accelerator dialog
- **Edit** the accelerator for an action using the Edit Accelerator dialog
- **Remove** the selected accelerator from an action
- **Set the selected accelerator as the default** for an action

**Add New Accelerator/Edit Accelerator dialog**

Enter your own key combination for the accelerator by hitting the keys in order; use the **Tab** button to enter a Tab key.

Use the **Backspace** key to erase the latest keypress, or the **Clear** button to erase the entire key combination.

**Compiler and Runtime preferences**

Set preferences for the compiler and runtime.

- **Run/debug configuration**
- **Compilation configuration**

**Run/debug configuration**

Set preferences for runtime and debug configuration.

**Console**

**Clear output before each launch**

Specify whether to clear the output after each launch.

**Logging**

**Enable proxy logging**

Enable proxy logging.

**Important**: Sensitive and personal data may be written to the output. Make sure that the log output is written to files that can only be read by application administrators.

**Compilation configuration**

Set preferences for compilation configuration

**General**

**Maximum number of parallel compilation tasks**

Specify the maximum number of parallel compilations that you want to allow. Although only one link at a time can be done, multiple file nodes can be compiled simultaneously. This speeds up the build process if the machine has multiple processors.
If you specify 0 maximum, Project Manager uses the number of processors available on the local machine (i.e. 2 for a dual core machine).

**Maximum number of erroneous files before stop compiling**
Specify the maximum number of files containing errors to be permitted before compilation is automatically stopped.

**Compute dependencies between files (recommended)**
This enables the computation of dependencies. They will be computed when the project is loaded, and is enabled by default. It can be useful to disable it for very large projects or projects on network drives with performance problems. However, it should ordinarily be enabled, in order for the build system to take the #include and globals into account.

**Compute additional information about functions**
Recommended.

**Save modified files before compile and build**
Specify whether to automatically save files before compile and build process.

**Verbose mode for build/link/execution rules**
Specify to display build, link, and execution rule commands in output.

**Output**
When selected, clears the Output view before each new compilation.

**Integrate your tools into Genero Studio**
To call your tools and scripts directly from Genero Studio, you can customize global menus and Toolbars and also the context menus that are used in the Project Manager and File Browser.

You can set up your commands to ask for parameters (using input variables) or use the preset parameters proposed by the Genero Studio environment (current file, current directory, and so on).

All commands run in the current environment set by the Genero Configuration or the Project. The working directory is set to the location of the file to which the command applies.

**Related concepts**
- [Menus configuration](#)
- [ToolBar configuration](#)

**Access Help**
Help is available within Genero Studio by selecting the **Help > Help** menu option, the `?` icon, or the F1 key.

**Configuring Genero Studio**
Configure Genero Studio to best meet your needs.

- [Software configuration scenarios](#)
- [Default configuration](#)
- [Setting up a local environment](#)
Software configuration scenarios

Genero Studio can be installed with all or some Genero components and for a local or remote environment.

Installations

You can install these Genero components when you install the Genero Studio software package:

- Genero Business Development Language (BDL)
- Genero Desktop Client (GDC) - to be used for display
- Genero Browser Client (GBC) - to be used for display
- Genero Web Services (GWS)
- Genero Application Server (GAS)

When you install the Genero Mobile software package, you install:

- Genero Studio
- Genero Business Development Language (BDL)
- Genero Mobile for Android™ (GMA), and/or Genero Mobile for iOS (GMI)

Local Environments

In a local environment with a single developer, the developer works individually with a complete Genero suite stack installed on the developer's local machine. The database may be on the local machine or may be accessed from a remote server. Some group development can be enabled through the use of an external Version Control System.
In a local environment with multiple developers, the Genero Studio, Genero DVM, and front-end client software (GDC/GBC) is on the local machines. These machines connect to database servers and source code on remote machines.

Remote Environments

In a remote environment, installation is done on the remote server and accessed by developers from their client machines.

- Genero applications are compiled, debugged, and executed on a remote server.
- Genero Studio Server must be installed on the remote server.
- Allows for the preservation of C function calls/libraries and/or system calls.
- Requires SSH access as well as Samba/NFS mounts.

The relational databases that the applications access can be stored on the same remote server as Genero Studio server, or on separate servers.
### Default configuration

Configurations allow Genero Studio to locate the installed Genero components necessary to create, compile, build, and execute Genero applications on local or remote hosts.

Some default configurations are provided, based on the software in the distribution package installed in the default directories.

If you have installed your own Genero software, or have not installed Genero Studio in the default directories, you will have to define a Genero Studio configuration.

Even if you use the default configuration, you must set the environment set appropriate for your database client software, and add any missing variables or values. See Access a database on page 189.

### Change the active configuration

When you have multiple configurations defined, only one of the configurations is active.

The list in the bottom right corner of the main Genero Studio window displays the currently active server and client configuration.

To change the active configuration, select a server and/or display client configuration from the lists.
Configuring for BAM

When using the Business Application Modeler, you can specify both a global setup and a template-specific setup.

**Factory setup**
Factory setup is the standard installation.

**Global setup**
The global setup overrides the factory setup. It applies to all configurations for the current user for the Genero Studio installation on the current machine or remote machine.

The **Tools > Global setup** menu option allows you to customize the global setup.

**Specific setup**
A specific setup is a template-specific setup. It is for the current user, is for a template set used by the Business Application Modeler. Use is option to override the global setup for a specific set of templates. The GSTSETUPDIR on page 168 environment variable, set in the Genero configuration, defines the template set. If this environment variable is not set, or not specified for the current configuration, the specific setup options are disabled.

The **Tools > Specific setup** menu option allows you to customize the specific setup.

Configure for Genero Mobile

Follow these instructions to complete the configuration for Genero Mobile for Android™ or iOS.

- Configure Genero Mobile for Android on page 142
- Configure Genero Mobile for iOS on page 151
- Display to the Genero Mobile Development Client on page 162

Configure Genero Mobile for Android™
The Android configuration allows you to run a program from Genero Studio and display it to your Android™ physical or virtual device.

**Requirements**
- Java SDK. (Installed as part of Install and configure Java SDK and Android SDK on page 143)
- Android™ SDK. (Installed as part of Install and configure Java SDK and Android SDK on page 143)
• Either
  • An Android™ emulator (Android™ Virtual Device) (Created as part of Display to an Android virtual device on page 145)
  • An Android™ mobile device that runs a minimum of Android™ 4.0 (Ice Cream Sandwich) and is connected to the development machine via USB.
• Genero Mobile, which includes:
  • Genero Mobile for Android™ (GMA).
  • Genero Studio (launched when launching Genero Mobile) configured to communicate with the device or emulator.
  
  **Note:** Genero Studio included with the Genero Mobile bundle is customized for mobile app development.
• You need an internet connection the first time you build an Android™ package. During this first build, an automated process will download and install Gradle with all necessary extensions into a directory in your user directory. Gradle is a project automation tool, find out more about Gradle at http://www.gradle.org.

**Install and configuration topics**
• Display to an Android physical device on page 147
• Configure multiple Android display devices on page 149

**Install and configure Java SDK and Android™ SDK**
Follow this procedure to install and configure the Java SDK and Android™ SDK. Once configured, the Tools > Android Tools menus are enabled in Genero Studio.

**Before you begin:**
• If a proxy is needed on your network, it must be defined in Tools > Preferences.

You must configure for the Java and Android™ SDKs.

1. Download the Java Standard Edition Software Development Kit and install it to a location of your choice.
   
   To download the Java package, visit http://www.oracle.com/technetwork/java/javase/downloads/index.html. Choose the Java package (32 bit or 64 bit) that matches your Genero installation (32 bit or 64 bit).
   
   **Note:** Genero Mobile requires Java SE Development Kit. To identify version requirements, refer to the Supported platforms and database document found on the Four Js Web site.

   Make a note of the installation path.

2. Launch Genero Studio.

3. Select Tools > Genero Configurations to open the Genero Configuration Management window.

4. Set the JAVA_HOME variable to the location of the Java SDK directory. To edit the JAVA_HOME environment variable, highlight the Java SDK environment set and double-click on the JAVA_HOME environment variable listed.
5. Download the Android™ Software Development Kit (SDK).
   a) Go to http://developer.android.com/studio/index.html#command-tools
   b) Find the Get just the command line tools section.
   c) Download the installer for your operating system.

6. Install the Android™ SDK to a location of your choice. Do not install in the Program Files directory.

   **Note:** If you install Android™ SDK in the Program Files directory, Genero Mobile must be run as administrator to auto-configure the SDK.

   Note your installation path. If the Android™ SDK Manager launches, close it.

7. Set the ANDROID_SDK_ROOT variable to the location of the Android™ SDK you installed, for example C:\Android\android-sdk. To edit the ANDROID_SDK_ROOT environment variable, highlight the Android environment set and double-click on the ANDROID_SDK_ROOT environment variable in the list.

8. Right-click on the Android configuration in the Configuration Name list and select Set Genero Configuration Active.
9. Select OK to save the changes.

10. Select Tools > Android Tools > Auto-configure Android SDK. Update progress is written to the Output view. When prompted, answer ‘y’ to any question. This step can take several minutes.

    **Important:** It is recommended to run Auto configure Android SDK regularly to ensure you have the latest SDK updates. The Auto configure Android SDK tool upgrades Android™ SDK and add-ons to the latest version.

11. You are now ready to setup your Android™ physical or virtual device. See Display to an Android physical device on page 147 or Display to an Android virtual device on page 145.

You need an internet connection the first time you build an Android™ package. During this first build, an automated process will download and install Gradle with all necessary extensions into a directory in your user directory. Gradle is a project automation tool, find out more about Gradle at [http://www.gradle.org](http://www.gradle.org).

**Configuration for extending Genero Mobile for Android™**

Follow this procedure if you plan to extend Genero Mobile for Android™.

This procedure is only necessary if you plan to extend Genero Mobile for Android™ using Java. See Extending the Language in the Genero Business Development Language User Guide.

1. Create a copy of the GMADIR directory.
   
   Find the default GMADIR at INSTALL_DIR/gma. You should never update the default GMADIR, as it can be overwritten during an upgrade.

2. Update the GMADIR environment variable to point to your new copy.
   
   a) Select Tools > Genero Configurations to open the Genero Configuration Management window.
   
   b) Set the GMADIR environment variable to the location of the copy of the GMA directory. To set the GMADIR environment variable, highlight the Android environment set and double-click on the GMADIR environment variable in the list.

   ![Figure 85: Setting the GMADIR](image)


**Display to an Android™ virtual device**

This configuration allows you to run a program from Genero Studio and display it to your Android™ virtual device (AVD).

Before you begin:

- Meet the requirements for Android™ emulator hardware acceleration. The emulator requires a processor with virtualization technology and a dedicated driver. Most recent Intel® processors support virtualization (VT-x, EM64T).

1. Install and configure Java SDK and Android SDK on page 143.

2. Unplug any Android™ hardware devices connected via USB or Configure multiple Android display devices on page 149.
3. Select the Genero Configuration for **Android**.

![Figure 86: Genero Configurations list](Image)

You are now ready to create a virtual device and install Genero Mobile for Android™ (GMA) onto the virtual device.

**Tip:** Genero Mobile provides an 'Android™' Virtual Device (AVD), however you can elect to use another AVD, such as Genymotion.

**Tip:** If you create your own emulator, you must specify an external storage > 100 MB.

4. Select **Tools > Android Tools > 'Create Android Virtual Device (x86)'**. When the console appears, you will be asked if you wish to create a custom hardware profile. Press Enter to accept the default answer, no.

5. The Intel(R) Hardware Accelerated Execution Manager (HAXM) must be installed on the system. Go to your `ANDROID_SDK_ROOT/extras/intel/Hardware_Accelerated_Execution_Manager` directory and double-click on the application to install it.

HAXM will improve x86 virtual device performance on Windows® and macOS™.

6. Select **Tools > Android Tools > Launch Android Emulator (x86)** and wait for the emulator to finish loading. This can take a few minutes if you did not install HAXM.

You will know it is finished loading when the emulator looks like a device screen.

7. On the AVD, confirm that it allows installation from unknown sources. Select **Settings > Security** and confirm that **Unknown sources** is selected. Return to the Home screen and then go to the Apps screen.

8. From Genero Studio, select **Tools > Android Tools > Deploy Genero Mobile for Android**.

In the Genero Studio output panel you will see that the deploy started and finished. On the AVD you will see a new icon on the Apps screen labeled Genero Mobile.

9. Test your configuration. From Genero Studio, find the **OfficeStoreMobile** project and open it. Execute the **OrdersApp**. You should now see the Orders program running on the virtual device.

You can now run your own apps from Genero Studio to your virtual device. Your app will display to the virtual device currently running or you can **Configure multiple Android display devices** on page 149.

**Troubleshoot Android™ SDK issues**

Here are troubleshooting tips for issues you may encounter when using the 'Android™' SDK during the setup of your Genero Mobile development environment.

You follow the instructions to **Install and configure Java SDK and Android SDK** on page 143 and **Display to an Android virtual device** on page 145 and you get an error. What steps should you take?

1. Ensure that you have set the `ANDROID_SDK_ROOT` environment variable correctly.
2. Run the auto-configure script for 'Android™' SDK (**Tools > Android Tools > Auto configure Android SDK**).

**Troubleshoot Android™ emulator issues**

Here are troubleshooting tips for issues you may encounter when using an Android™ Emulator during the development of your Genero Mobile app.

If you are having emulator issues, we recommend you read **Using the Emulator** on the Android™ Developers site.

**My Android™ Emulator crashes intermittently**

You followed the instructions to display to an Android™ virtual device, and you are getting intermittent crashes of the Android™ emulator when starting up. What is the issue?

It has been discovered that if you have an external monitor attached, or if you are moving your emulator between monitors in a dual monitor environment, it would save the location, which it would then use to restore when starting up. The location would then be read as an unexpected value, and would cause a core dump. To solve when this is the issue:
1. Navigate to the Android™ virtual device folder (.android/avd on a Mac, userdir\.android\avd on Windows).
2. Navigate to your Virtual Device (for example, Nexus7.avd).
3. Edit emulator-user.ini and set:
   • window.x = 0
   • window.y = 0

If this is not the problem, then you may wish to try a different Android™ emulator, such as Genymotion. Please check the system requirements before making the change!

To view the Android™ bug number for this issue: http://code.google.com/p/android/issues/detail?id=40556

**My Android™ Emulator is too slow**

Mobile emulators are virtual machines. As such, we highly recommend a desktop or laptop with CPU virtualization support and sufficient memory. If you do not have this, you will find the mobile emulators will run very slowly.

**Important:** The following system requirements are recommendations based on user experiences.

For Mac® users:
- Mac® or MacBook Pro® (Intel® based)
- macOS™ 10.7 or higher
- 8+ Gb RAM
- VT-x enabled (may require firmware upgrade, see http://support.apple.com/kb/TS2744

For Windows® users:
- Windows® 7 or higher
- 8+ Gb RAM
- Intel® CPU with VT-x or AMD-v CPU
- VT-x or AMD-v enabled (may require BIOS configuration)
- You can check if your CPU has VT-x with this Intel® CPU identification tool: http://www.intel.com/support/processors/tools/piu/sh/CS-014921.htm

At this time, we have no statistics regarding system requirements for Linux® users.

**Display to an Android™ physical device**

This configuration allows you to run a program from Genero Studio and display it to your Android™ device.

**Note:** These configuration instructions are for a Google Nexus 7, the default platform used, and are provided as an example. Check the device manufacturer documentation or web site for information related to your specific device.

1. Install and configure Java SDK and Android SDK on page 143.
2. Close the Android™ emulator if it is currently open or Configure multiple Android display devices on page 149.
3. Select the Genero Configuration for Android.
   You are now ready to install Genero Mobile for Android™ (GMA) onto your device.
4. Connect your Android™ device to your computer's USB port.
5. To deploy and validate your app to a device, you must configure the device for development. Enable USB debugging on your device by checking the Allow USB debugging option. Select Always allow from this computer so that you do not have to repeat this step.

On Android™ 4.0 and newer, it’s in Settings > Developer options. On Android™ 4.2 and newer, Developer options is hidden by default. To make it available, go to Settings > About phone (or About tablet) and tap Build number seven times. Return to the previous screen to find Developer options and to allow USB Debugging.
Figure 87:

6. Ensure your device uses Media Transfer Protocol (MTP) as the connection mechanism. Your Android™ device may use MTP by default, or you may need to set it manually.

   **Tip:** Go to **Settings > Storage > USB computer connection** or **Settings > Storage and USB**. To see the internal storage, set the USB computer connection mode to “Media device (MTP)”. If “Media device (MTP)” is not an option, your device probably uses MTP by default and you do not need to select anything.


8. Set your device to allow installation from unknown sources. Select **Settings > Security** and tap **Unknown sources** to select it. Return to the Home screen and then go to the Apps screen.

Figure 88: Install from unknown sources

9. Check the communication from the computer to your device. From Genero Studio, select **Tools > Android Tools > List Devices**. In the output panel you should see a list of devices attached. If several devices are connected,
you need to define device ID in the display client configuration. If a device is offline, restart the state Allow USB debugging.


In the output panel you see that the deploy started and finished. On the device, a new icon appears on the Apps screen labeled Genero Mobile. This is the development client.

11. Disable installation from unknown sources on your device. Select Settings > Security and tap Unknown Sources to deselect it.

12. Test your configuration. From Genero Studio, find the OfficeStoreMobile project and open it. Execute the OrdersApp. You should now see the Orders program running on your device.

You can now run your own apps from Genero Studio to your device. You can also deploy an app to the device. See Deploy a mobile app for testing on page 1178 for details.

Configure multiple Android™ display devices

Use this procedure to configure multiple Android™ display devices. For example, you may have an Android™ phone, tablet, and emulator.

By default, the device ID is not set in the Display client configuration and Genero Studio uses the default configuration for the first connected device it finds.

If you have multiple devices connected (by USB or as emulators started from Android™ Studio), create separate configurations where each configuration specifies one of the connected devices. The device is identified by its device ID.

1. Select Tools > Android Tools > List Devices. The list of connected devices appears in the Output view along with their respective device IDs.

   Android™ virtual devices are considered devices by this utility and appear in the list.

   ::info:(GS-09007) Listing Android devices
   List of devices attached
   emulator-5556       device
   emulator-5554       device
   06dc3f26            device
   ::info:(GS-09007) List devices finished successfully

2. Copy the device ID of a device.

   For example, a device ID may read "emulator-5554".

   **Tip:** If you have multiple devices listed and you are not sure which one is the right one, disconnect all of them except the one you wish to configure.

3. From the Genero Configuration Management dialog, click the Edit display client configurations icon, located next to the Use Display Client combobox.

4. In the Display Client management dialog, enter the ID into the Device ID field. You may edit the default configuration or add new Display Client configurations.

   Create a separate Genero configuration for each Android™ device you have connected.

   a) Add a new Display Client configuration.
   b) Enter the device ID into the Device ID field.
   c) Enter a port number for the device in the Port field.

   We recommend having a different port for each client available on the client machine. For example:

   - 6500 for an Android™ Nexus 7 device
   - 6501 for an Android™ Nexus 5 device
   - 6502 for an Android™ Emulator

   and so on.
You can have all devices connected and the emulator started. Selecting a Genero Configuration now specifies which device to use.

**Update GMA in existing GST installation**

Use this procedure to update a Genero Studio installation for a different version of Genero Mobile for Android™ (GMA).

1. Download the required version of Genero Mobile for Android.
   To download the latest release, visit the Products page of the Four Js web site.
2. Extract the downloaded package to a new directory.
   
   **Warning:** Do not extract to or overwrite the default GMADIR of the Genero Studio installation, as it could be overwritten if you install a new Genero Studio into the same directory.
3. Select Tools > Genero Configurations to open the Genero Configuration Management dialog.
4. Update the GMADIR environment variable of the Android™ environment set to reference your new GMA directory.
Figure 89: Android Environment Set

5. Select Tools > Android Tools > Deploy Genero Mobile for Android to update your device with the new GMA and/or rebuild and redeploy your package.

If you encounter an issue when building the .apk package, make sure to update your Android™ SDK with the missing features.

Configure Genero Mobile for iOS

Requirements for configuring Genero Mobile to develop apps for iOS.

Requirements

This configuration allows you to run a program and display it to your iOS physical or virtual device.

- Genero Mobile, which includes Genero Studio customized for mobile app development
- A Mac® running Mac® OS.

For a detailed list of supported operating systems, refer to the Supported platforms and databases document (available on the Products download page of the Four Js Web site) or contact your support center. This document also informs you which operating systems will no longer be supported as of the next release.

- Xcode. See Install and configure Xcode on page 152.
- Either the iOS Simulator or an iOS physical device connected to machine via USB
- A standard, freeApple® ID account or a purchased developer account

Note: A developer account is not required for creating apps that run on the simulator. You will need an account once you want to:

- Install an app on an iOS device (even your own one for debugging)
• Distribute an app on the iOS app store
• Distribute in-house iOS apps to your employees
• Genero Studio (launched when launching Genero Mobile) configured to communicate with the device or simulator

Install and configuration topics
• Install and configure Xcode on page 152
• Display to an iOS simulator on page 152
• Display to an iOS physical device on page 153
• Configure multiple iOS display devices on page 160
• Display to the Genero Mobile Development Client on page 162

Install and configure Xcode
Follow this procedure to configure for iOS. Once configured, the Tools > iOS Tools menus are enabled in Genero Studio.

1. Download and install Xcode from the Mac app store.
2. Install the Xcode command-line tools.
   They can be installed from Xcode by selecting Xcode > Open Developer Tool > More Developer Tools.
3. Launch Genero Studio.
4. Select Tools > Genero Configurations to open the Genero Configuration Management window.
5. You are now ready to Display to an iOS simulator on page 152 or Display to an iOS physical device on page 153.

Configuration for extending Genero Mobile for iOS
Follow this procedure if you plan to extend Genero Mobile for iOS (GMI).

This procedure is only necessary if you plan to extend Genero Mobile for iOS using Objective-C.

1. Install a new GMI.
2. Update the GMIDIR environment variable to point to your new GMI.
   a) Select Tools > Genero Configurations to open the Genero Configuration Management window.
   b) Set the GMIDIR environment variable to the location of the new GMI directory. To set the GMIDIR environment variable, highlight the iOS environment set and double-click on the GMIDIR environment variable in the list.
3. Follow the instructions in the Genero Business Development Language User Guide to extend Genero Mobile for iOS.

Display to an iOS simulator
This configuration allows you to run a program from Genero Studio and display it to your iOS Simulator. This configuration does not require an Apple iOS developer account.
Before you begin:
• Install and configure Xcode on page 152.
1. Launch Genero Studio.
2. Confirm that the Genero Configuration for iOS Simulator is selected.
   Note: The Device ID field in iOS Simulator is used to specify which simulator to use when deploying application or Genero Mobile Client, or launching Genero Mobile Client when executing or debugging an application.
You can see the list of simulator identifier by calling **Tools > iOS Tools > List Devices** action. You access the **Device ID** field by opening **Genero Configuration Management** dialog then opening the list of display clients and selecting an iOS Simulator display client. The value you can put in this field should be a substring of a simulator as displayed by **List Devices** action.

It can be the internal identifier or the label or a part of it. If the field is not filled, then “iPhone 4S” simulator will be used, as it is the less resource-intensive one.

3. **Select Tools > iOS Tools > Launch iOS Simulator.** The simulator should appear. Change the simulator hardware to match the device you want to simulate, such as an iPhone.

4. **Select Tools > iOS Tools > Deploy Genero Mobile for iOS.**
   
   In the Genero Studio output panel you will see that the deploy started and finished. On the Simulator you will see a demo app running.

5. **Test your configuration.** From Genero Studio, find the **OfficeStoreMobile** project and open it. Execute the **OrdersApp.** You should now see the Orders program running on the simulator.

6. You can now run your own apps from Genero Studio to the simulator.

### Display to an iOS physical device

This configuration allows you to run a program from Genero Studio and display it to your iOS device. This configuration **does** require an Apple® iOS developer account.

Before you begin:

- **Install and configure Xcode** on page 152.

**Note:** A USB connection is needed when you register a device in your iOS developer account in order to retrieve the device ID, which is necessary for listing devices in Genero Studio.

1. **Log into the iOS Developer Center.** If you are a member of the iOS Developer Center then just log in. If not, follow these steps to enroll in an iOS developer program. Please note, it can take Apple a day or two to approve memberships.
a) Go to http://developer.apple.com and select the iOS Developer Program option.
b) Select the Enroll Now option and follow the instructions. You will need to enroll in either a Developer or Enterprise program. When you are finished you will have an Apple ID with which to log in. If you are a member of the IOS Developer Program select Member Center to log in.

2. Generate and import a development certificate. The Development Certificate will be used to allow you to view your program on your iOS device which can be an iPhone, iPod, or iPad.
   a) Select Certificates, Identifiers & Profiles.
   b) Select Certificates under the iOS Apps section.
c) Under **Certificates**, select **Development** followed by the + symbol.

d) Follow the instructions, on the **Select Type** page and select the **iOS App Development** option for an iOS App Development certificate then click **Continue**.

e) Follow the instructions on the page to create a **CSR file** then click **Continue**.
f) Your certificate request is now available. Now you can go back to the Development Certificate section still active in your browser and click Choose file....

g) Navigate to the file you just saved and choose that file.

h) Click Generate.

i) Once the certificate is generated, click Download. The certificate will download into your Downloads folder.

j) Double-click this file to install it into Keychain.

k) When done, your new certificate should be listed in the Certificates list.

3. Provision your device for development. If you have only an iOS developer account, you need to register each device you will use to test your iOS app. These steps are not necessary if you have an iOS enterprise account. Follow Apple's instructions for provisioning your device.

4. For iOS 8 and above, enable UI automation.
   a) Open the Settings app.
   b) Select Developer.
   c) Turn on Enable UI automation.
5. Go back to the Member Center and you should have a provisioning profile granted. Please note, it can take some time before Apple® changes the status from Pending to Active.
6. Select the Provisioning Profile and click **Download**.

7. Launch Genero Studio.
8. Select Tools >> Genero Configurations.
9. Select the **iOS** environment set.
10. Set the **IDENTITY** environment variable in the **iOS** environment set, to identify which certificate to use from those defined in Keychain® Access.
The certificate name generally contain the first and last name of the developer as defined in your Apple® ID account. You can find it at http://appleid.apple.com/.

To view the list of certificates in the Keychain® Access, leave Genero Studio and complete these steps:

a) Open Applications > Utilities > Keychain Access.

b) Select login in the Keychain section.

c) Select My certificates in the Categories section.

The list of certificates displays. Use this list to find the sub string to enter for the IDENTITY environment variable. At a minimum, the IDENTITY environment variable must contain just enough characters to identify the certificate (amongst those listed in Keychain® Access) used to sign the package. In theory, it can be as small as two letters, if those two letters are sufficient enough to identify the certificate.

11. Set the PROVISIONING_PROFILE environment variable to the provisioning profile file you downloaded.

12. Select Tools > iOS Tools > Deploy Genero Mobile for iOS. A USB connection is required to deploy to the device.

In the Genero Studio output panel you will see that the deploy started and finished. On the device you will see a new icon labeled GMI.
13. Set up your device as a display client. Wi-Fi is used to display the app to the device in developer mode.
   a) Confirm that your computer and your mobile device are on the same Wi-Fi network.
   b) Get the IP address of your device. (From your device, select Settings > Wi-Fi and select the Wi-Fi network to see the network details and IP Address.)
   c) Edit the iOS Display client setting for the iOS Genero configuration. In the Host field, enter your device's IP address.
   d) Select OK to save the changes.

14. On the device, tap the GMI app to launch it.
15. Test your configuration. From Genero Studio, find the OfficeStoreMobile project and open it. Execute the OrdersApp. You should now see the Orders program running on the device.
16. You can now run your own apps from Genero Studio to the device.

Configure multiple iOS display devices
Use this procedure to configure multiple iOS display devices, for example a phone and a tablet.

By default, the device ID is not set in the Display client configuration and Genero Studio uses the default configuration for the first connected device it finds.

If you have multiple devices connected, create separate configurations where each configuration specifies one of the connected devices. The device is identified by its device ID.

1. Select Tools > iOS Tools > List Devices. The list of connected devices appears in the Output view along with their respective device IDs.

   List of iOS devices
   iPad: 93f794ca3b3f9aee092980aba8410102fffffff
   iPhone: 62d5ce684b97317379072f81b796eeceeeeee

2. Copy the device ID of a device.
3. From the Genero Configuration Management dialog, click the Edit display client configurations icon, located next to the Use Display Client combobox.
4. In the Display Client management dialog, enter the ID into the Device ID field. You may edit the default configuration or add new Display Client configurations.
5. If you have more than one USB device connected, create a separate Genero configuration for each device. Selecting a Genero Configuration will select the device to use.

**Update GMI in existing GST installation**

Use this procedure to update a Genero Studio installation for a different version of Genero Mobile for iOS (GMI).

1. Download the required version of Genero Mobile for iOS.
   To download the latest release, visit the Products page of the Four Js web site.
2. Extract the downloaded package to a new directory.
   **Warning:** Do not extract to or overwrite the default GMIDIR of the Genero Studio installation, as it could be overwritten if you install a new Genero Studio into the same directory.
3. Select Tools > Genero Configurations to open the Genero Configuration Management dialog.
4. Update the GMIDIR environment variable of the iOS environment set to reference your new GMI directory.
5. Select Tools > iOS Tools > Deploy Genero Mobile for iOS to update your device with the new GMI and/or rebuild and redeploy your package.

**Display to the Genero Mobile Development Client**

Available from Apple's App Store, the Genero Mobile Development Client serves two purposes: It allows you to view your app on an iOS device regardless of your development machine operating system, and it allows you to run a demo app.

Before you begin:

- The mobile device and the development machine must be connected to the same wireless network.
- The Genero Mobile Development Client app (available from the App Store) must be installed on the mobile device. To locate the Genero Mobile Development Client in the App Store, simply search on the full name of the app. After installation, the app appears with the name **Dev Client** and the standard Genero icon.

1. Launch Genero Studio.
2. Select the **iOS Dev Client** configuration.

The iOS Dev Client is configured for the DVM (fglrun) to listen on port 6400. To listen on a different port, modify the **Genero Installation FGL options** setting to a different port.
3. Run your app.
   • Execute your app from Genero Studio. If you have a firewall, you must allow the client to communicate on your network.
   • Alternatively, start the app on the server from the command line using the `--gui-listen` option. For example:

   ```
   % fgirun --gui-listen=6400 helloworld.42m
   ```

   The app starts and waits for the Genero Mobile Development Client to connect.

4. From your device, start the Dev Client app.

5. In the URL field, enter the following URL:

   ```
   fgl://yourdesktop-ip:port-number
   ```

   Replace `yourdesktop-ip` with the IP address of your development machine. Replace `port-number` with the port number specified in the iOS Dev Client configuration. The default is 6400. The firewall on your desktop needs to be open for the port used. For example:

   ```
   fgl://192.168.0.101:6400
   ```

6. Set the Timeout and Permanent retry options as needed. For the Timeout option, use the spinner to set the number of seconds for the timeout. Setting the timeout to zero indicates that you are not setting a time limit. If you have no service waiting, it will immediately fail. Turn Permanent retry on or off. If you switch on, and kill the app, then it tries for some seconds to connect, then retries and keeps retrying.

7. Tap Connect.
   Either the app will display, or you will get an error message in the RESULTS area of the interface. When you exit the app, you are returned to the developer interface.

### Setting up a local environment

These topics assist you in setting up a local Genero Studio environment.

- Local environment software requirements on page 164
- Define local Genero installations, GAS configurations, and environment sets on page 164
Local environment software requirements

Install the complete Genero stack on the local machine when a developer will work independently and compile files locally.

A local environment requires the following components:

- Genero Studio
- Installed display client such as GDC, GBC, GMI, GMA depending on your application development (desktop, web, mobile). See Configure the display clients on page 170.

If the project requires a database, the database (Informix®, Oracle®, etc.) and its accompanying database client (CSDK, Oracle® client, etc.) can be installed on the local machine or configured for remote access:

- The database server installed on a remote host
- The database client installed on the local machine

Project and source files must be local, but can be shared using a version control system if a networked drive (or samba) is configured.

Each database environment has its own client character set configuration. You must configure the database client locale properly in order to send/receive data to the database server. The settings for database client locale and application locale must match. See Localization: Database Client Settings in the Genero Business Development Language User Guide.

See the installation documentation and release notes for additional information.

Define local Genero installations, GAS configurations, and environment sets

Local default configurations based on your installation are set up for you. Modify the configurations if the defaults are not set up for your needs.

1. Select Local Host as the active server.

![Figure 92: Local Host selected from the Server connections combobox](image)

2. Select Tools > Genero Configurations to display the Genero Configuration Management dialog for the local host.

3. Add new configurations as needed to add Compiler / Runtime configuration (Genero Installations) on page 200, Environment sets on page 164, and Configure for the Web client on page 171.

4. Save the changes.

Related concepts

Genero Configuration Management dialog on page 197
Genero Studio configurations provide the information needed to run on your local or remote hosts.

Environment sets

Define sets of environment variables to add to or overwrite the default environment. These environment settings are also usable by each Genero Studio client that adds a remote Genero Studio Server to its list.

Select Tools > Genero Configurations to display the configurations for the currently selected server. Add and modify environment sets as needed.

When you launch any Genero Studio sub-process (run, GDC, compilation, debug, and so on), Genero Studio sets all redefined environment variables according to the current context (the Projects context may differ from the File Browser context).
Select the name of an environment set to view/modify the Environment Set variables. Right-click in the Environment Set area and select Add Environment Set... to add a new set of environment variables.

Figure 93: Environment sets

See Add or edit environment variables on page 168 for more details on using the Environment Variable dialog.

Default environment sets

Some default environment sets are included.

**Debug**

Contains variables that set the debug level  
(FGLSQLDEBUG for FGL/SQL, FGLWSDEBUG for Web Services, and GREDEBUG for Report Engine.) The values are already set to 9, the highest debug level, but can be changed to any number between 1 and 9. Debug files will be generated in the current directory.

**DB - xxxx**

Contains some variables required by database clients from the various vendors. The values for the variables must be entered before using these environment sets. See Database server/user information on page 365 for information about the specific variables for each database.

**Important**: This feature is not supported on mobile platforms.

**Global Database Schemas**

The GSTSCHEMANAMES environment variable defines the schema files (4db, 4dbx, sch) to be loaded by default into the Meta-schema Manager when Genero Studio is launched. Includes setting for GSTSCHEMANAMES and FGLDBPATH.

**Java SDK**

Environment settings for the installed Java SDK. Adds the Java executables path to the PATH environment variable. See Install and configure Java SDK and Android SDK on page 143.
Profile

Contains the environment variable FGLPROFILE; for the value, enter the path of the current configuration (FGLPROFILE) file to be used by the system before enabling this environment set; the default location is FGDLDIR/etc/fglprofile. If you are using a remote environment, the value should be set to the location on the remote host. See Access a database on page 189.

Report Writer

Contains the environment variables set by default to the installation directory of Genero Report Engine. This set is automatically enabled, as the environment variables are required when working with reports.

**Important:** This feature is not supported on mobile platforms.

Templates

Contains environment variables set by default to the template directory being used for Business Application Modeling. Includes setting for GSTSETUPDIR. See The code generation template set on page 1110.

Web Components

Defines GSTWCDIR, the directory in which the web component XML files and optional icon files are stored.

Android™

For Genero Mobile, environment settings for the installed Android™ SDK. See Install and configure Java SDK and Android SDK on page 143.

iOS

For Genero Mobile, environment settings for iOS. See Configure Genero Mobile for iOS on page 151.

Mobile Debug Packages

For Genero Mobile, sets the DEBUG_PACKAGE environment variable to TRUE so that packages built will include options for debugging. See Debugging a mobile app on page 615.

Locale

Sets the LANG environment variable. See Language support (text encoding) on page 189 and Language and character set settings in the Genero Business Development Language User Guide for more information.

Studio Libraries

Contains the GSTLIBRARYDIR environment variable which specifies the location of the libraries used with the database generation script.

Term

Contains the GSTTERM (Linux, Windows) environment variable which specifies how to open a terminal on the client machine (Genero Workspace Window, for example). For some Linux operating systems which do not have xterm, use this to specify the terminal name and the arguments to launch it. The default is xterm -e (cmd /K).

**GST-specific environment variables**

This section lists and describes all Genero Studio specific environment variables.

See the topic Genero environment variables in the Genero Business Development Language User Guide for Genero-specific environment variables such as FGDLDIR, FGLGUI, and FGLPROFILE.
See GRE environment variables on page 1013 for environment variables specific to the Genero Report Engine, such as GREDIR and GREDEBUG.

**GMADIR**
The GMADIR environment variable defines the installation directory of Genero Mobile for Android™ (GMA).

**GMIDIR**
The GMIDIR environment variable defines the installation directory of Genero Mobile for iOS (GMI).

**GMIEXTDIR**
The GMIEXTDIR environment variable indicates the location of the iOS extensions to be taken into account when building Genero Mobile for iOS (GMI) packages from within Genero Studio.

The extensions must be compiled before they are put into the GMIEXTDIR directory.

For example, rather than having a .m file (uncompiled version), you need to provide a static library with a .a extension. For example, if you have a file frontcalls.m, you first must compile the file by using the Makefile-gmi generic makefile:

```
make -f "$GMIDIR/lib/Makefile-gmi" USER_LIBNAME=frontcalls.a staticlib
```

This command creates the file frontcalls.a, which is the file to be placed in GMIEXTDIR.

This command provides a mechanism for implementing the \(--extension-libs\) option of the gmibuildtool. For more information about this option or about compiling the required static libraries, see the section Building a GMI app with C extensions or custom front calls in the Building iOS apps with Genero topic of the Genero Business Development Language User Guide.

**GSTDIR**
The GSTDIR environment variable defines the installation directory of Genero Studio.

**GSTLIBRARYDIR**
The GSTLIBRARYDIR environment variable defines the location of the libraries used with the database generation script. By default, this environment variable is set in the Studio Libraries environment set.

**Related tasks**
Generate a 4gl source file to be used to create or update a database that is described in the meta-schema file.

**GSTSCHEMANAMES**
The GSTSCHEMANAMES environment variable defines the schema files (.4db, .4dbx, .sch) to be loaded by default into the Meta-schema Manager when Genero Studio is launched. Available meta-schemas are displayed in the DB Schemas tab, and available to Genero Studio components such as Form Designer.

Use the GSTSCHEMANAMES environment variable to specify global schemas.

**Note:** Although you can specify global schemas, it is recommended that you add schemas to projects instead. Schemas added to projects are loaded when the project is opened, not at Genero Studio launch. Project can also be available to all developers without any additional configuration needed.

Select the default environment set (Global Database Schemes) or create a new one that includes the GSTSCHEMANAMES environment variable. Set the GSTSCHEMANAMES environment variable to specify the file names of the schemas to make available. Use the Value List environment variable type to list multiple schemas, separated by semicolons. Do not include the file extension.

Use FGLDBPATH to define the directories in which to find the schema files listed in the GSTSCHEMANAMES variable.
**GSTSETUPDIR**
The **GSTSETUPDIR** environment variable defines the BAM application generator template directory. Changing this variable launches synchronization from the server and reloads the templates.

Select the default environment set or create a new one that includes the **GSTSETUPDIR** specifying the location of the template directory to be used.

**Related concepts**
*Environment sets on page 164*
Define sets of environment variables to add to or overwrite the default environment. These environment settings are also usable by each Genero Studio client that adds a remote Genero Studio Server to its list.

**GSTUSERSAMPLESDIR**
The **GSTUSERSAMPLESDIR** environment variable defines the installation directory for the Genero Studio demo samples.

**GSTUSERSAMPLESDIR** is typically used when setting other environment variables.

By default, the **GSTUSERSAMPLESDIR** is not used in Genero Enterprise v3.0.

**GSTTERM**
The **GSTTERM** environment variable defines the command to open a terminal on a Linux® system.

**GSTWCDIR**
The **GSTWCDIR** environment variable defines the directory in which the WebComponent XML files (`.wcsettings`) and optional image files are stored. By default this environment variable is set in the Web Components environment set.

Set **GSTWCDIR** to your web component directory where the `.wcsettings` and the optional image files reside. The Web Components files themselves may reside in a separate directory for deployment.

Each of the `.wcsettings` XML files describes a single WebComponent object.

When you have set this directory, you may add a WebComponent widget to your form design document. The components described in the `.wcsettings` files will be available in the combobox list of the `componentType` property in the Properties view, allowing you to specify the particular WebComponent to add to the form.

**Add or edit environment variables**
The Environment Variable dialog is used to add and edit environment variables.

When the Environment Variable dialog appears, enter:

**Type**
The type of environment variable. Options are Value, Value List, Directory, Directory List, File, or File List.

**Name**
The name of the environment variable.

**Value**
The value of the environment variable. When entering the value, if the type is Value List, Directory, Directory List, File, or File List, select the ellipses (...) to browse for the correct value.

If the value contains a variable name, that name must be prefaced with $ and enclosed in parenthesis; for example `$ (FGLLDPATH)`.
The list separator is always a semicolon (;) on all systems (Windows® and UNIX™). The directory separator in a path is always a slash (/) on all systems.

Tip: Use the semicolon to separate directories in a list, and the slash (/) as the separator in a path, for portability of projects across operating systems.

Figure 94: Setting FGLLDPATH

Figure 95: Setting FGLPROFILE

Reusing existing environment variables

A variable defined for a parent or ancestor node can also be reused in definitions for a child node:

For example:

- The parent node defines: MY_VALUE=hello
- The child node can reuse the parent node variable: MY_COMPLETE_VALUE=$(MY_VALUE) world
- The final value of MY_COMPLETE_VALUE is "hello world".

For example:

- The parent node defines: MY_VALUE=foo
- The child node can reuse the parent node value and redefine the variable: MY_VALUE=$(MY_VALUE) bar
- The final value of MY_VALUE will be "foo bar".

As a result, the System environment variables or Genero Studio Configuration variables can be reused in User Variable definitions within Project Manager.
Configure the display clients

You can configure a variety of display clients in Genero Studio.

Configure for the Desktop client

GDC configurations contain information about the available Genero Desktop Clients (GDC).

You can create multiple GDC Configurations, each with a different name.

Note: For information on installing the Genero Desktop Client outside of Genero Studio, see the Install and License your Genero Products manual.

Select which GDC to use

In the Display section of the Genero Configuration Management dialog, select the Use Display Client radio button. In the combobox provided, select the GDC configuration to use. To add a new GDC configuration, click the Edit icon next to the Display Client configuration combobox. The Display Client management dialog opens.

Create or edit a GDC configuration

To add a new GDC configuration, click the Edit icon next to the Display Client configuration combobox. The Display Client management dialog opens. Select GDC from the Client type combobox.

Figure 96: Display Client Management dialog

Select a configuration from the list to modify its settings. Use the integrated Toolbar to add, duplicate, remove, or edit a configuration. Once a name is added, enter its settings.

The following fields apply to a GDC configuration:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Hostname or IP address where the GDC client executes. In local host mode, when the host field is empty, the...</td>
</tr>
</tbody>
</table>
client IP address will be automatically detected in order that GDC will always be accessible from fglrun. You can change this manually.

**Port**
- Port from which GDC client is to be launched.

**Use autostart script**
- Check this option to have Genero Studio automatically start the GDC.
- Uncheck this option if you want to manually start the GDC.

**Script path**
- Path to where GDC client is installed.

**Script options**
- Command line options for the execution of Genero Desktop Client (GDC). For a full list of valid command line options, see the *Genero Desktop Client User Guide*.

### Related concepts
- [Configure for Mobile clients](#) on page 177
- Android™ and iOS configurations contain information about the available Genero Mobile clients (GMA and GMI).

### Configure for the Web client
To display web applications or to run web services, you configure for both the Genero Browser Client (GBC) and for the Genero Application Server (GAS).

You can create multiple GAS Configurations, each with a different name.

**Note:** If you did not initially install the complete Genero Studio package, you must install and set up the Genero Application Server (GAS) outside of Genero Studio. See the *Install and License your Genero Products* manual.

When you select a GAS configuration, Genero Studio dynamically generates an application configuration file. This file enables you to run, debug, or preview an application or file.

During development, the GAS standalone dispatcher: [httpdispatch](#) on page 202 can be used instead of a web server.

For the Genero Browser Client, specify the browser to use in *General Preferences* on page 128 or in the Configuration Management dialog.

### Configure for the Genero Browser Client
In the Display section, select the **Use Application Server** radio button. Four fields are enabled for configuration of the Web client:

**Use Application Server**
- In this combobox, select which GAS configuration to use. To add a new GAS configuration, click the Edit icon. The *Genero Application Server Management* window opens.

**Application type**
- Select **Universal Agent** to use the Genero Browser Client.
- Select **Web Application** to use GAS 2.50 GWC-HTML5 (deprecated).

**Theme**
- Select a theme to use.

**Browser**
- Specify a specific browser to use. When left blank, the default browser is used.

**Options**
- Specify options to use when starting the browser.
- For example, to start a Chrome browser in Incognito mode, enter "--incognito" into this field.
Configure the Genero Application Server

In the Genero Application Server Management dialog, existing GAS configurations are listed in the left-hand panel. Select a configuration from the list to modify its settings. Use the integrated toolbar to add, duplicate, rename, or remove a configuration. Once a name is added, enter its settings.

**Figure 97: The Genero Application Server management dialog**

In the **GAS Connection Details** section:

**Host**

Specify the IP address or the network name of the server you are connecting to when running an application with the GAS.

To connect to a GAS standalone dispatcher using a direct connection, specify the IP address or the network name.
of the Application Server (the one where the compiler resides). If the GAS standalone dispatcher is local, you can leave this field blank.

To connect through a web server, specify the IP address or the network name of the web server.

**Port**

Specify either the port of the GAS (for a direct connection to the GAS standalone dispatcher) or of the web server.

**Important:** Changing the port number in the GAS Connection Details section does **not** automatically change the port number used when the GAS standalone dispatcher is launched!

If you are launching the GAS standalone dispatcher from Genero Studio, you must either modify the GAS configuration file to have the GAS listen at the same port, or you can add a startup script option to launch the standalone dispatcher at the proper port.

If you modify the GAS configuration file, the change will impact all users of the GAS. By default, the GAS configuration file is named `as.xcf`, and the port number where the GAS listens for connections is the sum of the values of `TCP_BASE_PORT` and `TCP_PORT_OFFSET`. For more information about specifying the listening port for the GAS, see the Genero Application Server User Guide.

To launch the standalone GAS with a port number that differs from that in the GAS configuration file, add this option in the **Start script options** field, where `XX` is the port offset from 6300:

```
-E res.ic.port.offset=XX
```

For example, if you set the port value to "6399", you could start the httpdispatcher with the offset value of "99".

**Figure 98: Specify a port offset**

**Connector**

Leave blank if you are using a direct connection to a GAS standalone dispatcher. This is common in development environments.
For a Web server, enter the alias configured for your dispatcher in your GAS installation (often, gas). See the configuration topics in the *Genero Application Server User Guide* for more information on configuring your dispatcher.

**Secure protocol**

The URL used when launching an application is impacted by this selection.

When selected, the protocol is "https".

When not selected, the protocol is "http".

In the **GAS Configuration Details** section:

**Installation directory**

Directory where the GAS is installed locally.

**Use start script**

The check box specifies whether to use the startup script to launch the GAS. When checked, the field specifies which start script to be launched when this configuration is used. The GAS standalone dispatcher: `httpdispatch` on page 202 can be specified here to launch `httpdispatch`. `httpdispatch` can be used as the GAS script. In versions prior to 2.50, set the environment by specifying `-p <GASDIR>` in the Start script options field. Alternatively, call a script that calls `. ./envas` before calling `httpdispatch`, such as:

```
./envas
httpdispatch
```

For GAS versions prior to 2.30, set the path for `gasd` instead of for `httpdispatch`.

**Note:** The start script is only used when no GAS is found on the host at the port specified. The start script is not called even if the GAS to be launched from start script is different from the one already launched.

**Start script options**

Specify any options to be used with the start script. See GAS standalone dispatcher: `httpdispatch` on page 202 for `httpdispatch` options. For more information on script parameters, see the *Genero Application Server User Guide*.

In the **Application Details** section:

**Application group**

The name of the Application Group defined in the GAS `as.xcf` file. Enter the value `_default` for the group to use the default configuration file, unless you have an explicit group to specify.

**Application directory**

The directory of the Application Group, the same path as the one referred to in the `as.xcf` file. If using the default application, the corresponding directory (usually GAS installation directory/app) has to be entered in the Group Directory field.

**Parent application**

The identifier of the parent GWC application; if no identifier is given, the default GWC application will be used.
### Services group
The name of the Web Services Group defined in the GAS `as.xcf` file. Enter the value `_default` for the group to use the default configuration file, unless you have an explicit group to specify.

### Services directory
The directory of the Web Services Group, the same path as the one referred to in the `as.xcf` file. If using the default application, the corresponding directory (usually `GAS installation directory/app`) has to be entered in the Group Directory field.

### Parent service
The identifier of the parent Web Services application; if no identifier is given, the default Web Service application will be used.

### URL example
The URL used when launching application from Project Manager is impacted by the Secure protocol option and the Application Type selection:

```
<protocol>://<host>:<port>/<connector>/<application_type>/r/<application_name>
```

- **protocol** is either "http" or "https", depending on the value of the Secure protocol option.
- **application_type** is either "ua" or "wa", depending on the value of the Application type option.
- **application_name** depends on the executed application in project manager.
- Other values depend on the GAS configuration parameters.

### Example GAS configuration
- **Host**: `<Automatic>`
- **Port**: 6394
- **Application group**: `mygroup`
- **Group directory**: `c:\work`
- **Parent application**: `defaultgwc`

In this configuration, GAS applications will be run using the GAS located on the current computer at port "6394". The group "mygroup" in the `as.xcf` file points to the directory "c:\work"; the corresponding fields in the GWC configuration dialog are filled with the same information.

**Note:** Users on Windows® 64-bit machines who are using a network proxy: The browser cannot open 127.0.0.1 or localhost unless you modify your Advanced Network settings to avoid going through the proxy for these addresses.

See the *Genero Application Server User Guide* for information on setting up GWC and GAS and adding applications.

### Related concepts
- **GAS standalone dispatcher: httpdispatch** on page 202
- `httpdispatch` is the standalone dispatcher that provides the Genero Application Server for development. No web server is needed.

### Related tasks
- **Application configuration files and Project Manager** on page 411
- Genero Studio creates a default application configuration file to display applications to the Web client. You can update the default configuration with a custom application configuration file.
An application or Web service configuration file provides details required by the Genero Application Server to launch an application or Web service. If you created your project using the Desktop Project (.4pw) or BAM Desktop Project options, you already have this file. However, if this is not the case, you can create this file in Genero Studio.

**Configure for a Web client customization**

You can set up your environment to display your application using a specific Genero Browser Client customization.

Before setting up your environment for a specific Genero Browser Client (GBC) customization, you should first ensure that you can display your application using the Genero Browser Client. See Configure for the Web client on page 171.

Once you successfully display your application in a browser using the Genero Browser Client, you have several options to have your application launch from Genero Studio and use a specific GBC customization.

**Deploy the customization as the default for the GAS**

You can deploy your customization to the Genero Application Server. Refer to the *Genero Browser Client User Guide* for information about deploying a customization project to the Genero Application Server.

Once deployed, use the `gasadmin` tool or the GAS deployment portal to set your desired customization as the default customization to use. As this change is registered in the GAS itself, it will affect all applications connecting to and using the GAS to deliver to the Genero Browser Client.

**Specify within GST**

To specify a specific configuration for all applications launched within Genero Studio, do two things:

First, in the *Genero Application Server Management* dialog, add an option to the *Start script options* field to specify the customization parent folder in the GBC project directory.

```
-E res.path.gbc.user=<path_to_customization_parent_folder>
```

For example, if the GBC project directory is `C:/gbc-project`, then you would set the resource to:

```
-E res.path.gbc.user="C:\gbc-project\dist\customization"
```

**Tip:** This is not the customization project directory. There can be multiple customization projects within the customization folder. This is the parent directory, which holds the compiled customization projects.

Second, create an empty file in the customization parent folder named "._default" (with no extension). Inside this file, specify the name of the customization to use. For example, to use a customization named red, the file would contain the following:

```
red
```

For more information on configuring an application to use a GBC customization, see the *Genero Browser Client User Guide*. For information on application configuration files, see the *Genero Application Server User Guide*.

**Specify a customization for a specific project**

To specify a specific configuration for an application within a single Genero Studio project, do two things:

First, in the *Genero Application Server Management* dialog, add an option to the *Start script options* field to specify the customization parent folder in the GBC project directory.

```
-E res.path.gbc.user=<path_to_customization_parent_folder>
```

For example, if the GBC project directory is `C:/gbc-project`, then you would set the resource to:

```
-E res.path.gbc.user="C:\gbc-project\dist\customization"
```
Tip: This is not the customization project directory. There can be multiple customization projects within the customization folder. This is the parent directory, which holds the compiled customization projects.

Second, create an external application configuration file to specify which customization project to use, and save this file within the project itself. For example, to apply a customization named "red" to the Hello World demo application, you would:

1. Open the HelloWorld.4pw project.
2. Create a new application configuration file named HelloWorld.xcf. The file name should match the name of the application. In this file, set the <GBC> element to the name of the customization project to use:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<APPLICATION Parent="defaultgwc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.4js.com/ns/gas/3.10/cfextwa.xsd">
    <UA_OUTPUT>
        <PUBLIC_IMAGEPATH>$(res.public.resources)</PUBLIC_IMAGEPATH>
        <GBC>red</GBC>
    </UA_OUTPUT>
</APPLICATION>
```

3. Save the file in your project, and in the same directory as the compiled executable.

For more information on configuring an application to use a GBC customization, see the Genero Browser Client User Guide. For information on application configuration files, see the Genero Application Server User Guide.

Configure for Mobile clients

Android™ and iOS configurations contain information about the available Genero Mobile clients (GMA and GMI).

In the Display section, select the Use Display Client radio button.

To set a mobile configuration, select the desired mobile option from the Client type combobox.

To open the Display Client Management dialog, click on the Edit icon.
Figure 99: Display Client Management dialog

Select a display client from the list to modify its settings. Use the integrated Toolbar to add, duplicate, remove, or edit a configuration. Once a name is added, enter its settings.

Display Client Configuration Details

- **Client type**: Select the client type for the configuration.
- **Host**: Hostname or IP address where the mobile client executes.
- **Port**: Port from which the client is to be launched.
- **Debug port**: Port for displaying debug information.
- **Log fetching delay**: The delay in milliseconds before a log is fetched.
- **Use autostart script**: If checked, autostart script will be executed if client is not running.
- **Script path**: Path of script to execute when Use autostart script is checked. For GMI and GMA clients, an implicit script is used.
- **Script options**: The options passed as parameters to the autostart script. Not used with GMI and GMA.
Device Information

Settings for identifying display devices so that multiple display devices can be connected at the same time.

**Device ID**

The identifier of device (iOS or Android™). Use the List Devices option in the Tools > Android Tools or Tools > iOS Tools menu to get list of device IDs.

**Tip:** If you are using iOS and Xcode, you can use a substring from the string displayed by the List Devices option. For example, you can use the label of the device as displayed in the General settings.

The Start Display Client button starts the display client.

**Related concepts**

Configure for the Desktop client on page 170
GDC configurations contain information about the available Genero Desktop Clients (GDC).

Configure for a TUI client

Set your Genero Configuration to use a TUI client for applications that need to run in a terminal (text mode).

When the display is set to text mode, the FGLGUI environment variable is set to 0, indicating to the DVM that no display client should be used and the application should run in a terminal. There are limitations:

- Text mode only works for a local configuration; an error displays in the Output view if used with a remote configuration.
- Form files must be text-based .per files with a SCREEN section instead of a LAYOUT section. For details about creating a .per form file, see the Genero Business Development Language User Guide.
- At this time, the "debug" action is not available. To debug a program run in text mode, use the attach to process feature. See Debug a running local program on page 590.

1. Select Tools > Genero Configurations.
2. In the Display section of the Genero Configuration Management dialog, select the Text Mode radio button.
3. Click OK.

   Genero Studio is now configured to use text mode.

Check the Output view to check for any errors.

**Setting up a remote environment**

These topics assist you in setting up a remote Genero Studio environment.

- Remote environment software requirements on page 179
- Add a remote host on page 180
- Define mount points to shared drives on page 187
- Define remote Genero installations, GAS configurations, and environment sets on page 188

**Remote environment software requirements**

In a remote environment, installation is done on the remote server and accessed by developers from their client machines.

**Important:** The versions of Genero Studio and Genero Studio Server must match.

**On the server(s)**

- Installed Genero Studio Server (plus server-side tools provided as part of the Genero Suite Server package).
• Installed relational database (Informix®, Oracle®, etc.) and its corresponding client (CSDK, Oracle® Client, etc.); or Database Client on this server and the corresponding Database on a different remote server.
• SSH server (if using the recommended SSH connections).

On the client
• Genero Studio
• Installed Genero Desktop Client

Each database software has its own client character set configuration. You must properly configure the database client locale in order to send/receive data to the database server, according to the locale used by your application. See Localization: Database Client Settings in the Genero Business Development Language User Guide.

Related concepts
Share projects / source code management on page 188
The Genero Source Code Management (SCM) module enables collaborative sharing and maintaining of the files in Genero projects.

Add a remote host
To connect a Genero Studio client to a remote Genero Studio Server, add the remote server to the Hosts defined in Genero Studio Configurations and define a remote configuration for that server.

1. Start Genero Studio on the client.
2. Select Tools > Server connections or select the Server Connections button in the lower right corner.
3. Select the button in the integrated Toolbar to Add a Genero Studio server.
Figure 100: Add a Genero Studio server

4. Enter the Host Information.
Figure 101: Host Information

**Host**
Hostname or IP address where Genero Studio Server is running.

**Port**
Enter the port number you have configured SSH to listen on. The default SSH port number is 22. If you are using TCP (deprecated), enter the port number used when Genero Studio Server was started. The default port number is 5321.

**Client IP address**
IP address of the client as viewed by the server. Displays the value of the IP address to be used by the GDC settings when *Automatic* is set.

**Connection type**
SSH (default) or TCP (deprecated).

**Installation directory**
The Genero Studio installation directory on the remote machine; enter the complete path, which will always end with /gst.
| Configuring Genero Studio | 183 |

User name
Valid user name. See SSH advanced security on page 183.

User password
Valid password.

5. Select Connect to validate the host configuration.

6. Once a valid connection has been made, existing Genero installations, GAS configurations and Environment sets can be imported from the server. Select the Import button. See Import Configuration dialog on page 200.

7. Define mount points to shared drives on page 187.

8. Select OK. Your new remote host will now be listed in the host list in the lower right corner of Genero Studio.

Figure 102: Host list

SSH
Each user with an SSH connection works with their own server process, as one instance of Genero Studio server is created for each SSH connection. If you choose this connection type, an SSH daemon must be running on the host. There is no need to launch Genero Studio Server on the host; the SSH connection will automatically launch it. On Windows™, a third party SSH server must be installed.

SSH advanced security

- Configuring Public Key Authentication on Windows™
- Configuring Public Key Authentication on Linux™

Configure public key authentication on Windows™

Genero Studio can support public key authentication on Windows™ if you use Pageant. You have to install PuTTY on your client machine to use it. Pageant is a utility that stores unencrypted private keys in memory and provides digital signatures when needed.

You configure it by giving the path of the encrypted private key file (See PuTTY documentation for more information). You enter the passphrase just once when Pageant is launched; then you will not have to bother with it again, until you launch a new Windows™ session.

Genero Studio uses plink to initiate an ssh connection. If Pageant is launched, plink automatically uses Pageant to get a signature. When using public key authentication in Genero Studio, don't fill in the password field in the SSH connection dialog, as it not used.

Pageant

First, you have to generate a pair of private/public keys. If you already have a SSH/OpenSSH key pair, see the How to use openSSH key with Pageant section.

- Launch PuTTYgen, and verify that the SSH-2 RSA radio button is selected.
- Select the number of bits you want as the size for the key and click the Generate button. During generation, move cursor into the blank area to randomize the keys.
- Enter a comment to identify the key pair more easily.
- Enter a passphrase. A passphrase is used to decrypt the private key (Private key is encrypted on the disc for more security). Don't forget it, you will have to enter it each time you begin a new Windows™ session.
- Once you have entered the passphrase, you can save the private and public keys on the disc.
Next, go to your home directory (/home/username) on your host machine, and edit the .ssh/authorized_keys file with your preferred text editor. If the file (or .ssh directory) doesn't exist, create it.

Copy your generated public key in the authorized_keys file. The key must be in exactly the same format as displayed in PuTTYgen (ssh-rsa ****...****== keyname) and must occupy only one line. If several lines were already present, enter a line return before writing the key.

Now launch Pageant. It will appear in the systray as a desktop computer wearing a hat. Right click on it and select Add key.
Figure 104: Pageant pop-up menu.

- An open file dialog lets you select a ppk (Private key) file. After selecting it, you will be asked to enter the corresponding passphrase.
- You can verify that the key has been successfully added by selecting "View keys" in the Pageant icon's context menu. The key signature is displayed instead of the private key, but you can recognize it by its name.

You should now be able to initiate an ssh connection with Genero Studio without entering a password.

How to use openSSH keys with Pageant

The ssh-keygen generated private key files cannot be directly used by Pageant; you have to convert them into .ppk files. To do so, open PuTTYgen and select Conversions>>Import key menu item; select your OpenSSH format private key.

Configure public key authentication on Linux

Genero Studio uses plink ssh client to initiate an ssh connection; plink is traditionally used with the Pageant authentication agent. Since Pageant doesn't exist on Linux, we will use ssh-agent, which is released with OpenSSH clients.

ssh-agent stores the private key unencrypted in memory, and provides the private key signature when asked. You configure it by giving it the path of the encrypted private key file (using the ssh-add command). When ssh-agent is launched, plink will automatically ask ssh-agent to get a signature to send to ssh server.

First, you have to generate a private\public key pair. If you already have a PuTTY generated key pair, see the How to use PuTTY keys with ssh-agent section; otherwise, we will generate it with ssh-keygen.

- Type ssh-keygen -t rsa in a terminal to generate a key pair using rsa encoding.
- You will be prompted for the file path where you want to save your private key. If you press enter without giving a path, the key will be saved as ~/.ssh/id_rsa.
- Next, enter the pass phrase (needed to decrypt the private key). Don't forget it as you will have to enter it again at each logon session. The public key file will be saved with the same path as the private key file, but with the .pub extension.
• Go to your home directory (\home/username) on your host machine, and edit the .ssh/authorized_keys file with your preferred text editor. If the file (or .ssh directory) doesn't exist, create it.

• Copy your generated public key in authorized_keys file. The key must be exactly in the same format as displayed in PuTTYgen (ssh-rsa ****...****== keyname) and must occupy just one line. If several lines were already present, enter a line return before writing the key.

Next, configure ssh-agent to automatically provide signatures from the private key.

• ssh-agent must be started at each logon session. Execute it if it isn't already running.

• Next, execute ssh-add with the private key file path as the parameter (For example ssh-add ~/.ssh/id_rsa). This will tell ssh-agent to test your private key each time you start an ssh session.

• Verify that the key has been successfully added by typing ssh-add -l. It should display you your private key fingerprint.

You can now use public key authentication with Genero Studio. For that, create an SSH connection without entering a password:

![Figure 105: Host Configuration](image)

How to use PuTTY keys with the ssh-agent

If PuTTY is installed on Windows™, start PuTTYgen, load your private key (ppk file), and select the conversions >> Export OpenSSH key menu item. It will ask you where you want to save the file.

If PuTTY is installed on Linux™, execute puttygen input_key.ppk -O private -o output_key.

SSH troubleshooting
This topic covers common SSH issues and how to resolve them.

Table 33: Errors when trying to connect to a server using SSH

<table>
<thead>
<tr>
<th>Type</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trying to connect to a server using SSH</td>
<td>\installation directory: No such file or directory.</td>
<td>The path given in installation directory field is incorrect.</td>
</tr>
<tr>
<td>Trying to connect to a server using SSH</td>
<td>Access denied.</td>
<td>Either the user name or password is incorrect.</td>
</tr>
<tr>
<td>Unable to open a connection</td>
<td>Host does not exist.</td>
<td>The host either does not exist or is not reachable.</td>
</tr>
<tr>
<td>Type</td>
<td>Issue</td>
<td>Resolution</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>--------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Unable to open a connection</td>
<td>Network error: Connection refused.</td>
<td>Verify that the port number corresponds to the SSH server's one.</td>
</tr>
<tr>
<td>Unable to open a connection</td>
<td>Network error: Connection reset by peer.</td>
<td>This happens when the networked machines lose the connection between them. Try to reconnect.</td>
</tr>
</tbody>
</table>

**TCP (deprecated): How to launch Genero Studio Server**

If you use a TCP connection, Genero Studio Server must be launched before use. (If you use an SSH connection, Genero Studio Server is launched automatically.)

This program has no user interface; it must be run from a console or terminal. On the server machine, from within `GSTDIR/bin`, execute the command for your operating system. The default port value is **5321**.

- On Windows™:
  
  `gsserver.exe`

- On GNU/Linux:
  
  `gsserver`

If you need it to listen to another port, you have to use the command-line argument **--port** each time you start Genero Studio Server.

- On Windows™:
  
  `gsserver.exe --port 1234`

- On GNU/Linux:
  
  `gsserver --port 1234`

For further information, start Genero Studio Server with the **--help** option, and see the installation notes in the release subdirectory of the Genero Studio software package.

**Define mount points to shared drives**

Shared drives allow you to compile, run, and debug files that are on the remote server.

This task assumes you have added a remote host.

Export the file system on your server using Samba / NFS / Windows® Share. You must share the drive using a standard tool, such as NFS (Network File System) for Linux™ networks. Samba, or a similar tool, can be used to allow Microsoft™ networks to share files stored on a Linux™ server.

All program files accessed remotely must reside on a shared drive on a server accessible to the client's local network.

1. Open the Genero Studio Server dialog and select a remote host from the list.
2. Select the Add a mount point … button from the Mount Point Table integrated Toolbar. A dialog lists the mount points available for this server.
3. Select a mount point and enter the remote path.

For a Windows® Genero Studio client:

**Mounted point**

The mounted drive, the network drive that has been mapped to a letter drive on your local machine.
Remote path

The related remote path. For example, if your local computer has a mounted drive G: that is mapped to /usr1/public on the server zebra, the remote path for drive G: is /usr1/public.

For a Linux™ Genero Studio client:

Mounted point

The location of the file system where the device is attached.

Remote path

The related remote path. For example, if your local computer has a mount point /mnt/work that is mapped to /usr2/public on the server zebra, the remote path for /mnt/work is /usr2/public.

4. Select Apply.
   The character set will be specified in the environment of the remote FGL installation (set the LANG variable in the remote configuration variables). On the client machines, set Language support (text encoding) on page 189, specifying the same character set.

5. Define remote Genero installations, GAS configurations, and environment sets on page 188.

Define remote Genero installations, GAS configurations, and environment sets

Genero installations, GAS configurations, and environment sets can be defined for the remote host.

This task assumes you have added a remote host.

1. Select the remote host from the list in the lower right corner of Genero Studio.

2. Select the configuration button to display the Genero Configuration Management dialog for the selected server.

3. Add new configurations as needed to add Compiler / Runtime configuration (Genero Installations) on page 200, Environment sets on page 164, and Configure for the Web client on page 171.

4. Save the changes.

Share projects / source code management

The Genero Source Code Management (SCM) module enables collaborative sharing and maintaining of the files in Genero projects.

You can share your Genero projects using a Version Control System (VCS) such as Apache's Subversion:

- A network drive/directory must be set up on the remote server for the storage of the project and source files.
- All paths in the 4pw files must be relative paths; only the path of the project(s) can be absolute.
- The network drive/directory must be mapped on each client machine.
- You will check out a copy of the project file and source files from the repository to your mapped network drive. After making changes, commit your files back to the repository.

Genero Studio for Windows® includes Apache's Subversion client. Genero Studio for GNU/Linux® relies on Subversion 1.6.2 or later, which must have been installed on the system. For more information about Apache's Subversion, see: http://subversion.apache.org/

To specify the location of your SVN client, select Tools > Preferences > Source Code Management > Subversion from the main menu.

Related concepts

Source Code Management (SCM) on page 624
The Genero Source Code Management (SCM) module enables collaborative sharing and maintaining of the files in Genero projects.

### Access a database

To execute an application that accesses a relational database, Genero Studio uses the Genero runtime system and the specific database client software, which must be properly configured.

Entries in a FGLPROFILE configuration file can be used to connect to the database. See the specific page for the supported database vendor's software in the Open Database Interface section of the Genero BDL documentation.

Genero Studio database meta-schema files allow you to use the database tables and columns that are listed in the schema to define variables and form fields in your application code and form definitions.

Default environment sets are provided for each database vendor. Values for the environment variables, as well as additional variables, can be added in the Genero Configuration Management dialog on page 197 as part of the configuration process.

**Related concepts**

- Setting up a remote environment on page 179
- These topics assist you in setting up a remote Genero Studio environment.

- Setting up a local environment on page 163
- These topics assist you in setting up a local Genero Studio environment.

### Language support (text encoding)

Text encoding specifies the character set to be used by Genero Studio. You can change the default encoding to work in your preferred language.

To support *internationalization*, characters typed at the keyboard are intercepted and changed automatically, based on the specified text encoding method.

The character set is specified in Tools > Preferences > General Preferences. The default text encoding method is the one found on your system. To use encodings for a different language, select a different text encoding choice from the list. For example, to use Arabic, set the Text Encoding to ISO-8859-6. In addition, text encoding plugins can be configured, which are loaded at startup to provide new encoding support.

Default encodings are much faster than plug-ins. You are strongly advised to use the provided encodings unless there are no other solutions.

The encoding must be correctly set before opening documents, as the selected encoding is used during file operations of text documents (load, save, parsing, highlighting, and so on).

**Other considerations:**

- If you change the text encoding in Genero Studio, the corresponding LANG environment variable must be set.
- If you are using Genero Studio in a remote configuration, your encoding method must match the encoding on the Genero Studio server.
- Mobile devices must use UTF-8.
- If you are editing text using a third-party tool, you must ensure that the tool uses the same encoding method.
- The database client locale (character set) and application locale (character set) settings must match.
- Genero BDL and Genero clients have settings that must also be changed when you want to use encodings and character sets for a language that is different from that specified in the default text encoding for your system. See the topic on Localization in the Genero Business Development Language User Guide for additional information about the settings for Genero Business Development Language. See the topic on Localization in the Genero Desktop Client User Guide for additional information about settings for Genero clients.

**Topics:**
Configure Genero Studio to use a different character set

This example explains how to set up Genero Studio client and servers to use the Polish language with ISO 8859-2 encoding.

In this example, the Genero Studio client runs on a Windows™ 7 Ultimate 64-bit platform using English with the CP1252 character set (Western European languages). The Genero Studio server runs on enterprise Linux™ using English with the UTF-8 character set.

1. Configure the preferred encoding for the Genero Studio client. Select Tools > Preferences > General and locate the preferred encoding (ISO-8859-2 in this example) in the list of Text Encodings.
   
   If the preferred encoding is not available in the list, add a text encoding plugin to incorporate the preferred character set.

   ![Figure 106: Text File Encoding in General Preferences](image)

2. Check to see if the preferred encoding is supported on the Linux™ server (pl_PL.iso88592 for the example):

   ```bash
   $ locale -a | grep 88592
   
   If the preferred locale isn't present, install it per your system documentation. For example:
   
   ```bash
   $ localedef pl_PL.iso88592 -i pl_PL -f ISO-8859-2
   ```

4. Configure the selected language for the Genero Studio Server environment.

5. Test the new configuration.

Add a text encoding plugin

Use this procedure to add support for a new encoding in a Genero Studio client-server configuration.

Before you begin, the preferred encoding must be installed on the Linux™ system.

The example shows how to configure a new encoding, using IBM852 as an example. The Genero Studio client runs on a Windows™ 7 Ultimate 64-bit platform using English with the CP1252 character set (Western European languages). The Genero Studio server runs on enterprise Linux™ using English with the UTF-8 character set.

1. Locate the IBM852 charmap on the internet or Linux™ system.

   ![Figure 108: Unix ls command showing the compressed IBM852 character map](image)

2. Copy the uncompressed charmap to the GSTDIR/conf/charmaps directory on both client and server. Create the charmaps directory if it does not already exist.
3. Create a new entry for the encoding in `GSTDIR/conf/encodingMap.xml` on both client and server.

4. Open Genero Studio on the client, and select **Tools > Preferences > General**. Check to see if the new plugin displays in the list of encodings.

   If the plugin is not present in the Text Encoding list, check the following:
   
   • Did you copy the charmap to the Genero Studio charmaps directory on both client and server?
   • Is the charmap a valid POSIX2 charmap?
Figure 111: New text encoding plugin displayed in Text Encoding list.

Character mapping table (encodingMap.xml)

The encodingMap.xml file maps character set aliases, fallback character set names to try if the alias is unsupported, the name of the character set used by Genero Studio, and the corresponding language definition for the LANG variable for Genero on Unix.

A default list of text encodings is provided in Genero Studio. The mappings between each character set alias and its corresponding attributes are defined in the encodingMap.xml file in the GSTDIR/conf directory.

For some of the encodings there is only a partial match between Unix and Windows™ platforms.

This file contains a table used to map each encoding name (alias) to:

- A list of fallback character set names to try if the alias is not supported.
- The name of the character set used for encoding/decoding.
- An implementation name in Genero Studio (impl attribute)
- a LANG qualified country for Unix/Linux (Country attribute)

```xml
<Alias name="IBM852" fallback="852,ibm852" impl="IBM852" unixCountry="pl_PL"/>
```

This means that:

- IBM852 is the name Genero Studio should use for this text encoding.
- 852 and ibm852 are character set names to try if IBM852 is not supported.
- IBM852 is the name of the character set used for encoding/decoding.
- Under Unix or Linux™ the LANG language definition is pl_PL.

The impl attribute is defined within the POSIX2 charmap file after the code_set_name. It appears in the Genero Studio preferences combobox: <code_set_name> IBM852.

The name attribute is the name Genero Studio should use for this text encoding. By default it is the character set name (code_set_name); when an alias to an existing encoding is needed, the name attribute should contain the alias name.

```xml
<Alias name="CP852" fallback="IBM852,852" impl="IBM852" unixCountry="pl_PL"/>
```
Configure keyboard and language on a Windows® client

Follow this procedure to change the keyboard and default input language on a Windows® client.

The details provided here are for Windows® 10. Refer to the documentation for your version of Windows™ for assistance.

1. Open the Settings dialog and select Time & Language.
2. Select the Region and language tab.
3. In the Languages section, select the language and click Set as default.

Note: If the preferred language isn’t listed, click Add a language and select from the list.

Figure 112: Time & Language Dialog
Configure LANG on a Genero Studio Server

Follow this procedure to configure the locale on a Genero Studio server (Linux™).

This task assumes you have added a remote host. In addition, the locale to be specified for the LANG environment variable must be installed on the Unix or Linux system.

Text encoding specifies the character set, and when compiling files, the LANG environment variable is used by Genero internal processes to determine which encoding the files use.

Note: Setting the LANG environment variable in an Environment Set overrides system defaults or user level (.profile) settings.

1. Launch the Genero Studio server (skip this step if using an ssh connection - it is automatic).
2. On the Genero Studio client, select the remote host from the list in the lower right corner.
3. On the Genero Studio client, navigate to Tools > Genero Configurations.
4. Select a configuration from the list in the far left pane then select the plus to add a new environment set.
5. Type in a meaningful name for the new environment set and press OK.

Figure 113: Add Remote Environment Set
6. Select the 
+ above the Environment Variables list to add a variable named LANG with the desired locale (pl_PL.iso88592 in this example).

With the LANG environment variable, you define the language, the territory (country) and the character set (codeset) to be used. The format is:

<language>_<country>.<codeset>

Figure 114: Configuring the LANG environment variable

7. Add the new environment set to other configurations for the remote host as needed.

Test text encoding configuration

Follow this procedure to test language settings in a Genero Studio client/server environment.

You can create a new project and compile it to test a new text encoding configuration or perform a simple test with the provided HelloWorld sample project as described below.
1. Select **Tutorials & Samples** in the Genero Studio Welcome Page and open the **HelloWorld** sample.
2. Double-click the **HelloForm.4fd** file to open it and scroll through the form properties until you locate the text property.
3. Use a translation tool such as Google Translate to create a phrase in the test language and replace the existing form text with the translated phrase.
4. Save the changes to the form, re-build and execute the **HelloWorld** application.
   The language settings are correct if characters from the extended character set display properly. Question marks displayed in place of extended characters indicate a problem with the configuration.

**Configuration reference**

Reference topics for configuring Genero Studio.

- Genero Configuration Management dialog on page 197
- Import Configuration dialog on page 200
- Import Preferences dialog on page 200
- GAS standalone dispatcher: httpdispatch on page 202

**Genero Configuration Management dialog**

Genero Studio configurations provide the information needed to run on your local or remote hosts.

A default configuration is provided, based on the software installed in the default directories. You can have multiple Genero configurations, distinguished by the name that you have assigned.
### Figure 115: Genero Studio Configuration

1. **Add a Configuration**
   
   The names of the available Genero configurations. Select the configuration name to view its settings. Use the integrated Toolbar to add the name of a new configuration. Once added, you can enter its settings.

   A check mark in the **Status** column indicates there are no missing settings.

   The integrated Toolbar also includes options to duplicate, rename, and delete configurations. Use the up or down arrow to modify the placement of the selected configuration in the list. The order of the configurations is organizational only and has no other effect.

2. **Import a Configuration**
   
   Displays the Import Configuration dialog on page 200 to select an installation from which to import configurations.

3. **Genero Installation**
   
   This field identifies which Genero Installation will be used to compile or run the application. A Genero Installation specifies the `FGLDIR` and the runner name. The names of the Genero installations that are available for the selected host are displayed in the list. Select the desired installation for your configuration. To view or edit the settings for the selected Genero Installation name, or to add a new Genero Installation, select the **Edit** button (See Genero Installations).

4. **Environment Sets**

<table>
<thead>
<tr>
<th>Configuration Name</th>
<th>Status</th>
<th>Compiler / Runtime</th>
<th>Environment Sets</th>
<th>Environment Variables</th>
<th>Display</th>
<th>Use Display Client</th>
<th>Use Application Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10.03 Desktop</td>
<td>✔</td>
<td>FGL 3.10.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.10.03 Web</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.10.03 Android</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.10.03 iOS Dev Client</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An environment set is a collection of environment variables to be set prior to compiling or running the application on the specified host. The name of the environment sets available for the selected host are listed. Use the up/down arrows to set the priority of environment sets. A check mark indicates the environment set is active. To view or edit the settings for the selected environment set, or to add a new environment set, select the Edit button. (See Environment Sets). Default environment sets have been created for the databases supported by Genero. Default variables have been entered for each set, and values have been provided where possible. Select your environment set, and enter any missing values or variables.

5. Environment Variables
   Add and modify environment variables within an environment set by selecting the Add or Edit button.

6. Use Text Mode
   Run the application in a terminal instead of using a display client. See Configure for a TUI client on page 179.

7. Use Display Client
   Possible front ends include:
   - Android™ ARM. See Configure for Mobile clients on page 177.
   - Android™ x86. See Configure for Mobile clients on page 177.
   - iOS Development Client. See Configure for Mobile clients on page 177.
   These configurations can be viewed and edited by selecting the Edit button.

8. Use Application Server
   This option configures the display for a Web client.
   The Use Application Server combobox allows you to specify which GAS to use.
   Select between Universal Rendering or Web Application with the Application Type combobox.
   Select the Theme to use. The list of available themes will depend on the Application Type selected. If the theme field is left empty, then default theme will be used.
   Use the Browser field to specify the web browser to use. If left empty, the web browser defined in preferences is used.
   You can specify Options to use when starting the browser by entering the options in the Options field.
   See Configure for the Web client on page 171 for more details.

Confirm or cancel changes
Use these button options to confirm or cancel changes.

**OK**  Save and apply all modifications, then exit.

**Cancel**  Undo all modifications and exit. The last saved values are restored.

**Apply**  Confirms your updates, allowing you to test the new configuration.

Related concepts
Setting up a remote environment on page 179
These topics assist you in setting up a remote Genero Studio environment.

Setting up a local environment on page 163
These topics assist you in setting up a local Genero Studio environment.

**Compiler / Runtime configuration (Genero Installations)**

A Genero Installation contains information about the compiler and runtime version of Genero that will be used by Genero Studio.

Select the name of the Genero Installation in the Host list to view or edit the Genero Installation settings.

- **FGLDIR**
  - The installation directory of the Genero BDL software.

- **FGL runner**
  - The name of the Genero executable. (For example, fglrun, fglrun.exe)

- **FGL options**
  - Options given to the FGL runner when an application is launched.

Select the Check Installation button to test the settings.

Confirm or cancel your changes.

**Import Configuration dialog**

When you install a new version of Genero Studio you can choose to import previous Configuration settings from older installations of Genero Studio.

The Import Configuration dialog can be displayed from the Genero Configuration Management dialog on page 197.

![](image)

**Figure 116: Import Configurations**

**Import Preferences dialog**

When you install a new version of Genero Studio you can choose to import previous Configuration settings from older installations of Genero Studio.

The Import Preferences dialog appears when you install Genero Studio if prior installations are detected.
Figure 117: Import Preferences

Select an installation to import its Preferences settings and update the new installation of Genero Studio.

**Show older versions**

If the version of Genero Studio is too old, it won't be listed in the dialog unless the **Show old versions** (partial import only) checkbox is checked. If an old version is chosen, only part of the Preferences will be imported (for example, Preferences such as menus or Toolbars won't be imported).

**Import options**

Check the boxes for importing the **Configurations**, **Histories**, and/or **Preferences** from the selected installation.

**Related concepts**

*Import Configuration dialog* on page 200
When you install a new version of Genero Studio you can choose to import previous Configuration settings from older installations of Genero Studio.

**GAS standalone dispatcher: httpdispatch**

httpdispatch is the standalone dispatcher that provides the Genero Application Server for development. No web server is needed.

**Important:** Only use httpdispatch in development mode. For deployment, use the dispatcher designed for your web server.

httpdispatch can be launched automatically when you start a Web application from within Genero Studio by specifying it in your GAS configuration. Alternatively, you can manually start httpdispatch by opening a Genero Workplace Window and entering the command information at the command line.

For more information about httpdispatch and the command line options available, see the *Genero Application Server User Guide*.

**Configuration error messages**

A list of configuration error messages. For messages that are not self-explanatory, additional information is provided.

**Table 34: Configuration error messages**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-19001</td>
<td>Unable to send data to a non connected remote task. Internal error.</td>
</tr>
<tr>
<td>GS-19002</td>
<td>Remote server unavailable. Cannot connect to the remote host.</td>
</tr>
<tr>
<td>GS-19004</td>
<td>Remote processing error. Internal error.</td>
</tr>
<tr>
<td>GS-19005</td>
<td>Incompatible server protocol. The protocol versions in the client and the server are not the same.</td>
</tr>
<tr>
<td>GS-19006</td>
<td>A server is listening on the given port, but it isn't a Studio Server. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-19007</td>
<td>Studio Server is listening on the given port, but it doesn't use the same protocol version. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-19008</td>
<td>Disconnection has been forced by the client. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-19009</td>
<td>The server disconnected. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-19010</td>
<td>Internal socket error. Internal error.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-19011</td>
<td>Request aborted.</td>
</tr>
<tr>
<td>GS-19012</td>
<td>Server is not reachable or doesn't listen on given port.</td>
</tr>
<tr>
<td>GS-19013</td>
<td>Connected to %1.</td>
</tr>
<tr>
<td>GS-19100</td>
<td>Host address and port pair must be unique: %1 host configuration has same host (%2) and port (%3)&quot;</td>
</tr>
<tr>
<td>GS-19101</td>
<td>Connecting to '%1'</td>
</tr>
<tr>
<td>GS-19102</td>
<td>Connection failed: cannot connect to the Genero Studio Server</td>
</tr>
<tr>
<td>GS-19103</td>
<td>Connection failed: the Genero Studio Server uses a different protocol version</td>
</tr>
<tr>
<td>GS-19104</td>
<td>Client IP address has been converted from IPv6 to IPv4. Client IP address has been converted from IPv6 to IPv4. Client IP address is invalid. FGL runner may not manage to connect to GUI.</td>
</tr>
<tr>
<td>GS-19105</td>
<td>No mount point defined</td>
</tr>
<tr>
<td>GS-19111</td>
<td>Cannot write temporary file '%1'</td>
</tr>
<tr>
<td>GS-19200</td>
<td>Path cannot be empty</td>
</tr>
<tr>
<td>GS-19201</td>
<td>FGL runner cannot be empty</td>
</tr>
<tr>
<td>GS-19202</td>
<td>Testing '%1'</td>
</tr>
<tr>
<td>GS-19300</td>
<td>&lt;Choose a Genero Install&gt;</td>
</tr>
<tr>
<td>GS-19301</td>
<td>&lt;Choose a Display Client&gt;</td>
</tr>
<tr>
<td>GS-19302</td>
<td>&lt;Choose an Application Server Configuration&gt;</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| GS-19303 | Fetching infos from GAS...  
Informational message.                                                                                                                                                                                   |
| GS-19304 | GAS has refused to give informations. Verify your GAS settings  
Failed to start GAS.  
This message will only be seen for GAS versions >= 2.40. Starting with 2.40, once the GAS version number is retrieved, Genero Studio tries to get the information about the application groups. This message appears when the retrieval of the GAS version number is successful, yet the retrieval of the application group information is not successful.  
To remedy the issue, verify that the GAS is started, and that the host and port values are correct. The GAS should also be configured to allow Genero Studio access to the GAS monitor. By default, the GAS monitor can only be accessed from the localhost, and Genero Studio/gsserver must be on the same machine as the GAS to access it. |
| GS-19400 | Path cannot be empty  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
| GS-19401 | Host IP address cannot be empty  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
| GS-19402 | Display client will not be accessible from other machines  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
| GS-19500 | Application group directory cannot be empty  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
| GS-19501 | Parent application cannot be empty  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
| GS-19502 | Application group cannot be empty  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
| GS-19503 | Start script path cannot be empty  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
| GS-19504 | Connector cannot contain only blanks  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
| GS-19505 | Web service group directory cannot be empty  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
| GS-19506 | Parent web service cannot be empty  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
| GS-19507 | Web service group cannot be empty  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
| GS-19600 | Incorrect environment variable '%1' value  
Error message and resolution should be self-explanatory.                                                                                                                                                  |
### Number | Description
--- | ---
GS-19601 | Put one and only one Genero Mobile for Android package in `$(GMADIR)`
 | Error message and resolution should be self-explanatory.
GS-19602 | Display Client type should be 'iOS' or 'iOS Simulator'
 | Error message and resolution should be self-explanatory.
GS-19603 | Debug server connection error, check if the mobile app is running on the device
 | Error message and resolution should be self-explanatory.
GS-19604 | Starting display client...
 | Informational message.

---

**Microsoft® Windows® Security Blocking**

The Windows® firewall, enabled by default, detects and blocks the Genero Studio Server and the Genero Desktop Client.

From the network point of view, Genero Studio Server and the Genero Desktop Client (used by Genero Studio to display forms) are both considered servers. When you start Genero Desktop Client, or attempt to connect to Genero Studio Server, the firewall detects this and blocks the programs.

A dialog allows you to unblock the program. Follow the instructions provided by the dialog.

If you select Keep Blocking or Ask Me Later, the firewall continues to block the program(s).

If Keep Blocking has been pressed by mistake, you can change this parameter in the Windows® Firewall settings. Ensure you add Genero Studio, Genero Studio Server and Genero Desktop Client to the list of exceptions or allowed apps, and make sure that their listings have a check mark.

---

**Business Application Modeling (BAM)**

Genero Studio Business Application Modeling (BAM) is a graphical design tool to generate the code that extracts data and creates data models in BDL. It automatically generates the logic and source code for a database application to query, add, update and delete rows in database tables. BAM can generate desktop, web, and mobile applications.

- Quick Start: Generate an application on page 206
- Quick Start: Generating a mobile app on page 214
- BAM Concepts on page 223
- BAM Projects on page 232
- Modeling the application on page 232
- Modeling the database on page 276
- Working with forms on page 285
- Adding custom code on page 303
- Modifying the look and feel on page 315
- BAM Reference on page 320
Quick Start: Generate an application

This quick start guides you through generating a basic Genero business application, using the default template, that can be used to add, update, delete, and query rows in a relational database. BAM allows you to visually model your app and generates the code from the design models. You focus on the models, BAM handles the coding.

The application accesses the **officestore** sample database included in the Genero Studio distribution. An FGLPROFILE file is provided in the samples/OfficeStoreRes directory. Entries in the file identify the SQLite officestore database.

Create a BAM project

Use the Simple BAM Desktop Project (.4pw) template to create your managed project.

Before you begin, select **Tools > Genero Configurations** and confirm that the template of your choice is selected in the Environment Sets list. The selected template will be the code generation template used for the project.

**Tip:** Unless you have a known reason to do otherwise, select the latest available template. Starting with dbapp4.0, you must also select between a Desktop and a Mobile version of the template.

1. Select **File > New... > Genero BAM Desktop > Simple BAM Desktop Project (.4pw)** and click OK.

   An unsaved project displays in the Projects view. When you create a structured BAM Desktop Project, nodes for the basic structure of the project are already defined and automatically contain the additional build rules needed for generated programs. The project structure has predefined nodes to contain the files for your project:

   - **Application** - for the program and form files, and any additional source code files
   - **Databases** - for the Genero database schema files (4dbx)
   - **Library** - for any additional resource files

2. Right-click on the Application node and select **Advanced Properties** from the menu. Note that the dependencies for the Databases and Library nodes have already been set, ensuring that any files they contain will be included in your application.

3. In the Advanced Properties dialog, add an FGLPROFILE environment variable to the Environment Variables page.

   a) Select **Environment Variables**.
   b) Under the **User Variables** section, click the green cross icon.
   c) In the Environment Variable dialog, add the FGLPROFILE user variable. Specify a type of "File List", and have the Value point to the location of the fglprofile file provided in the My Genero Files/samples/OfficeStoreRes/ directory.

   ![Figure 118: Edit environment variable dialog](image)

The FGLPROFILE file contains the configuration information to access the sample databases.

4. Click OK to close the Environment Variable dialog.

---

**Note:**

BAM allows you to visually model your app and generates the code from the design models. You focus on the models, BAM handles the coding. The application accesses the **officestore** sample database included in the Genero Studio distribution. An FGLPROFILE file is provided in the samples/OfficeStoreRes directory. Entries in the file identify the SQLite officestore database.

---

**Create a BAM project**

Use the Simple BAM Desktop Project (.4pw) template to create your managed project.

Before you begin, select **Tools > Genero Configurations** and confirm that the template of your choice is selected in the Environment Sets list. The selected template will be the code generation template used for the project.

**Tip:** Unless you have a known reason to do otherwise, select the latest available template. Starting with dbapp4.0, you must also select between a Desktop and a Mobile version of the template.

1. Select **File > New... > Genero BAM Desktop > Simple BAM Desktop Project (.4pw)** and click OK.

   An unsaved project displays in the Projects view. When you create a structured BAM Desktop Project, nodes for the basic structure of the project are already defined and automatically contain the additional build rules needed for generated programs. The project structure has predefined nodes to contain the files for your project:

   - **Application** - for the program and form files, and any additional source code files
   - **Databases** - for the Genero database schema files (4dbx)
   - **Library** - for any additional resource files

2. Right-click on the Application node and select **Advanced Properties** from the menu. Note that the dependencies for the Databases and Library nodes have already been set, ensuring that any files they contain will be included in your application.

3. In the Advanced Properties dialog, add an FGLPROFILE environment variable to the Environment Variables page.

   a) Select **Environment Variables**.
   b) Under the **User Variables** section, click the green cross icon.
   c) In the Environment Variable dialog, add the FGLPROFILE user variable. Specify a type of "File List", and have the Value point to the location of the fglprofile file provided in the My Genero Files/samples/OfficeStoreRes/ directory.

   ![Figure 118: Edit environment variable dialog](image)

   The FGLPROFILE file contains the configuration information to access the sample databases.

   d) Click OK to close the Environment Variable dialog.

---

**Note:**

BAM allows you to visually model your app and generates the code from the design models. You focus on the models, BAM handles the coding. The application accesses the **officestore** sample database included in the Genero Studio distribution. An FGLPROFILE file is provided in the samples/OfficeStoreRes directory. Entries in the file identify the SQLite officestore database.

---
e) Click OK to save and close the Advanced Properties dialog.

4. Using the File > Save … as menu, save the project with a unique name (such as TestSimpleBAM.4pw) to a folder on your file system.

Add a meta-schema to the project (4dbx)

Your application will access the sample officestore database. You must create a meta-schema file based on this database, and you will set the DBPATH environment variable to the directory containing the database file.

The file for a SQLite database named officestore.db, is located in My Genero Files/samples/OfficeStoreRes/database-sqlite/. From this database, you create a .4dbx schema file. It contains the information about the tables, columns, and relations of the relational database that is needed by your application. It will later be used to create items in your Business Application Diagram (4ba)

1. Select Database > Extract Schema...
The New Meta-schema dialog opens.

2. Complete the Meta-schema file step.
   a) Click on the File Browser icon next to the Meta-schema file path field. The Save File As dialog opens.
   b) Navigate to the My Genero Files/samples/OfficeStoreRes/database-sqlite/ directory. Enter "officestore" in the File name field. Select "Genero Database Schema Files (*.4dbx)" in the Save as type combobox. Click Save.
   c) Ensure the Insert the file in the project check box is selected.
   d) Select the Databases node.
   e) Click Next>.

3. Complete the Connection information step.
   a) Select "SQLite" as the Database type. The Database driver defaults to "dbmsqt".
   b) Click on the File Browser icon next to the Database file field. The Save File As dialog opens.
   c) Select the officestore.db file located in the My Genero Files/samples/OfficeStoreRes/database-sqlite/ directory.
   d) Click Test Connection to verify that the connection is successful.
   e) Click Finish.

The dialog closes, and the officestore.4dbx file is created in the

4. Add the DBPATH environment variable to the DB-SQLite environment set.

The application needs to know where the database file sits at runtime. DBPATH is used to specify the path to the file.

   a) Select Tools > Genero Configurations
      The Genero Configuration Management dialog opens.
   b) Under Environment Sets, select "DB-SQLite".
   c) Click the green plus icon.
      The Environment Variable dialog opens.
   d) In the Type combobox, select "Directory List".
   e) In the Name field, enter "DBPATH".
   f) In the Value field, use the File Browser to locate the path My Genero Files/samples/OfficeStoreRes/database-sqlite/.
   g) Click OK.
      The DBPATH environment variable is added to the DB-SQLite environment set.
   h) Ensure the DB - SQLite environment set is selected (checked).
   i) Click OK.
      The Genero Configuration Management dialog closes.

5. Save the changes to your project.
Related concepts
Database meta-schema (4dbx) on page 276
The 4dbx file is the database schema file for generated applications.

Create the Business Application diagram (4ba)
The Business Application diagram will contain a program and a form.

1. Select File > New > Genero BAM Desktop (for desktop applications) or File > New > Genero BAM Mobile (for mobile apps). Under the section Application Modeling, select Business Application Diagram (4ba) and click OK.
   A blank diagram (.4ba) opens.
2. Right-click on the blank diagram and select New > Program.
3. Right-click on the blank diagram and select New > CRUD Form.
4. Right-click the Program entity, and select Add Relation To. Click and drag from the Program to the Form entity to create a relation from program to form.

Figure 119: Defining a relationship
5. Save and name the BA Diagram (4ba) to your file structure and to the Application node in the project.
6. Save the changes to your project.

Implement the program and form
Implement the program and form that you placed on the Business Application diagram.

1. Right-click on the Program entity and select Implement Program. In the Save As dialog, assign a name for the program file and save it under the Application node. This file has a 4prg extension and is used to generate the source code for the main function.
2. Right-click on the Form entity and select Implement CRUD Form from Database to generate a form for the database table that you wish to access. Select the officestore database and the account table. Use the right arrow to add the first ten fields to the Selected Fields list.
Figure 120: New Form from Database wizard Column selection

3. Click Finish

4. Save the form (4fdm) to the project structure under the Application node.
Generate and run the application

With the program components implemented, you are ready to build and execute the application.

1. To build a generated application, right-click the Program entity on the BA diagram (4ba) and select Build Program from the context menu. Alternatively, right-click the Application node in the project and select Build. The results of the build are displayed in the Output view.

To better understand what is happening during the build of the program, turn on verbose mode using Tools > Preferences > Compiler and Runtime > Compilation Configuration.

The generated and compiled files are stored in the path specified in the Target Directory property of the Group node in the project.

2. To execute a generated application, right-click the Program entity in the BA Diagram (4ba) and select Execute Program. Alternatively, right-click the Application node in the project and select Execute. Toolbar icons are generated for a program's default actions, depending on the properties selected for the module's screen records and the fields on the form. The necessary business logic was created in the generated BDL files (4gl) to implement the relevant actions. See Working with forms on page 285.
Add a detail list to the form

Add a list to your form and create the master-detail relationship.

A Genero application can display a form that contains a master-detail relationship between two tables. The user can search for a row in the master table, and the corresponding rows in the detail table will also be displayed. The values in rows from both tables can be added, deleted, or modified.

The form must contain fields from both tables and the table relationships must be set.

1. In the form, enlarge the grid around the fields from the master table to make room for the fields from the other table.
2. Select Container > Data Control.
   The Extend Form from Database dialog displays, showing the Column Selection page.
3. Select these columns from the orders table: orderid, userid, orderdate, totalprice and creditcard. Click Next >.
4. Select Table as the container. and click Finish.
   The cursor is set to allow you to place your container on the form.
5. Draw the container within the enlarged grid, under the existing fields.
   This form design now contains fields from the master table (account) and a detail table (orders).

Figure 123: Form tab

6. Select the Records tab. Select the master table record and make sure the active property is checked. Select and identify a column as the unique key, if not already identified.

   Tip: The unique key is identified by the key icon.
7. Do the same for the detail table record.

8. Set the relationship between the master and detail records. Right-click the foreign key field in the detail table record (orders.userid, in our example), and select Add Relation To. Drag the arrow to point to the primary key in the master table record (account.userid).
9. Select **File > Save All** to save the project and the modifications to its contents.

10. Build and execute the program.

---

**Figure 126: Master-detail relationship**

**Figure 127: The application executes**
Related concepts
Add Reports on page 245
You can add a Genero Report Writer report to your generated application.

Related tasks
Add CRUD forms on page 238
A generated application can include multiple forms.
Add Zoom forms on page 240
Zoom forms contain logic that allow the user to pick a value from a list that is displayed in a form in a popup window.

Quick Start: Generating a mobile app

This quick start guides you through generating and running a simple, one form mobile app. The Business Application Modeler (BAM) allows you to visually model your app and generates the code from the design models. You focus on the models, BAM handles the coding.

Important: This quick start assumes you have configured Genero Mobile to run an app to a mobile device or emulator. See the appropriate configuration topics.

• Create a BAM mobile project on page 214
• Model the database on page 216
• Model the app on page 218
• Generate and run the app on page 218
• Add phone functionality to the app on page 219
• Customize the app on page 221
• Package and Deploy on page 222

Create a BAM mobile project
Create a new mobile project. By using the structure imposed by this project type, building, packaging and deploying your app to a mobile device is simplified.

1. Select File > New > Genero BAM Mobile > BAM Mobile Project (4pw) to create a project for your generated app.
2. Identify a project name and directory location for your project files.

Figure 128: BAM Mobile Project
3. From your new project, expand the **Project** group. Right-click on the **Application** node and select **Advanced Properties**. Note that the dependencies have already been set, ensuring that any files contained in the checked nodes will be included in your app.

4. If you are using an emulator, launch your emulator.
   
   This quick start assumes you have configured Genero Mobile to run an app to a mobile device or emulator. See **Configure for Genero Mobile** on page 142 to set up your iOS or Android™ mobile device or emulator.

**Mobile projects**

BAM Mobile Projects have predefined nodes for the basic structure of the project. These nodes correspond to the project directories created on disk in the project directory you specified. The project also includes the build rules needed to generate, package, and deploy the application.

---

**Table 35: Default Project structure**

<table>
<thead>
<tr>
<th>Node</th>
<th>Files to be saved to this node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>4prg (implemented program file from 4ba diagram)</td>
</tr>
<tr>
<td>Applicationflow</td>
<td>4ba</td>
</tr>
<tr>
<td>Config</td>
<td>FGLPROFILE</td>
</tr>
<tr>
<td>Database</td>
<td>4dbx, db</td>
</tr>
<tr>
<td>Entities</td>
<td>4fdm (forms or other entities from 4ba diagram)</td>
</tr>
<tr>
<td>Resources</td>
<td>XML files such as 4st for styles</td>
</tr>
</tbody>
</table>

**Related tasks**

Create a BAM mobile project   on page 214
Create a new mobile project. By using the structure imposed by this project type, building, packaging and deploying your app to a mobile device is simplified.

**Model the database**

From the database meta-schema file, you can graphically model a database by adding tables and constraints and generate a database creation script.

The database meta-schema file will contain the information about the tables, columns, and relations of the relational database that is needed by your app. By default, the database meta-schema file `mydatabase.4dbx` has been provided for you.

**Tip:** This procedure uses the provided default database. To rename the database file, see Rename the database in a BAM Mobile project on page 281.

1. In the **Project** group, expand the **Database** node, and double-click on the `4dbx` to open it. The Meta-Schema Manager is launched for viewing and modifying the schema.

   **Note:** The meta-schema diagram includes a seqreg table. For this quick start, you do not need to worry about this table. For more information about what this table does, see Managing SERIALs in a generated application on page 283

2. Right-click and select **Add Table**. Set the table’s **name** property to **account**. Right-click on the table to add the columns as shown.

   ![Figure 130: Create table](image)

3. Make the **id** column **SERIAL** and **not null**. Make the **id** column the primary key for the table. With the table selected, right-click to see the menu option for adding a constraint.
Figure 131: Set primary key

4. Save your file.

5. Right-click on the 4dbx file in your project and select Generate Database Creation Script. Select the option to populate database with sample data and then Generate.

Figure 132: Generate Database Creation Script

6. Save the file to the databaseMaintenance directory of your project ($ProjectDir) to overwrite the default file with your file.

a) Uncheck the Insert the file in the project option as the file is already in the project; you are just saving the file to disk.
7. In the project, expand the **DatabaseMaintenance** group. Right-click on the **CreateDatabase** program and select **Execute**. This will compile and run your script, creating and populating the database.

**Model the app**

The Business Application Diagram allows you to model your application graphically.

In this quickstart, you create a form based on a database table and implement the program.

1. Expand the **Applicationflow** node and open the **appflow.4ba** file. This is your Business Application Diagram and shows the flow of your program. You will see there is one Program entity (**main**) and one Form (**form**) entity.
2. Right-click on the Form entity and select **Implement CRUD Form from Database** to generate a form for the database table that you wish to access. Select all the fields in your table and transfer this selection to the Selected Fields list, which lists the desired fields for your form. Select **Finish** to accept the defaults from the rest of the wizard prompts.

Per the default, this form will be organized in a Grid layout.
3. Save your form to the **src** directory of your project ($ProjectDir) and in the **Entities** node of the project.
4. Return to the **appflow.4ba** file. Right-click on the Program entity (**main**) and select **Implement Program**.
5. Save the program to the **src** directory of your project ($ProjectDir) and in the **Application** program node.
6. Save the changes to your project.

**Generate and run the app**

Run your app to display to a device or emulator.

1. Right-click on the **Application** node in the project or on the **Program** entity in the BA diagram and select **Build**.

   **Note:** To better understand what is happening during the build of the program, turn on verbose mode using **Tools > Preferences > Compiler and Runtime > Compilation Configuration**

2. Right-click on the **Application** node in the project or on the **Program** entity in the BA diagram and select **Execute**.
Add phone functionality to the app

You can generate the code to interact with the mobile device's features. This example shows how to call a selected phone number from the app.

1. Open the appflow.4ba file.
2. Right-click on the BA diagram and select New > Phone.
3. Right-click on the Form entity and select Add Relation. Click on the Form entity again and drag the Relation arrow to the Phone entity.

4. Select the Relation between the Form and the Phone entity and set the Action property to callContact. Set the PhoneNumber property to the phone column (account.phone in our example database).
5. Save your changes to `appflow.4ba`.

6. Double-click on the **Form** entity to open it in Form Designer. Add a button that triggers your `callContact` action.
   a) Expand the canvas and the grid that is containing the form fields if you need room to add a button.
   b) Select **Widget > Button** from the menu.
   c) Draw a button onto your form. Set the `name` property of your button to `callContact`. The name of the button is also the action to be triggered by the button.
   d) Set the `text` property of the button to `Call Contact`.

7. To preview your form, select **Build > Preview**.

8. Make further changes to your form if you wish.

9. Save your changes to the form.

10. Execute your program, this will rebuild your program and execute it in one step.

11. Test the new button.

**Tip:** The sample data in the phone field is not a valid phone number. Run the app and select **Modify** to update the phone field with a phone number of your choice. On the Android emulator, the **MENU** button displays the available program actions.
Customize the app

Customize the app by changing a property value in the BA diagram.

At this point, you would continue to build your BA diagram to add and customize forms, zoom forms, and mobile peripheral entities, relations between the entities, and to customize the generated code as needed. There are many customization options in BAM.

Table 36: Customization example resources

<table>
<thead>
<tr>
<th>Example</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating forms for mobile devices.</td>
<td>See Working with Forms section in the Genero Studio User Guide.</td>
</tr>
<tr>
<td>Mobile form patterns. The Mobile Patterns demo includes a BA diagram with examples of various forms, relationships between forms, and form behavior.</td>
<td>See the Mobile form patterns topic in the Genero Studio User Guide.</td>
</tr>
<tr>
<td>Add custom code to the BAM generated code.</td>
<td>See the Adding custom code topic in the Genero Studio User Guide.</td>
</tr>
<tr>
<td>Change default rendering of the actions in the app.</td>
<td>See the Action rendering topic in the Genero Business Development Language User Guide.</td>
</tr>
</tbody>
</table>
For the purposes of this quick start, make a simple customization to your app by changing the form to open in ADD mode, instead of the default DISPLAY mode.

1. Open appflow.4ba.
2. Select the Form entity. The properties view shows all of the properties set on the form. Find the UI Settings section. (You can always display a view with the menu Window > Views).
3. Change the Open Mode property to ADD.
   This changes the program so that this form will open ready for the user to enter a new row of data.

4. Save your changes to appflow.4ba.
5. Execute the app again. (Right-click on the program entity (main) and select Execute.) New code is generated to reflect this property change.
6. Test the program by adding a new row.

**Package and Deploy**

Package the app and deploy it for testing on your connected device or emulator.

1. Expand the Packages group in your project.
2. Find the package that corresponds to the device or emulator you have configured.

3. Right-click on the package and select Deploy. This will build the package and deploy it.
What is Business Application Modeling (BAM)?
Genero Studio Business Application Modeling (BAM) is a graphical design tool to generate the code that extracts data and creates data models in BDL. It automatically generates the logic and source code for a database application to query, add, update and delete rows in database tables. BAM can generate desktop, web, and mobile applications.

BAM design process
BAM provides a complete framework for design and development, incorporating the entire life cycle of an application.
Figure 138: Application life cycle

With BAM:

- Create a project type pre-configured for a desktop or mobile app.
- Model the entities of your applications and their interconnection in a high level business diagram.
- Model the detailed design of your database, forms, services, and reports in sub-diagrams.
- Select options for the entities for user interface design and code generation.
- Automatically generate an application matching the behavior described in the diagrams without coding.
- Enhance the generated code with your own logic.

Is my application a good candidate for BAM?

If your report data is to be extracted from a database using SQL, then your report application is a good candidate for BAM. All CRUD forms -- to view, add, update, or delete data from the database are perfect candidates for BAM.

BAM does not generate code for:

- Non-database forms, such as login screens or forms displaying only computed fields
- Web components
- Specific process logic, such as database synchronization for mobile apps

Custom code can be added for any functionality that is not generated by BAM.
How code is generated

When you build an application from a Business Application diagram, the build rules define the various files that are input into the Code Generation Engine and the application code files that are output.

Figure 139: Code Generation flow

BAM Consolidates and Generates

Data Consolidation

The input from the BA diagram and related entities is gathered into a single XML file, which consolidates all the inputs into one package. This file is used when processing and generating the application code. This file could also be used to provide input to create the application models.

Code Generation

The XML file and a code template are used to generate the application code. The default Tcl template produces Genero 4gl files, but another tool could be used to generate the code (XSL translator for example). Custom code is preserved; if any custom code was created earlier, it is automatically restored in the newly generated application code.

Example

The build rules define the series of commands used to build and generate the code. In general, the Build rule for code generation:

- Saves custom code added by the user to the generated source files
- Generates the new source files without user code
- Restores the user code in the generated files
- Compiles the written and generated source files
- Links the compiled files

View default build rules selecting Projects > Edit Build Rules. This example shows the build rules used to generate the code for a Program entity in the Business Application diagram.
Figure 140: Default build rule for a generated program entity

Table 37: Default build rule example

<table>
<thead>
<tr>
<th>Build rule command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(generate)</td>
<td>The $(generate) command creates an intermediary XML file from modeled entities.</td>
</tr>
<tr>
<td>$(blockpoint) -code</td>
<td>BLOCK/POINT is extracted from previously generated and modified code.</td>
</tr>
<tr>
<td>tclsh on page 322</td>
<td>The tclsh executable generates the final file by using both a Tcl template file and the intermediary XML file created by the $(generate) command.</td>
</tr>
<tr>
<td>$(blockpoint) -storeGenerated</td>
<td>Extracted BLOCK/POINT code is put back into the generated code.</td>
</tr>
<tr>
<td>$(fglcomp)</td>
<td>The fglcomp tool compiles BDL program sources files into a p-code version.</td>
</tr>
<tr>
<td>Build rule command</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>$(move)</td>
<td>Moves the given file or directory to the given destination in a platform independent way.</td>
</tr>
</tbody>
</table>

**Reviewing the Build**

To better understand what is happening during the build of the program, turn on verbose mode using **Tools > Preferences > Compiler and Runtime > Compilation Configuration**. Compile a diagram file, program, or application and view the results in the output.

**Related concepts**

[The code generation template set](#) on page 1110

Genero Studio provides a standard template set of files that are used for code generation. The files are written in XML and Tcl.

**The modeling diagrams**

Business Application Modeling is based on several diagrams. Each diagram has its own purpose for modeling the application's features and behavior.

Diagrams are used at all stages in the development. They serve as inputs for code generation, and are considered source code for the application. They are always up to date. Diagrams can be customized with additional properties using the **settings.agconf** configuration file.

![Business Application diagram](image)

**Figure 141: Business Application diagram**

The Business Application diagram models the application flow and provides a high-level overview of the application.
Figure 142: Meta-schema Manager diagram

The Meta-schema Manager diagram models the database tables, columns, and constraints. See Meta-schema Manager on page 343.

Figure 143: Form Designer diagram

The Form Designer diagram models an application's forms and records. See Form Designer on page 486.
Business Application Modeling (BAM)

Figure 144: Business Record diagram

Business records model the data definition, structure and table relationships of the data used in a form, report, and/or web service.

See Business records (data sets) on page 493.

Mobile apps vs Desktop applications

Mobile apps and desktop applications can be generated using BAM, though mobile apps require a different way of thinking about and organizing an application.

Typical desktop applications often require large, complex master-detail forms, whereas mobile apps require multiple small, simple forms interacting with each other. BAM supports both of these patterns. When modeling the application, you can determine the number of forms and the way in which the forms will relate to each other.

Mobile apps do not support typical reporting and web services, so the options for modeling the app are restricted to program and form entities when working from a mobile project.

Related concepts

Quick Start: Generating a mobile app on page 214
This quick start guides you through generating and running a simple, one form mobile app. The Business Application Modeler (BAM) allows you to visually model your app and generates the code from the design models. You focus on the models, BAM handles the coding.

Mobile projects on page 215
BAM Mobile Projects have predefined nodes for the basic structure of the project. These nodes correspond to the project directories created on disk in the project directory you specified. The project also includes the build rules needed to generate, package, and deploy the application.

**Mobile form patterns** on page 285
Mobile platforms require special form patterns. These patterns can be modeled in BAM.

### The default template features

The default template set is designed to generate organized and functional code for a data-driven business application.

#### Code architecture

![Diagram of code architecture](image)

**Figure 145: Overview of template features**

#### Table 38: Features

This table lists the general features of the default template set used to generate Genero applications.

<table>
<thead>
<tr>
<th>Forms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUD and Zoom forms</td>
<td>The user interface of a generated program is based on the forms created for the program. See Add Forms on page 236.</td>
</tr>
<tr>
<td>Form states</td>
<td>See Form behavior in CRUD states on page 293</td>
</tr>
<tr>
<td>Functionality</td>
<td>See Enable and disable CRUD logic on page 291</td>
</tr>
<tr>
<td>Lookups</td>
<td>See Lookup fields on page 299</td>
</tr>
<tr>
<td>Pre-filled comboboxes</td>
<td>The values for a ComboBox list can be automatically retrieved from a database table. Typical usage includes providing a list of state or country values. Define a dynamically populated ComboBox on page 297</td>
</tr>
<tr>
<td>Data management</td>
<td>The generated code manages query by example (QBE), default values, multiple level data sets, all database operations through the database entity (Database meta-schema (4dbx) on page 276), data set key list in memory, and up to date checks before editing. See Modeling the database on page 276.</td>
</tr>
<tr>
<td>UI actions management</td>
<td>The generated code manages a Toolbar and menu and the action visibility and enabling, depending on the state of the program. See Default actions on page 316, Default Topmenu and Toolbar on page 319</td>
</tr>
<tr>
<td>Field activation</td>
<td>See Field activation on page 296.</td>
</tr>
</tbody>
</table>
### Database Management

<table>
<thead>
<tr>
<th>Database Management</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints</td>
<td>See Database meta-schema (4dbx) on page 276</td>
</tr>
<tr>
<td>Serials</td>
<td>See Managing SERIALs in a generated application on page 283</td>
</tr>
<tr>
<td>Table operations</td>
<td>See Default actions on page 316</td>
</tr>
<tr>
<td>Nested transactions</td>
<td>See Understanding what gets generated on page 303</td>
</tr>
<tr>
<td>Concurrency</td>
<td>See Managing concurrency on page 284</td>
</tr>
</tbody>
</table>

### Reports

<table>
<thead>
<tr>
<th>Reports</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database management</td>
<td>See Implement reports on page 247</td>
</tr>
<tr>
<td>Designs</td>
<td>See Add a Report Design Document (4rp) on page 248</td>
</tr>
</tbody>
</table>

### Web services

<table>
<thead>
<tr>
<th>Web services</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUD services</td>
<td>A service can be created either from a CRUD form or from a Zoom form. From a CRUD form the web service can have Create, Read, Update, and/or Delete operations. From a Zoom form, operations are limited to Read. See Create a Web Service from a form on page 256.</td>
</tr>
<tr>
<td>Database level services</td>
<td>See Modeling the database on page 276</td>
</tr>
</tbody>
</table>

### Related concepts

Adding custom code on page 303
This section includes topics on how to add custom code to the generated code.

### Configuring for BAM

When using the Business Application Modeler, you can specify both a global setup and a template-specific setup.

**Factory setup**

Factory setup is the standard installation.

**Global setup**

The global setup overrides the factory setup. It applies to all configurations for the current user for the Genero Studio installation on the current machine or remote machine.

Tools > Global setup menu option allows you to customize the global setup.

**Specific setup**

A specific setup is a template-specific setup. It is for the current user, is for a template set used by the Business Application Modeler. Use is option to override the global setup for a specific set of templates. The GSTSETUPDIR on page 168 environment variable, set in the Genero configuration, defines the template set. If this environment variable is not set, or not specified for the current configuration, the specific setup options are disabled.

The Tools > Specific setup menu option allows you to customize the specific setup.
BAM Projects

BAM projects help you organize the files necessary for generating Genero applications.

You have different types of BAM projects: first choose between a Desktop or Mobile application, then choose between a structured or simple project.

A structured project is created with a default structure of nodes and files, which guides the organization of your project. Once created, however, you have the freedom to modify the structure as you see fit. To view an example that starts with a structured BAM project, see the Quick Start guide for generating a mobile app.

A simple project is created without an expansive setup of nodes and files. Choose a simple project if you wish to define your own structure and organization from the beginning. To view an example that starts with a simple BAM project, see the Quick Start guide for generating a simple desktop application.

Related concepts
Structured projects on page 400
A structured project starts with a default logical structure in the Projects view and a default physical structure on disk.

Organizing projects on page 410
Information about organizing projects.

Modeling the application

Model the application by laying out the programs, forms, reports, and relationships between the entities on a BA diagram. Then, implement the programs and forms and reports to specify the details about each entity. Last, build the application to generate the code.

• The Business Application (BA) diagram on page 232
• Create a BA diagram on page 235
• Add and implement a program on page 235
• Add Forms on page 236
• Add Reports on page 245
• Add Web services on page 250
• Add Relations on page 270
• Add mobile device features (Photo, Gallery, Phone, Mail, SMS, Contact, Maps, Barcode) on page 272
• Import files into the diagram from the project on page 275

The Business Application (BA) diagram

The BA diagram is designed to model the application flow and provide a high-level overview of the application to be generated.

The BA diagram can include various entities to model the application. Relations connect the entities and specify the relationships between them. Each entity has properties to specify all the information needed to generate the code. Until implemented, the entities on the diagram are simply icons. Once implemented, files are created that represent the information and relationships about the entity. These files are used to generate the application code.
Table 39: Entities

<table>
<thead>
<tr>
<th>Entity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programs</td>
<td>A <em>Program</em> entity contains the information needed to generate the main logic to drive the application.</td>
</tr>
<tr>
<td>CRUD Form entity on page 237</td>
<td>A <em>CRUD Form</em> entity generates the user interface for the program. CRUD forms are used to Create, Read, Update, and Delete data from a database.</td>
</tr>
<tr>
<td>Zoom form entity on page 239</td>
<td>A <em>Zoom Form</em> generates a form used to select a value from a list and return the value to the program. It is generally related to a CRUD form field to assist the user with completing the form data entry.</td>
</tr>
<tr>
<td>Custom Form entity</td>
<td>A <em>Custom Form</em> is a form for which the code remains free.</td>
</tr>
<tr>
<td>Report entity on page 246</td>
<td>A <em>Report</em> entity generates the logic to retrieve data from the database and run a report based on a defined data definition and report layout.</td>
</tr>
<tr>
<td>Important: This feature is not supported on mobile platforms.</td>
<td></td>
</tr>
<tr>
<td>SOAP Web Service Server entity on page 254, JSON Web Service Server entity on page 261</td>
<td>A <em>Webservice Server</em> entity contains the main logic to publish services. It listens for incoming requests and executes the relevant service operation.</td>
</tr>
<tr>
<td>Important: This feature is not supported on mobile platforms.</td>
<td></td>
</tr>
<tr>
<td>SOAP Web Service entity on page 251, JSON Web Service entity on page 257</td>
<td>A <em>Webservice</em> entity generates a standalone Webservice.</td>
</tr>
<tr>
<td>Important: This feature is not supported on mobile platforms.</td>
<td></td>
</tr>
<tr>
<td>Entity</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>CRUD form with SOAP Web Service entity on page 253, Zoom form with SOAP Web Service entity on page 254, CRUD form with JSON Web Service entity on page 259, Zoom form with JSON Web Service entity on page 260</td>
<td>A CRUD form or Zoom form and Webservice generates the form and a standalone Webservice. <strong>Important:</strong> This feature is not supported on mobile platforms.</td>
</tr>
<tr>
<td>Relations on page 271</td>
<td>A Relation entity is used to define a relationship between entities on the BA diagram. Multiple relations can be set to the same form providing easier maintenance, but also flexibility on what CRUD operations are available to the form when it is opened by different actions.</td>
</tr>
<tr>
<td>Phone / PhoneRelation</td>
<td>A Phone and PhoneRelation entity is used to define that the form will generate code to launch the default phone app on the device and initiate dialing of the specified phone number.</td>
</tr>
<tr>
<td>Mail / MailRelation</td>
<td>A Mail and MailRelation entity is used to define that the form will generate code to invoke the user's default mail application for a new mail to send.</td>
</tr>
<tr>
<td>Gallery / GalleryRelation</td>
<td>A Gallery and GalleryRelation entity is used to define that the form will generate code to let the user select a picture from the mobile device's photo gallery and return a picture identifier.</td>
</tr>
<tr>
<td>Photo / PhotoRelation</td>
<td>A Photo and PhotoRelation entity is used to define that the form will generate code to let the user take a picture with the mobile device and return the corresponding picture identifier.</td>
</tr>
<tr>
<td>SMS / SMSRelation</td>
<td>An SMS and SMSRelation entity is used to define that the form will generate code to send an SMS text to one or more phone numbers.</td>
</tr>
<tr>
<td>Contact / ContactRelation</td>
<td>A Contact and ContactRelation entity is used to define that the form will generate code to let the user choose a contact from the mobile device contact list and return the vCard.</td>
</tr>
<tr>
<td>Map / MapRelation</td>
<td>A Map and MapRelation entity is used to define that the form will generate code to invoke a maps app with the current Global Positioning System (GPS) location of the mobile device.</td>
</tr>
<tr>
<td>Barcode / BarcodeRelation</td>
<td>A Barcode and BarcodeRelation entity is used to define that the form will generate code to use a barcode scanner from the mobile device.</td>
</tr>
</tbody>
</table>

**Related tasks**

Add mobile device features (Photo, Gallery, Phone, Mail, SMS, Contact, Maps, Barcode) on page 272
The default template for modeling mobile apps includes features for interacting with a mobile device's default apps such as phone, email, text, photos, and maps.

**Create a BA diagram**

The BA diagram is designed to model the application flow and provide a high-level overview of the application to be generated.

1. Select **File > New > Genero BAM Desktop** or **File > New > Genero BAM Mobile**.
2. Select **Application Modeling > Business Application Diagram**.
3. Right-click in the background of the diagram to display a context menu of options. Add entities such as Programs, CRUD forms, Zoom forms, Reports, and Web Services. Repeat this step until you have added all entities that model your application.
4. Define the relationships between the entities with a relation entity. Right-click on an entity, such as a Program, and select **New Relation**. Click and drag the Relation arrow to the entity to which you want to set the relationship. Repeat this step until all relationships have been defined.
5. Set properties on the entities.

**Add and implement a program**

Implementing a program entity on the BA diagram creates the 4prg file which is used to generate the program code.

1. Open the BA diagram to which you want to add a program.
2. Right-click on the diagram and select **New, Program** to add a new Program entity.
3. From the BA diagram, select a program entity.
4. Right-click and select **Implement Program**. Save and name the 4prg file and optionally select to insert it into a location in the project.
   The 4prg file is in the project along with placeholders in the Intermediate Files folder for the xml and 4gl source files that will be generated for the program.

**Program entity**

A Program entity contains the information needed to generate the main logic to drive the application.

A Program entity is represented as a 4prg file.

When a program entity is implemented, a 4gl source file is generated which contains the MAIN function for the application. This 4gl file is also used to combine the other generated 4gl files into a Genero application.

Any changes to the diagram are included in the subsequent re-generating of the program code.

The generated code can be customized.

**Table 40: Entity Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of entity.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of entity.</td>
</tr>
<tr>
<td>File Name</td>
<td>Full path to file.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the program entity.</td>
</tr>
</tbody>
</table>

Right-click a Program entity in the diagram to display a context menu of options.
Table 41: Context Menu

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement Program</td>
<td>Implementing a program entity on the BA diagram creates the 4prg file which is used to generate the program code. Until you create the Program 4prg file, the Program entity on the diagram is simply an icon.</td>
</tr>
<tr>
<td>New Relation</td>
<td>Add a relationship to another entity on the diagram.</td>
</tr>
<tr>
<td>Execute Program / Build Program</td>
<td>Builds and/or executes the source code for the program based on the related entities in the diagram.</td>
</tr>
<tr>
<td>Rename</td>
<td>Changes the name property of the entity.</td>
</tr>
<tr>
<td>Convert to</td>
<td>Converts entity from one type to another.</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Locates and highlights the selected item in the project.</td>
</tr>
<tr>
<td>Hide / Show all Items</td>
<td>Hides the entity from view. Show again by selecting the Show all Items option.</td>
</tr>
<tr>
<td>Filter Items</td>
<td>The Filter View dialog allows you to hide and show items on a diagram.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the entity.</td>
</tr>
<tr>
<td>Select All</td>
<td>Select all entities or all entities of the same type on the diagram.</td>
</tr>
</tbody>
</table>

Related concepts

Understanding what gets generated on page 303
4gl files are generated for each diagram entity.

Add Forms

The user interface of a generated program is based on the forms created for the program.

There are three types of generated forms: CRUD, Zoom, and Custom forms.

**CRUD Form**

CRUD forms are used to Create, Read, Update, and Delete data from a database. CRUD forms are typical data input and browsing forms. One CRUD form can call another CRUD form or a Zoom form.

**Zoom Form**

A Zoom form generates a form used to select a value from a list and return the value to the program. It is generally related to a CRUD form field to assist the user with completing the form data entry. For example, a Zoom form could be created that displays a list of country codes from the country database table. When the user triggers the zoom action to occur, either by selecting the zoom icon on the Toolbar or from the zoom field, a list of country codes displays in a popup window. When a country code is chosen from the list, the zoom form closes and the value is populated in the corresponding field on the main form.

**Custom Form**

A Custom Form is a form for which the code remains free.

Examples of custom forms include:

- A non-database form, such as a login form.
- A database form with only a single operation, such as an insert-only form.
Business Application Modeling (BAM)  

- A form that serves as the user interface for JSON calls.

**Related concepts**  
**Working with forms** on page 285  
Special properties are used to specify what logic should be generated for a form and how the form should behave during the various states (DISPLAY, MODIFY, ADD, SEARCH).

**CRUD Form entity**  
A **CRUD Form** entity generates the user interface for the program. CRUD forms are used to Create, Read, Update, and Delete data from a database.

A CRUD form entity is represented as a 4fdm file which can be opened and edited in Form Designer.

When a CRUD form entity is implemented from the diagram, the 4fdm file is created as well as 4gl source files containing the program logic to access and manipulate the database tables contained in the form.

Any changes to the CRUD form entity properties in the BA diagram or in the 4fdm file are included in the subsequent re-generating of the program code.

The generated code can be customized.

Properties can be set specify how the form should behave during the various states (DISPLAY, MODIFY, ADD, SEARCH).

If your form has a valid relation on the BA diagram to a Report entity, additional actions are generated for the Toolbar and Topmenu to launch the report. The Report Option properties are used to define which report actions should be generated.

**Important:** This feature is not supported on mobile platforms.

**Table 42: Report Option Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>quickPrint</td>
<td>Defines if print action is available.</td>
</tr>
<tr>
<td>quickPreview</td>
<td>Defines if a preview action is available. When implemented, a Preview button is placed on the form. If the application is running in the Genero Desktop Client (GDC), the report is previewed in the Genero Report Viewer (GRV). If the application is running in the Genero Browser Client (GBC), the report is previewed in the Genero Report Viewer for HTML5 (GRV for HTML5)</td>
</tr>
<tr>
<td>quickPDF</td>
<td>Defines if export to PDF action is available.</td>
</tr>
<tr>
<td>quickHTML</td>
<td>Defines if export to HTML action is available.</td>
</tr>
<tr>
<td>quickXLS</td>
<td>Defines if export to XLS action is available.</td>
</tr>
<tr>
<td>quickRTF</td>
<td>Defines if export to RTF action is available.</td>
</tr>
<tr>
<td>canExport</td>
<td>Defines if the PDF, HTML, XLS, and/or RTF print actions are available. If canExport is checked and quickPDF, quickHTML, quickXLS, or quickRTF are also checked, the action will be available.</td>
</tr>
</tbody>
</table>

Right-click the Form entity in the diagram to display a context menu of options.

**Table 43: Context Menu Options**

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement CRUD Form</td>
<td>Creates a new blank managed form definition file (4fdm) to design from scratch.</td>
</tr>
</tbody>
</table>
### Menu Option | Description
--- | ---
Implement CRUD Form from Database | Provides a wizard to create a managed form definition file (4fdm), allowing you to pick the columns and general display of your form. This option is generally preferred. The form can be modified after it is created.
Open | Opens form in Form Designer.
Convert to | Converts entity from one type to another.
Add Relation To | Add a relation starting from the select form.
Locate in Project | Locates and highlights the selected item in the project.

**Related concepts**

**Understanding what gets generated** on page 303
4gl files are generated for each diagram entity.

**Form behavior in CRUD states** on page 293
The **UI Settings** properties specify the initial and default state of the selected form as well as the behavior of the form during each state (**DISPLAY**, **MODIFY**, **ADD**, and **SEARCH**). These settings can be set on the form entity in the BA diagram, but overwritten by a relation to the form.

### Add CRUD forms
A generated application can include multiple forms.

This task assumes that you have created a **BA diagram** with a Program entity.

1. Right-click on the open BA diagram and select **New CRUD Form**.
2. Right-click the new form and select **Implement CRUD Form from Database** from the context menu.
3. Select the database table and fields that you want to have on the new form (the products table, in our example).
   Finish and save the form definition file (4fdm) to your project.
4. To call one form from another, create a second CRUD form by repeating steps 2 and 3.
5. Right-click the first form icon and select **New Relation**. Click the Form icon again and drag the relation arrow to the second form.
6. Select the relation arrow in the BA diagram and set the **action** property in the Properties view to the action name that you wish to trigger (**products** in our example).
7. Save the forms and modified BA diagram.
8. Build and run the program. A button is displayed on the calling form to trigger the action to call the secondary form. This button is added at runtime and can be removed by adding the action to the Toolbar and/or Topmenu. To do so, modify the default Toolbar (4tb) and/or Topmenu (4tm) files. Modify the action's display attributes by modifying the action defaults file (4ad).
**Figure 147: BA diagram with two forms**

**Related concepts**

*Enable and disable CRUD logic* on page 291

Form functionality properties (canDisplay, canAdd, canModify, canDelete, canSearch, canEmpty) can be set on each record of a form to specify whether the program logic of display, add, update, delete, search and/or display empty should be generated. Generated functionality can be disabled on a relation to the form in the BA diagram.

**Zoom form entity**

A *Zoom Form* generates a form used to select a value from a list and return the value to the program. It is generally related to a CRUD form field to assist the user with completing the form data entry.

A Zoom form entity is represented as a 4fdz file which can be opened and edited in Form Designer.

When a Zoom form entity is implemented from the diagram, the 4fdz file is created as well as 4gl source files that contain the logic to allow a user to pick a value from a list that is displayed in a form in a popup window.

Any changes to the Zoom form entity properties in the BA diagram or in the 4fdz file are included in the subsequent re-generating of the program code.

The generated code can be customized.

Properties can be set specify how the form should behave during the various states (DISPLAY, SEARCH).

Right-click the Zoom Form entity in the diagram to display a context menu of options.

**Table 44: Context Menu Options**

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Opens form in Form Designer.</td>
</tr>
<tr>
<td>Implement Zoom</td>
<td>Creates a new blank zoom form definition file (4fdz) to design from scratch.</td>
</tr>
<tr>
<td>Implement Zoom from</td>
<td>Provides a wizard to create a zoom form definition file (4fdz), allowing you to pick the columns and general display of your form. This option is generally preferred. The form can be modified after it is created.</td>
</tr>
<tr>
<td>Database</td>
<td></td>
</tr>
<tr>
<td>Convert to</td>
<td>Converts entity from one type to another.</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Locates and highlights the selected item in the project.</td>
</tr>
</tbody>
</table>
Menu Option | Description
---|---
Filter Items... | The Filter View dialog allows you to hide and show items on a diagram.

Related concepts
Understanding what gets generated on page 303
4gl files are generated for each diagram entity.

Form behavior in CRUD states on page 293
The UI Settings properties specify the initial and default state of the selected form as well as the behavior of the form during each state (DISPLAY, MODIFY, ADD, and SEARCH). These settings can be set on the form entity in the BA diagram, but overwritten by a relation to the form.

Related tasks
Add Zoom forms on page 240
Zoom forms contain logic that allow the user to pick a value from a list that is displayed in a form in a popup window.

Add Zoom forms
Zoom forms contain logic that allow the user to pick a value from a list that is displayed in a form in a popup window.

This task assumes that you have created a BA diagram with a Program entity and at least one CRUD form entity.

1. Right-click on the open BA diagram and select New Zoom Form.
2. Right-click the new form and select Implement Zoom from Database from the context menu.
3. Select the database table (in our example, the country table) that contains the values to be displayed in the zoom form. Select the column that corresponds to the field on the main form. In our example, this is the country code. (You can select the country name also, to display it in the zoom form if you wish; the code field will be identified as the unique key later in this process.)

![Figure 148: Example zoom form definition](image)

4. Switch from the form design to the Records tab and select the master table record. Confirm that the active property is checked.
5. Select the record and confirm that the unique key property is set. If not, set the unique key on one of the fields in the record.

![Figure 149: Records tab.](image)

6. Save the Zoom form (4fdz) and add it to the project structure under the Application node.
7. The CRUD form has to be modified to trigger the zoom form when needed. In this example, the country.code field is changed to a ButtonEdit, which can trigger an action when the user clicks it. When the user selects the country code from the zoom form, it is automatically inserted into the country.code field on the main form. From the BA diagram, right-click on the main form and choose **Open Form**.

8. Select the field containing the foreign key (in this example, the `account.country` field.) Right-click and select **Convert Widget** to change the widget to a ButtonEdit.

9. Scroll to the bottom of the properties list, and set the value of the **action** property for the ButtonEdit field to a unique action name (such as `zoom`).

10. Save the form.

11. Create a relation between the CRUD form and the Zoom in the BA diagram. On the open BA diagram, Right-click the CRUD form icon and select **New Relation**. Click the CRUD form icon again and drag the relation arrow to the second form.

12. Select the relation arrow to display the relation properties, and complete the entries for the following properties:

<table>
<thead>
<tr>
<th>Section</th>
<th>Property</th>
<th>What to enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source UI Settings</td>
<td>Action</td>
<td>Enter the action name in the <strong>action</strong> property that is the same as the one assigned to your ButtonEdit field (<em>zoom1</em> in this example). If there is more than one zoom form in an application, the action name must be unique.</td>
</tr>
<tr>
<td>Position</td>
<td>Source Field</td>
<td>Set the Source Field to the name of the field from the business record of the form that represents the foreign key that will ultimately be populated by the primary key selected from the zoom form.</td>
</tr>
<tr>
<td>Section</td>
<td>Property</td>
<td>What to enter</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Filter</td>
<td>Source Field</td>
<td>Restrict which records appears in the zoom form by using this field to specify the criteria used to filter the records displayed in the zoom. When set, a single value is sent to the zoom form to use as a filter. <strong>Tip:</strong> If the properties in the Filter group are not set, the zoom displays all rows from its underlying table. It is common to not set Filter properties in a zoom relation, as the goal is to select a value from a full list, not a filtered list.</td>
</tr>
<tr>
<td></td>
<td>Destination Field</td>
<td>Specify the field in the zoom form record to use when applying a filter based on the value provided from the source record.</td>
</tr>
<tr>
<td>Destination UI Settings</td>
<td>Open Mode</td>
<td>Set to DISPLAY or SEARCH.</td>
</tr>
<tr>
<td></td>
<td>Default Mode</td>
<td>Set to DISPLAY or SEARCH.</td>
</tr>
</tbody>
</table>
Figure 151: Business Application diagram

13. Save all diagrams and forms.

**Custom Form entity**

A **Custom Form** is a form for which the code remains free.

Examples of custom forms include:

- A non-database form, such as a login form.
- A database form with only a single operation, such as an insert-only form.
- A form that serves as the user interface for JSON calls.

A Custom Form entity is represented as a 4fdec file which can be opened and edited in Form Designer.

When a Custom Form entity is implemented from the diagram, the 4fdec file is created as well the following generated code:

- The PUBLIC TYPE definition related to the records of the custom form.
• The `uiOpenForm` function used by incoming relations to open the custom form (and `uiOpenFormByKey` if the records of the custom form contain a unique key).
• The functions related to the outgoing relations.

**Important:** If you create a non-database custom form and involve it in a relation with a form that specifies a database, update the non-database custom form to specify the name of the database in its `databaseName` property.

The generated code can be customized.

Custom forms provide features not supported by CRUD forms:
• Multiple master records are supported.
• A form without any record is supported. This is typically a form with a static field.

Right-click the Custom Form entity in the diagram to display a context menu of options.

**Table 45: Context Menu Options**

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Opens form in Form Designer.</td>
</tr>
<tr>
<td>Implement Custom Form</td>
<td>Creates a new blank Custom Form definition file (4fdc) to design from scratch.</td>
</tr>
<tr>
<td>Implement Custom Form from Database</td>
<td>Provides a wizard to create a Custom Form definition file (4fdc), allowing you to pick the columns and general display of your form. This option is generally preferred. The form can be modified after it is created.</td>
</tr>
<tr>
<td>Convert to</td>
<td>Converts entity from one type to another.</td>
</tr>
<tr>
<td>Add Relation To</td>
<td>Add a relation starting from the select form.</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Locates and highlights the selected item in the project.</td>
</tr>
<tr>
<td>Filter Items...</td>
<td>The Filter View dialog allows you to hide and show items on a diagram.</td>
</tr>
</tbody>
</table>

**Related tasks**

Add Custom Forms on page 244
Custom forms rely on you to provide the necessary code.

Add Custom Forms
Custom forms rely on you to provide the necessary code.

This task assumes that you have created a BA diagram with a Program entity.

1. Right-click on the open BA diagram and select **New > Custom Form**.
2. Right-click the new form and select either **Implement CRUD Form** or **Implement CRUD Form from Database** from the context menu.
3. Complete the design of the form and save your changes.
4. Add and configure the relation from another entity in the BA diagram to your newly created custom form.
   When you add a relation to a custom form, the relation type is set to CustomFormRelation.
5. Provide custom code for the custom form within the provided `<POINT>` and `<BLOCK>` elements of the generated Genero source file.
   The Genero source file for the custom form shares the same name as the custom form, but with a `.4gl` extension. It can be found in the Intermediate Files virtual folder of the node containing the custom form file. For more information on coding within `<POINT>` and `<BLOCK>` elements, see Adding custom code on page 303.

Examine the demo applications for examples of custom forms. For example:
• In the **OfficeStore** demo, the **OrderLoginCustomForm** form provides an example of a custom form implementing a login.
• In the **OfficeStoreMobile** demo, the **AccountSplash** form provides an example of a custom form implementing a splash screen.

To view the forms, open the `.4fdc` file. To view the generated code, open the associated `.4gl` file.

**Implement a form**

Implementing a form entity on the BA diagram creates the file used to generate the form code.

1. From the BA diagram, select a form entity.
2. Right-click and select an option to implement the form.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement CRUD Form</td>
<td>Creates a new blank managed form definition file (<code>4fdm</code>) to design from scratch.</td>
</tr>
<tr>
<td>Implement CRUD Form from Database</td>
<td>Provides a wizard to create a managed form definition file (<code>4fdm</code>), allowing you to pick the columns and general display of your form. See Data Control wizard on page 535.</td>
</tr>
<tr>
<td>Implement Zoom Form</td>
<td>Creates a new blank zoom form definition file (<code>4fdz</code>) to design from scratch.</td>
</tr>
<tr>
<td>Implement Zoom Form from Database</td>
<td>Provides a wizard to create a zoom form definition file (<code>4fdz</code>), allowing you to pick the columns and general display of your form. See Data Control wizard on page 535.</td>
</tr>
<tr>
<td>Implement Custom Form</td>
<td>Creates a new blank Custom Form definition file (<code>4fdc</code>) to design from scratch.</td>
</tr>
<tr>
<td>Implement Custom Form from Database</td>
<td>Provides a wizard to create a Custom Form definition file (<code>4fdc</code>), allowing you to pick the columns and general display of your form.</td>
</tr>
</tbody>
</table>

3. Save the form to your project.

   Placeholder `4gl` and `xml` files are created in the Intermediate Files folder in the project. When the form or project is compiled, these files are populated with the logic for your form.

**Related concepts**

- Understanding what gets generated on page 303
- `4gl` files are generated for each diagram entity.

*Form Designer* on page 486

*Form Designer* is a visual editor that supports the creation, editing, and layout of Genero forms in Genero Studio.

**Add Reports**

You can add a Genero Report Writer report to your generated application.

1. Open the BA diagram to which you want to add a report.
2. Right-click on the diagram and select **New Report Data** to add a new Report Data entity.
3. Right-click on the Form entity from which you want to access the report and select **New Relation**. Click on the Form entity again and drag the relation link from the Form entity to the Report Data entity.
4. Select the relation link between the Form and Report Data entities. In the Properties panel, change the **Type** property to **ReportRelation**.
A Report entity generates the logic to retrieve data from the database and run a report based on a defined data definition and report layout.

A Report entity is represented as a 4rd file which can be opened and edited.

When a Report entity is implemented from the diagram, the 4rd file is created as well as 4gl source files containing the report’s driver and routing instructions.

Any changes to the Report entity properties in the BA diagram or in the 4rd file are included in the subsequent regenerating of the program code.

The generated code can be customized.

Right-click the Report Data entity in the diagram to display a context menu of options.

### Table 46: Context Menu Options

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement Report Data</td>
<td>Creates a new blank Report Data entity (4rd) to build from scratch.</td>
</tr>
<tr>
<td>Implement Report Data</td>
<td>Provides a New Record wizard to select tables and columns to include in the 4rd file (4rd). This option is generally preferred. The business record can be modified after it is created.</td>
</tr>
<tr>
<td>from Database</td>
<td></td>
</tr>
<tr>
<td>Open Business Record</td>
<td>Opens report business record.</td>
</tr>
<tr>
<td>Convert to</td>
<td>Converts entity from one type to another.</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Locates and highlights the selected item in the project.</td>
</tr>
</tbody>
</table>
Implement reports
The report record contains the data definition, structure, and table relationships required to generate a rdd (Report Data definition file). The rdd file is used in conjunction with a Genero report definition (4rp) file to automatically generate the reports.

1. Right-click on the Report Data entity and select Implement Report Data from Database. Select the desired tables and columns.

2. Select the business record to display its properties. Make sure that the active property is checked and that the masterTable property contains the database table name.

3. If more than one table was added to the record, select the Query property and specify the joins between the tables in the Query Editor.

4. Confirm that the unique key property is set on the field in the record that represents the primary (unique) key in the database table.

5. Save the Report Data file (4rd) to your project.

6. Build the application. The needed data definition file (rdd) is generated for you.

7. Run the application. Notice that additional actions have been added to the Toolbar and Topmenu for reports.

8. Select Preview from the Toolbar to view the report. The report will run using the default layout. You can add a Report Design Document (4rp) to customize the look and feel of the report.
Business records model the data definition, structure and table relationships of the data used in a form, report, and/or web service. Business records are designed and modified in the Business Records diagram.

**Add a Report Design Document (4rp)**
You can customize the look and feel of your report with a Report Design Document (4rp).

1. From the Genero Studio main menu, select **File>>New, Reports, Report Designs** and select the type of report you wish to create.

2. Select the **Data View** tab from the Project View.

3. Select **Open Schema File...** to associate the generated report data definition file (rddl) with this report design. You can find the name of the generated rddl file in your project's **Intermediary Files** listing.
4. **Design your report** by dragging and dropping fields from the Data View tab to your report design.

5. Save and name the Report Design Document (4rp) to your project, such as `listreport.4rp`.

6. Return to the BA diagram and select the relation link between the Form and Report Data objects.

7. Supply the name of your Report Design Document, such as `listreport.4rp`, for the **Report File** property.

8. Build and run the application and select Preview from the Toolbar to view the report with the new design.

**Report print settings**

The **Report Print Settings** dialog appears when the reportsetup action is triggered from a generated application.

By default, the reportsetup action is bound to the **Print...** button in the generated user interface Toolbar.

Options on the dialog are controlled by properties set on the **ReportRelations**, the relationships between the current Form object and one or more Report Data entities on the Business Application diagram (4ba).
Figure 154: Report Print Settings

Table 47: Report Print Settings Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Fields</td>
<td>Select Fields is enabled if there is not a Report Designer (4rp) file specified in the Report File property for the ReportRelation to the current Form. Select Fields is then used to select the fields to print on the report.</td>
</tr>
<tr>
<td>Preview</td>
<td>Displays the selected report in the Genero Report Viewer.</td>
</tr>
<tr>
<td>Printer</td>
<td>Select printer to print report. Options are Default Local Printer or Choose a Local Printer.</td>
</tr>
<tr>
<td>Export Format</td>
<td>Export selected report to PDF, HTML, XLS, or RTF.</td>
</tr>
</tbody>
</table>

Related concepts

Default Topmenu and Toolbar on page 319
The form for your generated program contains default action views (in a Topmenu and Toolbar) allowing the user to trigger the program actions.

Add Web services
You can generate SOAP or JSON Web services.

Web services are a standard way of communicating between applications over an intranet or Internet. They define how to communicate between two entities: a server that exposes services and a client that consumes services. Web services may provide programmable access to the functions of a form or a standalone service without a form.
You can generate both SOAP or JSON Web services. For more information on the differences between SOAP and JSON (or RESTful) Web services, refer to the *Genero Business Development Language User Guide*.

**SOAP Web services**
You can generate SOAP Web services.

**SOAP Web Services**
SOAP Web Services are modeled with:

- SOAP Web Service Server
- SOAP Web Service
- CRUD Form + SOAP Web Service
- Zoom Form + SOAP Web Service

For SOAP Services, you must create the client. See *Create the client for a SOAP Web Service* on page 257.

For more information regarding SOAP Web services, refer to the *Genero Business Development Language User Guide*.

**SOAP Web Service entity**
A Web Service entity generates a web service with its CRUD operations, but with no accessible form. CRUD operations are used to Create, Read, Update, and Delete data from a database.

It includes one global *read* operation that reads all records in once and one global *create* operation that creates all records at once.

For a SOAP Web Service entity, the Type property states "WebService". The XML representation of a SOAP Web Service entity (the Web service definition file) is a .4ws file.

The Web Service entity can be opened and edited in the Form Designer Records tab. When a Web Service entity is implemented from the diagram, both the XML file (.4ws or .4w3j) and the .4gl source files containing the program logic to create and set up a web service and its CRUD operations are created. Any changes to the Web Service entity properties in the BA diagram or in the XML file (.4ws or .4w3j) are included in the subsequent re-generating of the program code. The generated code can be customized.

See the topic *Introduction to Web Services* in the *Genero Business Development Language User Guide* for more information on Web Services concepts.

**Properties of the Web Service entity**
Select the Web Service entity in the Business Application diagram to view and set the entity properties.

| **Table 48: Web Service entity Properties, Object Category** |
|---|---|---|
| **Group** | **Property** | **Description** |
| Object | Name | Name of entity. |
| | Type | Type of entity: |
| | | - WebService (for a SOAP Web Service). |
| | | - WebServiceJSON (for a JSON Web Service). |
| | File Name | Full path to file. |
| | Description | Description of web service. Optional. |
| Web Service | Service Name | Unique publishable service name. |
| | Comment | Service comment that will appear in the \texttt{wsdl} file. |
Contextual menu for the Web Service entity

Right-click a Web Service entity in the diagram to display a context menu of options.

Table 49: Web Service entity context menu options (partial listing)

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement Web Service</td>
<td>Creates a new blank managed Web service definition file to design from scratch.</td>
</tr>
<tr>
<td>Note:</td>
<td>Once created, this menu option disappears; use Open for any future modifications.</td>
</tr>
<tr>
<td>Implement Web Service from Database</td>
<td>Provides a wizard to create a managed Web service definition file, allowing you to pick the columns to use in the CRUD operations. This option is generally preferred. The Web service can be modified after it is created.</td>
</tr>
<tr>
<td>Note:</td>
<td>Once created, this menu option disappears; use Open for any future modifications.</td>
</tr>
<tr>
<td>Open</td>
<td>Open the Business Record for the selected Web Service entity.</td>
</tr>
<tr>
<td>Convert to</td>
<td>Converts entity from one type to another.</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Locates and highlights the selected item in the project.</td>
</tr>
</tbody>
</table>

Web Service entity Business Record

Once the Web service definition file ( .4ws or .4wj ) exists, right-click the Web Service entity and select Open to view the Business Record diagram.

In the Structure view, select the Root node. All Web services share the same set of properties in the Root node.

Object group

name, databaseName

In the Structure view, select a Record node. All Web services share the same Record node properties for the following groups:

Object group

name

Modeling group

active, masterTable, unique key

Query group

query

Functionality group

canDisplay, canAdd, canModify, canDelete, canSearch

For the same Record node, however, only SOAP Web services contain the properties that provide for XML and XSD Schema Serialization attributes:

Web Service group

XMLAll, XMLSequence, XMLList, XMLElementNamespace, XMLAttributeNamespace, XSTypename, XSTypenamespace

See the section on XML serialization in the Genero Business Development Language User Guide for more information.

In the Structure view, select a RecordField node. All Web services share the same RecordField node properties for the following groups:

Object group

name

Modeling group

lookup

Field group

fieldType, sqlTabname, colName, fieldIdRef, dataType

Web Service group

public

Miscellaneous group

defaultValue
For the same RecordField node, however, only SOAP Web services contain the properties that provide for XML and XSD Schema Serialization attributes:

**Web Service group**

- XMLOptional, XMLElement, XMLAttribute,
- XMLName, XSDType, XSDLength, XSDMinLength,
- XSDMaxLength, XSDEnumeration, XSDWhiteSpace,
- XSDPattern, XSDMinInclusive, XSDMaxInclusive,
- XSDMinExclusive, XSDMaxExclusive,
- XSDTotalDigits, XSFractionDigits

See the section on XML serialization in the *Genero Business Development Language User Guide* for more information.

**Related concepts**

Add Web services on page 250

You can generate SOAP or JSON Web services.

**CRUD form with SOAP Web Service entity**

CRUD forms with web services allow programmable access to a form's functionality through a published service.

A **CRUD form and Web Service** entity generates the code for a CRUD form (a GUI doing create, read, update and delete functionality) and the code for a single standalone Web Service. There is no relation between the form and standalone Web service, however you can use the same object in two different situations: to display a CRUD form and to define a Web Service.

For a **CRUD form with SOAP Web Service**, the Type property states "FormWebService". The XML representation of a CRUD form with SOAP Web Service entity is a .4fdmws file.

When you implement a CRUD form with Web Service entity, you have the option to:

- Implement Form and Web Service
- Implement Form and Web Service from Database

The object files can be opened and edited in Form Designer. In addition to the properties of a **CRUD Form entity** on page 237, additional properties are found under the heading **Web Service**.

**Table 50: Web Service properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service name</td>
<td>Unique publish-able service name.</td>
</tr>
<tr>
<td>Comment</td>
<td>Service comment that will appear in the wsdl file.</td>
</tr>
</tbody>
</table>

Right-click the entity in the diagram to display a context menu of options.

**Table 51: Context Menu Options (partial list)**

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement CRUD Form and Web Service</td>
<td>Creates a new blank managed web service definition file to design from scratch.</td>
</tr>
<tr>
<td>Implement CRUD Form and Web Service from Database</td>
<td>Provides a wizard to create a managed web service definition file, allowing you to pick the columns to use in the CRUD operations. This option is generally preferred. The web service can be modified after it is created.</td>
</tr>
<tr>
<td>Convert to</td>
<td>Converts entity from one type to another.</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Locates and highlights the selected item in the project.</td>
</tr>
</tbody>
</table>
Related concepts
Form behavior in CRUD states on page 293

The UI Settings properties specify the initial and default state of the selected form as well as the behavior of the form during each state (DISPLAY, MODIFY, ADD, and SEARCH). These settings can be set on the form entity in the BA diagram, but overwritten by a relation to the form.

Zoom form with SOAP Web Service entity
Zoom forms with web services allow programmable access to a zoom form’s functionality through a published service.

A Zoom form and Web Service entity generates the code for a Zoom form (a GUI supporting read operations only) and the code for a single standalone Web service. There is no relation between the form and standalone Web service, however you can use the same object in two different situations: to display a read-only form and to define a Web Service.

Note: Only Read operation is generated for a Zoom form and Web Service entity.

For a Zoom form with SOAP Web Service entity, the Type property states "ZoomWebService". The XML representation of a Zoom form with SOAP Web Service entity is a .4fdzws file.

When you implement a Zoom form with Web Service entity, you have the option to:

• Implement Form and Web Service
• Implement Form and Web Service from Database

The object files can be opened and edited in Form Designer. In addition to the properties of a Zoom form entity on page 239, additional properties are found under the heading Web Service.

Table 52: Web Service properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service name</td>
<td>Unique publish-able service name.</td>
</tr>
<tr>
<td>Comment</td>
<td>Service comment that will appear in the wsdl file.</td>
</tr>
</tbody>
</table>

Right-click the entity in the diagram to display a context menu of options.

Table 53: Context Menu Options (partial list)

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement Zoom Web Service</td>
<td>Creates a new blank managed web service definition file to design from scratch.</td>
</tr>
<tr>
<td>Implement Zoom Web Service from Database</td>
<td>Provides a wizard to create a managed web service definition file, allowing you to pick the columns to use in the read operations. This option is generally preferred. The web service can be modified after it is created.</td>
</tr>
<tr>
<td>Convert to</td>
<td>Converts entity from one type to another.</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Locates and highlights the selected item in the project.</td>
</tr>
</tbody>
</table>

SOAP Web Service Server entity
The Web Service Server entity contains the main logic to publish services. It listens for incoming requests and executes the relevant service operation.

For a entity, the Type property states "WebServiceServer". The XML representation of a SOAP Web Service Server entity is a .4wsprg file.

When a Web Service Server entity is implemented from the diagram, a .4gl source file is generated which contains the MAIN function for the application. This .4gl file is also used to combine the other generated .4gl files into a Genero application.
Any changes to the BA diagram are included in the subsequent re-generating of the program code.

The generated code can be customized.

**Properties**

All properties of a Web Service group of a Record node have a sense for SOAP Web Service only.

All properties of Web Service group of RecordField node except 'public' have a sense for SOAP Web Service only.

The 'public' property on the RecordField node is used by SOAP and JSON web services.

**Table 54: Entity Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of entity.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of entity (WebServiceServer or WebServiceServerJSON)</td>
</tr>
<tr>
<td>File Name</td>
<td>Full path to file.</td>
</tr>
<tr>
<td>Description</td>
<td>Optional description.</td>
</tr>
<tr>
<td>Namespace</td>
<td>(SOAP Web Service Servers only) Common namespace used for the published services.</td>
</tr>
</tbody>
</table>

**Context Menu**

Right-click a Web Service Server entity in the diagram to display a context menu of options. The options displayed depend on the type of Web Service Server and whether it has been implemented.

For a SOAP Web Service Server that has not been implemented:

- Implement SOAP Web Service Server
- Convert to

For a JSON Web Service Server that has not been implemented:

- Implement JSON Web Service Server
- Convert to

For a SOAP and JSON Web Service Server that have been implemented:

- Execute Program / Build Program
- Convert to
- Locate in Project

**Table 55: Context Menu Options**

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement SOAP Web Service Server</td>
<td>Creates the SOAP Web Service Server file (.4wsprg). The .4wsprg generates the logic for the MAIN function. Until you create the SOAP Web Service Server file, the Web Service Server entity on the diagram is simply an icon.</td>
</tr>
<tr>
<td>Implement JSON Web Service Server</td>
<td>Creates the JSON Web Service Server file (.4wsjprg). The .4wsjprg generates the logic for the MAIN function. Until you create the JSON Web Service Server file, the Web Service Server entity on the diagram is simply an icon.</td>
</tr>
<tr>
<td>Execute Program / Build Program</td>
<td>Builds and/or executes the source code for the Web Service Server based on the related entities in the diagram.</td>
</tr>
<tr>
<td>Convert to</td>
<td>Converts entity from one type to another.</td>
</tr>
<tr>
<td>Menu Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Locates and highlights the selected item in the project.</td>
</tr>
</tbody>
</table>

**Related concepts**

*Understanding what gets generated* on page 303

4gl files are generated for each diagram entity.

*Add Web services* on page 250

You can generate SOAP or JSON Web services.

**Create a standalone SOAP Web Service**

The standalone service provides CRUD operations on specified business records (data sets).

There are two options to select from:

- SOAP Web Service (\*.4ws)
- SOAP Web Service from Database (\*.4ws)

1. Create a new Web service.
   - From an open BA diagram, right-click on the background and select **New** from the contextual menu, then select from the types of Web Services.
   - Alternatively, select **File > New > Genero BAM Desktop** and look under the **Sources** section.
2. Right-click on the service to implement it. Create the business records that you want to handle in the services.
3. Save the file in your project. (This project can now be imported into a BA diagram using the **Import Project Files to BA Diagram** menu option.)
4. Functionality properties can be set on each record to specify whether the services operation of add, update, delete and/or search should be generated.

**Create a Web Service from a form**

A service can be created either from a CRUD form or from a Zoom form. From a CRUD form the web service can have Create, Read, Update, and/or Delete operations. From a Zoom form, operations are limited to Read.

1. Open the BA Diagram and select a form.
2. In the properties view, enter the service name into the service name property. Service CRUD operations apply on the records (data sets) that are already defined in the form. A new data set is not necessary, but the data set can be customized.
3. Compile the form. The service is created during form compilation.

**Related concepts**

*Business records (data sets)* on page 493

Business records model the data definition, structure and table relationships of the data used in a form, report, and/or web service. Business records are designed and modified in the Business Records diagram.

*Public fields* on page 257

A public field is a field that appears in the service API.

**Create a SOAP Web Service server**

A Web service server is in charge of publishing services. It listens for incoming requests and executes the relevant service operation.

1. Open the BA Diagram, right-click on the background of the diagram and select **New > SOAP Web Service Server**.
2. Specify a common namespace used for published services. This is optional.
3. Draw a relation from the Web Service Server to each Web Service or Form with Web Service entity that are to be connected to the Web Service Server.
   - Service publication is done by creating the relations between the server entity and the form with service or service standalone entities.
4. Select the Web Service Server and select **Implement SOAP Web Service Server**. Save and name the server in your project.

**Related concepts**

- **Understanding what gets generated** on page 303
- 4gl files are generated for each diagram entity.

**Public fields**

A public field is a field that appears in the service API.

Only public fields are treated in the service CRUD operations. By default all fields in a data set are public.

**Create the client for a SOAP Web Service**

You are provided with a WSDL. You can generate it from Project Manager with the help of a wizard.

1. Right-click on the Application node.
2. Select **Add Web Service > Client consuming WSDL**.
3. Complete the **Web services selection** dialog.

See **The Web Services wizard** on page 1076 for more information.

**JSON Web services**

You can generate JSON Web services.

JSON Web Services are modeled with:

- JSON Web Service Server
- JSON Web Service
- CRUD Form + JSON Web Service
- Zoom Form + JSON Web Service

The exchange protocol is HTTP/REST with JSON encoded messages.

For JSON Services, both the server implementation and a client (consumer) API are generated from the model.

In order to write a client application, link the generated client stub with the service consumer application and the files `libdbappWSCore.42m` and `libdbappWSClient.42m` of the dbapp library. See **Creating a client for a JSON Web service** on page 263.

For more information regarding JSON (or RESTful) Web services, refer to the *Genero Business Development Language User Guide*.

**JSON Web Service entity**

A Web Service entity generates a web service with its CRUD operations, but with no accessible form. CRUD operations are used to Create, Read, Update, and Delete data from a database.

It includes one global *read* operation that reads all records in once and one global *create* operation that creates all records at once.

For a **JSON Web Service** entity, the Type property states "WebServiceJSON". The XML representation of a JSON Web Service entity (the Web service definition file) is a `.4wsj` file.

The Web Service entity can be opened and edited in the Form Designer Records tab. When a Web Service entity is implemented from the diagram, both the XML file (.4ws or .4wsj) and the 4gl source files containing the program logic to create and set up a web service and its CRUD operations are created. Any changes to the Web Service entity properties in the BA diagram or in the XML file (.4ws or .4wsj) are included in the subsequent re-generating of the program code. The generated code can be customized.

See the topic **Introduction to Web Services** in the *Genero Business Development Language User Guide* for more information on Web Services concepts.

**Properties of the Web Service entity**

Select the Web Service entity in the Business Application diagram to view and set the entity properties.
Table 56: Web Service entity Properties, Object Category

<table>
<thead>
<tr>
<th>Group</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Name</td>
<td>Name of entity.</td>
</tr>
<tr>
<td></td>
<td>Type</td>
<td>Type of entity:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• WebService (for a SOAP Web Service).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• WebServiceJSON (for a JSON Web Service).</td>
</tr>
<tr>
<td></td>
<td>File Name</td>
<td>Full path to file.</td>
</tr>
<tr>
<td>Web Service</td>
<td>Description</td>
<td>Description of web service. Optional.</td>
</tr>
<tr>
<td></td>
<td>Service Name</td>
<td>Unique publishable service name.</td>
</tr>
<tr>
<td></td>
<td>Comment</td>
<td>Service comment that will appear in the wsdl file.</td>
</tr>
</tbody>
</table>

Contextual menu for the Web Service entity

Right-click a Web Service entity in the diagram to display a context menu of options.

Table 57: Web Service entity context menu options (partial listing)

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement Web Service</td>
<td>Creates a new blank managed Web service definition file to design from scratch.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Once created, this menu option disappears; use Open for any future modifications.</td>
</tr>
<tr>
<td>Implement Web Service from Database</td>
<td>Provides a wizard to create a managed Web service definition file, allowing you to pick the columns to use in the CRUD operations. This option is generally preferred. The Web service can be modified after it is created.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Once created, this menu option disappears; use Open for any future modifications.</td>
</tr>
<tr>
<td>Open</td>
<td>Open the Business Record for the selected Web Service entity.</td>
</tr>
<tr>
<td>Convert to</td>
<td>Converts entity from one type to another.</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Locates and highlights the selected item in the project.</td>
</tr>
</tbody>
</table>

Web Service entity Business Record

Once the Web service definition file (.4ws or .4wj) exists, right-click the Web Service entity and select **Open** to view the Business Record diagram.

In the Structure view, select the **Root** node. All Web services share the same set of properties in the Root node.

**Object group**

name, databaseName

In the Structure view, select a **Record** node. All Web services share the same Record node properties for the following groups:

**Object group**

name

**Modeling group**

active, masterTable, unique key

**Query group**

query

**Functionality group**

canDisplay, canAdd, canModify, canDelete, canSearch

For the same Record node, however, only SOAP Web services contain the properties that provide for XML and XSD Schema Serialization attributes:
Web Service group
XMLAll, XMLSequence, XMLList,
XMLElementNamespace, XMLAttributeNamespace,
XSTypename, XSTypenamespace

See the section on XML serialization in the Genero Business Development Language User Guide for more information.

In the Structure view, select a RecordField node. All Web services share the same RecordField node properties for the following groups:

Object group
name

Modeling group
lookup

Field group
fieldType, sqlTabname, colName, fieldIdRef, dataType

Web Service group
public

Miscellaneous group
defaultValue

For the same RecordField node, however, only SOAP Web services contain the properties that provide for XML and XSD Schema Serialization attributes:

Web Service group
XMLOptional, XMLElement, XMLAttribute,
XMLName, XSDType, XSDLength, XSDMinLength,
XSDMaxLength, XSDEnumeration, XSDWhiteSpace,
XSDPattern, XSDMinInclusive, XSDMaxInclusive,
XSDMinExclusive, XSDMaxExclusive,
XSDTotalDigits, XSFractionDigits

See the section on XML serialization in the Genero Business Development Language User Guide for more information.

Related concepts
Add Web services on page 250
You can generate SOAP or JSON Web services.

**CRUD form with JSON Web Service entity**
CRUD forms with web services allow programmable access to a form's functionality through a published service.

A CRUD form and Web Service entity generates the code for a CRUD form (a GUI doing create, read, update and delete functionality) and the code for a single standalone Web Service. There is no relation between the form and standalone Web service, however you can use the same object in two different situations: to display a CRUD form and to define a Web Service.

For a CRUD form with JSON Web Service, the Type property states "FormWebServiceJSON". The XML representation of a CRUD form with JSON Web Service entity is a .4fdmwsj file.

When you implement a CRUD form with Web Service entity, you have the option to:

- Implement Form and Web Service
- Implement Form and Web Service from Database

The object files can be opened and edited in Form Designer. In addition to the properties of a CRUD Form entity on page 237, additional properties are found under the heading Web Service.

**Table 58: Web Service properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service name</td>
<td>Unique publish-able service name.</td>
</tr>
<tr>
<td>Comment</td>
<td>Service comment that will appear in the wsd1 file.</td>
</tr>
</tbody>
</table>
Right-click the entity in the diagram to display a context menu of options.

### Table 59: Context Menu Options (partial list)

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement CRUD Form and Web Service</td>
<td>Creates a new blank managed web service definition file to design from scratch.</td>
</tr>
<tr>
<td>Implement CRUD Form and Web Service from Database</td>
<td>Provides a wizard to create a managed web service definition file, allowing you to pick the columns to use in the CRUD operations. This option is generally preferred. The web service can be modified after it is created.</td>
</tr>
<tr>
<td>Convert to</td>
<td>Converts entity from one type to another.</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Locates and highlights the selected item in the project.</td>
</tr>
</tbody>
</table>

### Related concepts

**Form behavior in CRUD states** on page 293

The **UI Settings** properties specify the initial and default state of the selected form as well as the behavior of the form during each state (DISPLAY, MODIFY, ADD, and SEARCH). These settings can be set on the form entity in the BA diagram, but overwritten by a relation to the form.

**Zoom form with JSON Web Service entity**

Zoom forms with web services allow programmable access to a zoom form's functionality through a published service.

A **Zoom form and Web Service** entity generates the code for a Zoom form (a GUI supporting read operations only) and the code for a single standalone Web Service. There is no relation between the form and standalone Web service, however you can use the same object in two different situations: to display a read-only form and to define a Web Service.

**Note:** Only Read operation is generated for a Zoom form and Web Service entity.

For a **Zoom form with JSON Web Service** entity, the Type property states “ZoomWebServiceJSON”. The XML representation of a Zoom form with JSON Web Service entity is a `.4fdzwsj` file.

When you implement a Zoom form with Web Service entity, you have the option to:

- Implement Form and Web Service
- Implement Form and Web Service from Database

The object files can be opened and edited in Form Designer. In addition to the properties of a **Zoom form entity** on page 239, additional properties are found under the heading **Web Service**.

### Table 60: Web Service properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service name</td>
<td>Unique publish-able service name.</td>
</tr>
<tr>
<td>Comment</td>
<td>Service comment that will appear in the wsdl file.</td>
</tr>
</tbody>
</table>

Right-click the entity in the diagram to display a context menu of options.

### Table 61: Context Menu Options (partial list)

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement Zoom Web Service</td>
<td>Creates a new blank managed web service definition file to design from scratch.</td>
</tr>
</tbody>
</table>
Menu Option | Description
--- | ---
Implement Zoom Web Service from Database | Provides a wizard to create a managed web service definition file, allowing you to pick the columns to use in the read operations. This option is generally preferred. The web service can be modified after it is created.
Convert to | Converts entity from one type to another.
Locate in Project | Locates and highlights the selected item in the project.

**JSON Web Service Server entity**
The Web Service Server entity contains the main logic to publish services. It listens for incoming requests and executes the relevant service operation.

For a entity, the Type property states "WebServiceServerJSON". The XML representation of a JSON Web Service Server entity is a .4wsjprg file.

When a Web Service Server entity is implemented from the diagram, a .4gl source file is generated which contains the MAIN function for the application. This .4gl file is also used to combine the other generated .4gl files into a Genero application.

Any changes to the BA diagram are included in the subsequent re-generating of the program code.

The generated code can be customized.

**Properties**
All properties of a Web Service group of a Record node have a sense for SOAP Web Service only.

All properties of Web Service group of RecordField node except 'public' have a sense for SOAP Web Service only.

The 'public' property on the RecordField node is used by SOAP and JSON web services.

**Table 62: Entity Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of entity.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of entity (WebServiceServer or WebServiceServerJSON)</td>
</tr>
<tr>
<td>File Name</td>
<td>Full path to file.</td>
</tr>
<tr>
<td>Description</td>
<td>Optional description.</td>
</tr>
<tr>
<td>Namespace</td>
<td>(SOAP Web Service Servers only) Common namespace used for the published services.</td>
</tr>
</tbody>
</table>

**Context Menu**
Right-click a Web Service Server entity in the diagram to display a context menu of options. The options displayed depend on the type of Web Service Server and whether it has been implemented.

For a SOAP Web Service Server that has not been implemented:
- Implement SOAP Web Service Server
- Convert to

For a JSON Web Service Server that has not been implemented:
- Implement JSON Web Service Server
- Convert to

For a SOAP and JSON Web Service Server that have been implemented:
- Execute Program / Build Program
• Convert to
• Locate in Project

Table 63: Context Menu Options

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implement SOAP Web Service Server</td>
<td>Creates the SOAP Web Service Server file (.4wsprg). The .4wsprg generates the logic for the MAIN function. Until you create the SOAP Web Service Server file, the Web Service Server entity on the diagram is simply an icon.</td>
</tr>
<tr>
<td>Implement JSON Web Service Server</td>
<td>Creates the JSON Web Service Server file (.4wsjprg). The .4wsjprg generates the logic for the MAIN function. Until you create the JSON Web Service Server file, the Web Service Server entity on the diagram is simply an icon.</td>
</tr>
<tr>
<td>Execute Program / Build Program</td>
<td>Builds and/or executes the source code for the Web Service Server based on the related entities in the diagram.</td>
</tr>
<tr>
<td>Convert to</td>
<td>Converts entity from one type to another.</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Locates and highlights the selected item in the project.</td>
</tr>
</tbody>
</table>

Related concepts

Understanding what gets generated on page 303
4gl files are generated for each diagram entity.

Add Web services on page 250
You can generate SOAP or JSON Web services.

Create a standalone JSON Web Service
The standalone service provides CRUD operations on specified business records (data sets).

There are two options to select from:
• JSON Web Service (.4wsj)
• JSON Web Service from Database (.4wsj)

1. Create a new Web service.
   • From an open BA diagram, right-click on the background and select New from the contextual menu, then select from the types of Web Services.
   • Alternatively, select File > New > Genero BAM Desktop and look under the Sources section.
2. Right-click on the service to implement it. Create the business records that you want to handle in the services.
3. Save the file in your project. (This project can now be imported into a BA diagram using the Import Project Files to BA Diagram menu option.)
4. Functionality properties can be set on each record to specify whether the services operation of add, update, delete and/or search should be generated.

Create a Web Service from a form
A service can be created either from a CRUD form or from a Zoom form. From a CRUD form the web service can have Create, Read, Update, and/or Delete operations. From a Zoom form, operations are limited to Read.

1. Open the BA Diagram and select a form.
2. In the properties view, enter the service name into the service name property. Service CRUD operations apply on the records (data sets) that are already defined in the form. A new data set is not necessary, but the data set can be customized.
3. Compile the form. The service is created during form compilation.

Related concepts

Business records (data sets) on page 493
Business records model the data definition, structure and table relationships of the data used in a form, report, and/or web service. Business records are designed and modified in the Business Records diagram.

Public fields on page 257
A public field is a field that appears in the service API.

Create a JSON Web Service server
A Web service server is in charge of publishing services. It listens for incoming requests and executes the relevant service operation.

1. Open the BA Diagram, right-click on the background of the diagram and select New > JSON Web Service Server.
2. Draw a relation from the Web Service Server to each Web Service or Form with Web Service entity that are to be connected to the Web Service Server.
   Service publication is done by creating the relations between the server entity and the form with service or service standalone entities.
3. Select the Web Service Server and select Implement JSON Web Service Server. Save and name the server in your project.

Related concepts
Understanding what gets generated on page 303
4gl files are generated for each diagram entity.

Public fields
A public field is a field that appears in the service API.

Only public fields are treated in the service CRUD operations. By default all fields in a data set are public.

Creating a client for a JSON Web service
How you approach creating a JSON Web service client depends on whether you are using Genero BDL or another programming language.

The BAM generates a client stub which encapsulates the lower levels of the HTTP REST JSON operations (requests and responses) in the form of the WebService_serverclient.4gl file. This file can be included as an API library for a Genero Web service client application, or can be used as a reference when creating a non-Genero Web service client application. The Web service client itself must be coded by hand.

Note: The Customers web service example used for these topics can be found under the Four Js Genero github repository. See http://github.com/FourjsGenero/ex_bam_json_web_service

• Create a Genero BDL client for a JSON Web service on page 263
• Create a non-Genero client for a JSON Web service on page 266

Create a Genero BDL client for a JSON Web service
Web services generated using the BAM provide an API that you can use to write your Genero BDL Web service client.

When a JSON Web service (WebServiceJSON) is generated, an API is created for your use. BAM names the API by appending ",serviceclient" to the end of the Web service name:

• WebService_serviceclient.4gl

This API provides the methods needed to create a JSON Web service client. Use the methods provided in the API to ask for CRUD operations from the JSON Web service. It must be provided by the Web service producer.

Tip: An example of the server client API usage is provided in the OfficeStore demo; examine the JSONClient.4gl file found in the node named JSONClient.
**Import the libraries**

In your Web services client application, use an IMPORT statement to include the API, as well as the common library libdbappWSClient, as shown in this example:

```plaintext
IMPORT FGL libdbappWSClient
IMPORT FGL JSONSuppliers_serviceclient
```

The common library provides functions necessary for setting the URL to the Web service server.

**Provide the URL for the Web service**

The only URL you need to provide is the URL to the Web service server itself, as shown in this example:

```plaintext
CALL libdbappWSClient.setUrlServer("http://localhost:6394/ws/r/JSONServer")
```

The format of the URL:

http://<server>/<alias>/ws/r/<group>

where:

- `<server>` - Service server name or IP address.
- `<alias>` - fastcgi / http dispatcher configuration
- `<group>` - Web service server, for example, GAS group configuration or external configuration file name

**Understanding the methods**

For this example, start with this example of a Customers Web service, containing three records: Accounts, Orders, and LineItems. The Accounts record is the master.

**Note:** The Customers web service example used for these topics can be found under the Four Js Genero github repository. See [http://github.com/FourjsGenero/ex_bam_json_web_service](http://github.com/FourjsGenero/ex_bam_json_web_service)
From this example, you can see that you have methods that affect the whole document, the Accounts, the Orders, and the LineItems, as shown in this table below:

**Table 64: Client API functions**

<table>
<thead>
<tr>
<th>Function suffix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_createAll()</td>
<td>Creates a collection of documents.</td>
</tr>
<tr>
<td>_readAll()</td>
<td>Reads a collection of documents from a Query by Example (QBE).</td>
</tr>
<tr>
<td>_Accounts_create()</td>
<td>Creates a collection of new accounts.</td>
</tr>
<tr>
<td>_Accounts_readRow()</td>
<td>Reads the account specified by accountID.</td>
</tr>
<tr>
<td>_Accounts_update()</td>
<td>Updates a collection of accounts</td>
</tr>
<tr>
<td>_Accounts_updateRow()</td>
<td>Updates the account specified by accountID.</td>
</tr>
<tr>
<td>_Accounts_delete()</td>
<td>Deletes a collection of accounts</td>
</tr>
<tr>
<td>_Accounts_deleteRow()</td>
<td>Deletes the account specified by accountID.</td>
</tr>
<tr>
<td>_Orders_create()</td>
<td>Creates a collection of new orders.</td>
</tr>
<tr>
<td>_Orders_readRow()</td>
<td>Reads the order specified by orderID.</td>
</tr>
<tr>
<td>_Orders_update()</td>
<td>Updates a collection of orders</td>
</tr>
<tr>
<td>_Orders_updateRow()</td>
<td>Updates the order specified by orderID.</td>
</tr>
<tr>
<td>_Orders_delete()</td>
<td>Deletes a collection of orders</td>
</tr>
<tr>
<td>Function suffix</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>_Orders_deleteRow()</td>
<td>Deletes the line item specified by <code>lineItemID</code>.</td>
</tr>
<tr>
<td>_LineItems_create()</td>
<td>Creates a collection of new line items.</td>
</tr>
<tr>
<td>_LineItems_readRow()</td>
<td>Reads the line item specified by <code>lineItemID</code>.</td>
</tr>
<tr>
<td>_LineItems_update()</td>
<td>Updates a collection of line items</td>
</tr>
<tr>
<td>_LineItems_updateRow()</td>
<td>Updates the line item specified by <code>lineItemID</code>.</td>
</tr>
<tr>
<td>_LineItems_delete()</td>
<td>Deletes a collection of line items</td>
</tr>
<tr>
<td>_LineItems_deleteRow()</td>
<td>Deletes the line item specified by <code>lineItemID</code>.</td>
</tr>
</tbody>
</table>

Create a non-Genero client for a JSON Web service

In order to consume a BAM-generated JSON Web service, a developer must understand the format of the HTTP REST JSON operations (requests and responses).

Before you begin, get the `WebService_serverclient.4gl` file from the Web Service provider. Although written in Genero BDL, this file contains the information needed to create the JSON body for the requests and responses. Consider this file as the documentation of how the Web service works.

From this document and from the information and examples provided below, you should be able to construct the appropriate HTTP REST JSON operations. This means you need to be able to construct:

- The URL of the REST resource.
- The HTTP header, to include the HTTP method, additional REST resource, and a possible CRUD operation.
  - The HTTP method will be either GET, PUT, POST, or DELETE.
  - The resource provides additional URL details that specify which record is involved in the request.
  - Some CRUD operations require you to provide the CRUDOperation in the header, specifically when you read, update or delete a collection.
- The request body and response body. Data is exchanged in JSON format over HTTP. Examine the `WebService_serverclient.4gl` file to identify what is expected for both the request AND the response.
  - The request body provides the data that the Web service needs to do its work. In the `WebService_serverclient.4gl` file, this is managed using a record with the suffix `_IN_type`.
  - The response body provides the client application with the status of the operation (where 200 indicates success) along with any additional data sent back by the service. In the `WebService_serverclient.4gl` file, this is managed using a record with the suffix `_OUT_type`.

URL of the REST resource

The URL of the REST resource takes the following format:

```
http://<server>/<alias>/ws/r/<group>/REST/<resource>[(<IDs>)]
```

where:

- `<server>` - Service server name or IP address.
- `<alias>` - fastcgi / http dispatcher configuration
- `<group>` - Web service server, for example, GAS group configuration or external configuration file name
- `<resource>` - Web service entity
- `<IDs>` (optional) - List of identifiers for records. IDs are enclosed by single quotes and separated by commas.

JSON Service example

Study these examples regarding the URLs and methods for a RESTful Web service. In this example, the JSON Web Service entity (with a Service Name of "Customers") involves three records (Accounts, Orders, and LineItems) defined with master-detail relationships.
Note: The Customers web service example used for these topics can be found under the Four Js Genero github repository. See http://github.com/FourjsGenero/ex_bam_json_web_service

![Diagram of Accounts (Master) and Orders](image)

**Figure 156: Customers Web Service example with master-detail relationships**

**Table 65: This table is describing the REST JSON request and response**

<table>
<thead>
<tr>
<th>Functional description</th>
<th>URL of the REST resource</th>
<th>Scope</th>
<th>HTTP Method</th>
<th>CRUD operation</th>
<th>Request Body</th>
<th>Response Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a list of all customers (i.e., accounts + orders + lineitems)</td>
<td><a href="http://localhost:8090/ws/r/JSONServer/REST/Customers">http://localhost:8090/ws/r/JSONServer/REST/Customers</a></td>
<td>Collection</td>
<td>POST</td>
<td>READCollection</td>
<td>Query By Example</td>
<td>Error number + Collections of customers</td>
</tr>
<tr>
<td>Return the account &quot;dupont&quot;</td>
<td><a href="http://localhost:8090/ws/r/JSONServer/REST/Customers('dupont">http://localhost:8090/ws/r/JSONServer/REST/Customers('dupont</a>')</td>
<td>Item</td>
<td>GET</td>
<td>Primary Key</td>
<td></td>
<td>Error number + Account data for customer dupont</td>
</tr>
<tr>
<td>Return the order 1</td>
<td><a href="http://localhost:8090/ws/r/JSONServer/REST/Customers/Accounts/Orders(1)">http://localhost:8090/ws/r/JSONServer/REST/Customers/Accounts/Orders(1)</a></td>
<td>Item</td>
<td>GET</td>
<td>Primary Key</td>
<td></td>
<td>Error number + Order data for order 1</td>
</tr>
<tr>
<td>Functional description</td>
<td>URL of the REST resource</td>
<td>Scope</td>
<td>HTTP Method</td>
<td>CRUD operation</td>
<td>Request Body</td>
<td>Response Body</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------</td>
<td>-------</td>
<td>-------------</td>
<td>----------------</td>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Delete a list of accounts</td>
<td><a href="http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS/Accounts">http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS/Accounts</a></td>
<td>Collection</td>
<td>POST</td>
<td>DELETE Collection</td>
<td>Array of Keys</td>
<td>Error number</td>
</tr>
<tr>
<td>Delete the account &quot;dupont&quot;</td>
<td><a href="http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS/Accounts('dupont">http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS/Accounts('dupont</a>')</td>
<td>Item</td>
<td>DELETE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update a list of accounts</td>
<td><a href="http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS/Accounts">http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS/Accounts</a></td>
<td>Collection</td>
<td>PUT</td>
<td>UPDATE Collection</td>
<td>Array of Data</td>
<td>Error number</td>
</tr>
<tr>
<td>Update the account &quot;dupont&quot;</td>
<td><a href="http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS/Accounts('dupont">http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS/Accounts('dupont</a>')</td>
<td>Item</td>
<td>PUT</td>
<td></td>
<td>Data</td>
<td>Error number</td>
</tr>
<tr>
<td>Create a document (createAll)</td>
<td><a href="http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS">http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS</a></td>
<td>Collection</td>
<td>POST</td>
<td></td>
<td>Array of Data</td>
<td>Error number + collection of keys</td>
</tr>
<tr>
<td>Create a list of accounts</td>
<td><a href="http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS/Accounts">http://localhost:8090/ws/r/JSONServer/REST/CUSTOMERS/Accounts</a></td>
<td>Collection</td>
<td>POST</td>
<td></td>
<td>Array of Data</td>
<td>Error number + collection of keys</td>
</tr>
</tbody>
</table>

**JSON resources**

The implementation of a JSON Item of the Business Application diagram defines a data set. In a data set, each record defines one resource and provides methods to create, read, update, and delete data.

Table 66: Resources on page 268 shows the resources defined for the JSON item Customers of our example. This item provides three records, Accounts, Orders, and LineItems.

**Table 66: Resources**

<table>
<thead>
<tr>
<th>Resource</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customers</td>
<td>/Customers</td>
</tr>
</tbody>
</table>
Table 67: HTTP requests on page 269 provides examples of the HTTP header information necessary to complete CRUD operations on the resources in our example. In addition, the final three columns would map to the related function, the record used for the request body, and the record used for the response body.

Table 67: HTTP requests

<table>
<thead>
<tr>
<th>HTTP header</th>
<th>Description</th>
<th>Related function</th>
<th>Request record</th>
<th>Response record</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST /Customers</td>
<td>Creates a collection of documents.</td>
<td>Customers_client_createAll()</td>
<td>createAll_IN_type</td>
<td>createAll_OUT_type</td>
</tr>
<tr>
<td>POST /Customers/Accounts</td>
<td>Creates a collection of new accounts.</td>
<td>Customers_client_Accounts_create()</td>
<td>Accounts_create_IN_type</td>
<td>Accounts_create_OUT_type</td>
</tr>
<tr>
<td>GET /Customers/Accounts('accountID')</td>
<td>Reads the account specified by accountID.</td>
<td>Customers_client_Accounts_readRow()</td>
<td>Accounts_ws_br_uk_type</td>
<td>Accounts_read_OUT_type</td>
</tr>
<tr>
<td>PUT /Customers/Accounts</td>
<td>Updates the account specified by accountID.</td>
<td>Customers_client_Accounts_updateRow()</td>
<td>Accounts_ws_br_uk_type</td>
<td>Accounts_update_OUT_type</td>
</tr>
<tr>
<td>POST /Customers/Accounts</td>
<td>Deletes a collection of accounts.</td>
<td>Customers_client_Accounts_delete()</td>
<td>Accounts_delete_IN_type</td>
<td>Accounts_delete_OUT_type</td>
</tr>
<tr>
<td>DELETE /Customers/Accounts('accountID')</td>
<td>Deletes the account specified by accountID.</td>
<td>Customers_client_Accounts_deleteRow()</td>
<td>Accounts_ws_br_uk_type</td>
<td>Accounts_delete_OUT_type</td>
</tr>
<tr>
<td>POST /Customers/Accounts/Orders</td>
<td>Creates a collection of new orders.</td>
<td>Customers_client__Orders_create()</td>
<td>Orders_create_IN_type</td>
<td>Orders_create_OUT_type</td>
</tr>
<tr>
<td>GET /Customers/Accounts/Orders('orderID')</td>
<td>Reads the order specified by orderID.</td>
<td>Customers_client__Orders_readRow()</td>
<td>Orders_ws_br_uk_type</td>
<td>Orders_read_OUT_type</td>
</tr>
<tr>
<td>PUT /Customers/Accounts/Orders</td>
<td>Updates a collection of orders.</td>
<td>Customers_client__Orders_updateRow()</td>
<td>Orders_ws_br_uk_type</td>
<td>Orders_update_OUT_type</td>
</tr>
</tbody>
</table>
HTTP header

<table>
<thead>
<tr>
<th>Description</th>
<th>Related function</th>
<th>Request record</th>
<th>Response record</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT /Customers/Accounts/Orders('orderID')</td>
<td>Update the order specified by orderID.</td>
<td>Customers_client__Customers_updateRow() type</td>
<td>Orders_update_OUT_type</td>
</tr>
<tr>
<td>POST /Customers/Accounts/Orders</td>
<td>CRUDOperation: DELETECollection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deletes a collection of orders</td>
<td>Customers_client__Customers_delete() type</td>
<td>Orders_delete_OUT_type</td>
<td></td>
</tr>
<tr>
<td>DELETE /Customers/Accounts/Orders/LineItems('lineItemID')</td>
<td>Deletes the line item specified by lineItemID.</td>
<td>Customers_client__Customers_deleteRow() type</td>
<td>Orders_delete_OUT_type</td>
</tr>
<tr>
<td>POST /Customers/Accounts/Orders/LineItems</td>
<td>CRUDOperation: CREATECollection</td>
<td>Customers_client__LineItems_create() type</td>
<td>LineItems_create_OUT_type</td>
</tr>
<tr>
<td>GET /Customers/Accounts/Orders/LineItems('lineItemID')</td>
<td>Reads the line item specified by lineItemID.</td>
<td>Customers_client__LineItems_readRow() type</td>
<td>LineItems_read_OUT_type</td>
</tr>
<tr>
<td>PUT /Customers/Accounts/Orders/LineItems</td>
<td>CRUDOperation: UPDATECollection</td>
<td>Customers_client__LineItems_updateRow() type</td>
<td>LineItems_update_OUT_type</td>
</tr>
<tr>
<td>POST /Customers/Accounts/Orders/LineItems</td>
<td>CRUDOperation: DELETECollection</td>
<td>Customers_client__LineItems_deleteRow() type</td>
<td>Orders_delete_OUT_type</td>
</tr>
</tbody>
</table>

Add Relations

Define a relationship from one entity to another with a relation entity.

1. Right-click on an entity, such as a Program, and select New Relation.
2. Click and drag the Relation arrow to the entity to which you want to set the relationship, such as a Form. If you try to set an invalid relation, for example from a Form entity to a Program, you will see the constraint icon indicating that this relation is not valid.
3. Repeat this step until all relationships have been defined.
Relations

A Relation entity is used to define a relationship between entities on the BA diagram. Relations are represented with arrows on the BA diagram.

There are specific types of relations:

- **Relation**: Relation between Program and Form entities and between Form entities. A form can have multiple relations to another form, thus allowing the same form to behave differently depending on how it is used in the application. See UI Settings.
- **ReportRelation**: Relation between Form and Report entities.
- **WebServiceRelation**: Relation between Web Service Server and Web Service entities.
- **PhotoRelation**: Relation to Photo entity in mobile app diagram.
- **GalleryRelation**: Relation to Gallery entity in mobile app diagram.
- **PhoneRelation**: Relation to Phone entity in mobile app diagram.
- **SMSRelation**: Relation to SMS entity in mobile app diagram.
- **ContactRelation**: Relation to Contact entity in mobile app diagram.
- **MapsRelation**: Relation to Maps entity in mobile app diagram.
- **BarcodeRelation**: Relation to Barcode entity in mobile app diagram.

**Note**: See Mobile device function properties on page 273 for properties specific to mobile app entity relations.

Table 68: Entity Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of relation.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of relation.</td>
</tr>
<tr>
<td>Source</td>
<td>Relation source entity name.</td>
</tr>
<tr>
<td>Destination</td>
<td>Relation destination entity name.</td>
</tr>
<tr>
<td>Action</td>
<td>Action name to be used in generated code to trigger relationship. For example, &quot;zoom1&quot; might be an action name between a Form and a Zoom Form entity.</td>
</tr>
<tr>
<td>Source Record</td>
<td>If an action is desired only for a specific subdialog, this action is set to a specific record in the form. When empty, the action defined on the relation is global to the DIALOG except if Row Bound property is checked.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Row Bound</td>
<td>Renders a contextual action to the current row. See Rowbound actions on page 289. If a form has more than one record, Row Bound must be set to one of the records.</td>
</tr>
<tr>
<td>Source Field (Position)</td>
<td>Specifies the fields used to locate a specific row when opening a Form. They usually correspond to database Foreign Keys. They must match in number and type the Business Record unique key of the target Form. See Control the row position in form on page 296.</td>
</tr>
<tr>
<td>Source Field (Filter)</td>
<td>Specifies the fields whose values are used for filtering when opening a Form. See Opening a form with a subset of data on page 296.</td>
</tr>
<tr>
<td>Destination Field (Filter)</td>
<td>Specifies the fields to filter on when opening a Form. See Opening a form with a subset of data on page 296.</td>
</tr>
<tr>
<td>Open Mode</td>
<td>Open Mode is the initial state of the form when opened. The rendered form's default Toolbar allows the user to switch modes.</td>
</tr>
<tr>
<td>Default Mode</td>
<td>Default Mode is the mode in which you return after leaving another mode.</td>
</tr>
<tr>
<td>Data Refresh</td>
<td>If the source form is in DISPLAY mode and opens a destination form, when the destination form closes, the data displayed in the source form refreshes. See Data refresh on page 295.</td>
</tr>
<tr>
<td>Functionality</td>
<td>Change the form's behavior in this relation. See Form behavior in CRUD states on page 293.</td>
</tr>
<tr>
<td>Target Behavior</td>
<td>Change the form's behavior in this relation. See Form behavior in CRUD states on page 293.</td>
</tr>
<tr>
<td>Report File</td>
<td>ReportRelation type relations only. Names the Report Design Document (4rp) to be used for the report. If blank, user can use Select Fields option on Print Report Settings dialog to choose fields for report output.</td>
</tr>
<tr>
<td>Report Label</td>
<td>Name of report shown in print list.</td>
</tr>
</tbody>
</table>

Right-click the Program entity in the diagram to display a context menu of options.

**Table 69: Context Menu Options**

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter Items ...</td>
<td>The Filter View dialog allows you to hide and show items on a diagram.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the entity.</td>
</tr>
</tbody>
</table>

**Add mobile device features (Photo, Gallery, Phone, Mail, SMS, Contact, Maps, Barcode)**

The default template for modeling mobile apps includes features for interacting with a mobile device's default apps such as phone, email, text, photos, and maps.

The process for adding mobile device features to a BA diagram is the same for each mobile device feature, but each feature has specific properties to set on the relation to generate the appropriate code.

This task assumes that you have created a BA diagram with a CRUD or Zoom Form entity.

1. Right-click on the BA diagram and select New. Select a mobile device feature to add to your form.

<table>
<thead>
<tr>
<th>Mobile feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photo</td>
<td>Lets the user take a picture with the mobile device and returns the corresponding picture identifier.</td>
</tr>
<tr>
<td>Gallery</td>
<td>Lets the user select a picture from the mobile device's photo gallery and returns a picture identifier.</td>
</tr>
<tr>
<td>Mobile feature</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Phone</td>
<td>Calls the selected telephone number.</td>
</tr>
<tr>
<td>Mail</td>
<td>Invokes the user's default mail application for a new mail to send.</td>
</tr>
<tr>
<td>SMS</td>
<td>Sends an SMS text to one or more phone numbers.</td>
</tr>
<tr>
<td>Contact</td>
<td>Lets the user choose a contact from the mobile device contact list and returns the vCard.</td>
</tr>
<tr>
<td>Maps</td>
<td>Invokes a maps app with the current Global Positioning System (GPS) location of the mobile device.</td>
</tr>
<tr>
<td>Barcode</td>
<td>Lets the user to scan a barcode with a mobile device and returns the string representation of the barcode and its type.</td>
</tr>
</tbody>
</table>

2. Right-click the form icon and select **New Relation**. Click the Form icon again and drag the relation arrow to the new mobile device feature icon (such as **Photo**).

3. Select the relation.

4. Set the **action** property. The **action** property on the relation must match the **action** property on the form widget used to invoke the feature. Set any other properties specific to the mobile device feature. See **Mobile device function properties** on page 273.

5. Build and run your program.

**Mobile device function properties**
Relation properties specific to mobile device feature entities.

**Note:** For all mobile feature relations, the **Action** property value on the relation must match the **Action** property value on the form widget used to invoke the feature.

**Photo / PhotoRelation**
Lets the user take a picture with the mobile device and returns the corresponding picture identifier.

Table 70: PhotoRelation properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Path</td>
<td>Column to store the picture identifier.</td>
</tr>
</tbody>
</table>

**Gallery / GalleryRelation**
Lets the user select a picture from the mobile device's photo gallery and returns a picture identifier.

Table 71: GalleryRelation properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Path</td>
<td>Column to store the picture identifier.</td>
</tr>
</tbody>
</table>
Phone / PhoneRelation
Calls the selected telephone number.

Table 72: PhoneRelation properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone Number</td>
<td>Column corresponding to the telephone number.</td>
</tr>
</tbody>
</table>

Mail / MailRelation
Invokes the user's default mail application for a new mail to send.

Table 73: MailRelation properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>To</td>
<td>Columns corresponding to the To line of the email.</td>
</tr>
<tr>
<td>CC</td>
<td>Columns corresponding to the CC line of the email.</td>
</tr>
<tr>
<td>Bcc</td>
<td>Columns corresponding to the Bcc line of the email.</td>
</tr>
<tr>
<td>Subject</td>
<td>Column corresponding to the subject line of the email.</td>
</tr>
<tr>
<td>Content</td>
<td>Column corresponding to the body of the email.</td>
</tr>
<tr>
<td>Attachment</td>
<td>Columns corresponding to email attachments. Limit 2.</td>
</tr>
</tbody>
</table>

SMS / SMSRelation
Sends an SMS text to one or more phone numbers.

Table 74: SMSRelation properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>To</td>
<td>Column corresponding to the SMS number.</td>
</tr>
<tr>
<td>Content</td>
<td>Column corresponding to the body of the text.</td>
</tr>
</tbody>
</table>

Contact / ContactRelation
Lets the user choose a contact from the mobile device contact list and returns the vCard.

Table 75: ContactRelation properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vcard</td>
<td>Column corresponding to the contact.</td>
</tr>
<tr>
<td>Person</td>
<td>Columns corresponding to the contact.</td>
</tr>
<tr>
<td>Address Work</td>
<td>Columns corresponding to the contact's work address.</td>
</tr>
<tr>
<td>Address Home</td>
<td>Columns corresponding to the contact's home address.</td>
</tr>
<tr>
<td>Phone</td>
<td>Columns corresponding to the contact's phone numbers.</td>
</tr>
<tr>
<td>Email</td>
<td>Columns corresponding to the contact's email addresses.</td>
</tr>
</tbody>
</table>
Barcode / BarcodeRelation

Lets the user to scan a barcode with a mobile device and returns the string representation of the barcode and its type.

<table>
<thead>
<tr>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barcode</td>
<td>Column corresponding to the barcode string value.</td>
</tr>
</tbody>
</table>

Map / MapRelation

Passes latitude and longitude values to the mobile device map utility.

<table>
<thead>
<tr>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>Columns corresponding to the latitude and longitude values.</td>
</tr>
</tbody>
</table>

Import files into the diagram from the project

If there are files in the project structure that are not in the Business Application diagram, they can be imported.

Right-click on the background of the diagram, and select Import Project Files to BA Diagram.

![Figure 157: Import project files to Business Application diagram dialog](image)

Files that can be imported are listed in the dialog. Select a file, and choose an option from the integrated Toolbar or right-click context menu.

- **Ignore**: Do not add selected files to the diagram.
- **Process later**: Decide later if the selected files should be added to the diagram.
- **Add**: Add selected files to the diagram.
Resolve conflict

Resolve an existing conflict.

Modeling the database

Model the database by creating a meta-schema or extracting one from an existing database.

Warning: The Build command does not automatically rebuild the entire project for each modification of the meta-schema file (.4db, .4dbx). It is the responsibility of the developer to recompile the appropriate parts of the project or to use the Rebuild all command.

• Database meta-schema (4dbx) on page 276
• Create a meta-schema on page 277
• Extract meta-schema information from database on page 277
• Add a meta-schema to a project on page 281
• Managing SERIALs in a generated application on page 283
• Managing concurrency on page 284
• Cascade delete on page 284

Database meta-schema (4dbx)

The 4dbx file is the database schema file for generated applications.

The 4dbx file contains the information about the tables, columns, and relations of the relational database that is needed by your application. It is used to create items in your Business Application diagram (4ba).

The 4dbx file handles CRUD operations for the generated application. For each table with a primary key in the database schema, the application generator creates its database CRUD operations (respectively INSERT, SELECT, UPDATE, and DELETE database statements).

You must have access set up for the database that the application will use.

Constraints management

The Application Generator creates functions to manage the database schema constraints. These constraints are checked during the CRUD operations for each table:

• uniqueness check for its primary key (only if a primary key exists)
• uniqueness check for each unique constraint (only if a primary key exists)
• for one column, the not null constraint is checked
• foreign key existence is checked

SERIAL management

Primary keys you define as SERIAL are managed by the BAM using a table named seqreg in the meta-schema (4dbx). BAM uses this table to provide a sequence of unique integer numbers for SERIAL columns keys.

Related concepts

Managing SERIALs in a generated application on page 283
Primary keys you define as SERIAL are managed by the BAM using a table named seqreg in the meta-schema (4dbx). BAM uses this table to provide a sequence of unique integer numbers for SERIAL columns keys.

Add constraints or indexes on page 350
The Business Application Modeler requires that tables are defined with primary keys. Constraints and indexes that are part of the database structure are displayed as part of the table, but additional constraints (such as primary keys) can be added.

Modeling the database on page 276
Model the database by creating a meta-schema or extracting one from an existing database.

Create a meta-schema

Create a database meta-schema file.

1. Select File > New.
   The New dialog opens.
2. Select a category from the left-side panel.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genero BAM Desktop</td>
<td>Select this option if you are creating a database meta-schema for use in a Business Application Modeling project for a desktop application. The file created will be a 4dbx.</td>
</tr>
<tr>
<td>Genero BAM Mobile</td>
<td>Select this option if you are creating a database meta-schema for use in a Business Application Modeling project for a mobile application. The file created will be a 4dbx.</td>
</tr>
<tr>
<td>Genero</td>
<td>Select this option if you are creating a database meta-schema for use in a standard Genero program. The file created will be a 4db.</td>
</tr>
</tbody>
</table>

3. Select an option from the Database section from the right-side panel.

<table>
<thead>
<tr>
<th>Database option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Schema</td>
<td>Creates a new meta-schema file.</td>
</tr>
<tr>
<td>DB Schema from Database</td>
<td>Opens the New Meta-schema dialog to which you enter your database and connection information. See Extract meta-schema information from database on page 277.</td>
</tr>
</tbody>
</table>

The meta-schema file displays in the document view. A new meta-schema file is automatically added to the list in the DB Schemas Tab if you have saved the file in the current project.

Extract meta-schema information from database

The New Meta-schema dialog assists in extracting schema information from a database.

To extract a database meta-schema file from a database, you must have access and permissions for the database. If you have trouble connecting to a database, make sure the database and the corresponding database client software are installed and configured properly.

When you extract the meta-schema information from the database, you overwrite the existing schema. Any user changes that had been made to the schema are lost when using the extract schema option. If you wish to keep user changes, you must update the schema. See Update a meta-schema from database on page 359.

1. Select Database>>Extract Schema. The first step is specifying the name and location of the meta-schema file.
Figure 158: New Schema dialog

**Meta-schema file path**

Enter the name and path for the new database meta-schema file. Specify a `.4db` file extension for standard projects, or a `.4dbx` file extension if you are working with a Business Application Modeling managed project.

You can either click on the browse button to browse for a location on disk, or you can enter the full path and file name directly in the **Meta-schema file path** field. If you browse for the location, you select the file type in the **Save as type** combobox when saving the file. If you enter the file name directly, you must ensure that you specify the correct file extension (`.4db` or `.4dbx`).

**Insert the file in the project**

Check this box to add the meta-schema file in the project. Select the node where the file should be added.

2. Click the **Next** button to continue to **Connection information**. This connection information is only used to extract the information for the database meta-schema file from the referenced database.
Figure 159: Connection information

a) In the **Database Connection Information** section, select either **Use explicit settings** or **Use external settings**.

**Use explicit settings, previous connection**

You can use a previous connection that was created for the same database. The drop down list provides a list of the existing connections.

**Use explicit settings, database type**

You can enter the **Database Type** by selecting the desired type from the drop down list, and the corresponding information for that type. The **Database driver** for the database type is automatically entered. If other drivers exist, they are available in the drop down list.

**Use external settings**

Information in the FGLPROFILE configuration file is used to extract the corresponding connection information for the specified database. Genero Studio will use the schema name that you entered to check for any related entry in the FGLPROFILE configuration file, and will use those values to define the connection. See information on the FGLPROFILE file in the *Genero Business Development Language User Guide*. 
b) In the **Database User Information** section, provide the necessary database user details. The required information varies based on the database type selected. See **Database server/user information** on page 365.

c) Click **Test Connection** to verify that the information is correct and that you are able to access the database.

3. Click the **Next** button to continue to **Extraction Options**. Select the options for the meta-schema file.

![Figure 160: Extraction options](image)

- **Case sensitivity**
  - Specify how case in database object names should be handled. **Case sensitive**: case won't be changed on database objects. **Lower case**: database object names will be converted to lower case. **Upper case**: database object names will be converted to upper case.

- **Import system tables**
  - Check this box to include system tables in the schema.

- **Ignore errors**
  - Specify that conversion errors should be ignored. If this option is unchecked, the extraction will stop as soon as an error occurs (for example, if a table column has an unsupported type.)

- **Conversion method**
  - Select the type of conversion you wish for the specific data types; the default choice is Type A.

4. Click **Finish** to begin the extraction process.
If you didn't already do so, save the database meta-schema file in a node in the project; the database meta-schema will be added to the DB Schemas tab and made available to other modules.

Any application that uses the meta-schema file must have a dependency to the node where the meta-schema file was added. See Add a meta-schema to a project on page 281.

**Add a meta-schema to a project**

You may have to add the meta-schema file to a project.

If you used File > New to create your project, the default structure of your project includes nodes for a **project**, **application**, **library**, and **databases**; the dependencies between the default nodes has been predefined. When you save your Meta-schema file in the Databases node of the project, the dependency for the application node in the project already exists.

However, if you created your own project structure, you must follow these steps.

1. Open the project.
2. Right-click on the application or library node in the Project view to which you want to add the meta-schema file and select **Add Files**. Locate and add the **meta-schema file**.
3. Add a dependency to the Meta-schema file for any application or library nodes. Right-click the node and select **Advanced Properties, Dependencies**. Check the box for the node containing the Meta-schema file.

**Rename the database in a BAM Mobile project**

The BAM Mobile Project includes a SQLite database named *mydatabase.db*. Follow these steps to rename it.

1. Create a BAM Mobile project.
   Select File > New > Genero BAM Mobile > BAM Mobile Project (.4pw). Provide a project name and location and click **OK**.
   A BAM Mobile Project is created. It includes a **Project** group that includes the following:
   - a **Config** library node containing the *fglprofile* file; and
   - a **Database** library node with two files: a database file (*mydatabase.db*) and a database meta-schema file (*mydatabase.4dbx*)

   It also includes a **DatabaseMaintenance** group containing a **CreateDatabase** application node, which contains the default database creation script *mydatabase_sqt_x.4gl*.

   The **DBPATH** environment variable is set for the project, specifying the location on disk where the default database file can be found: `$(ProjectDir)\database;$(DBPATH)`

2. Rename the database and database meta-schema files to a name of your choosing.
   Under the **database** node, right-click on the file name and select **Rename**. Provide the new database name. Ensure you keep the proper file extension during the renaming process.

   **Important:** When renaming, use lower case letters.

   For example, to create a database named "test", rename the files *test.db* and *test.4dbx*.

3. Update *fglprofile* (located in the **config** node) to reference your new database name.
   For our example, this involves replacing "mydatabase" with "test":
   ```
   dbi.default.driver = "dbmsqt"
   dbi.database.test.source = "test.db"
   ```

   Save your changes.

4. In the database meta-schema file (*.4dbx*), use the context menu to model the database by adding tables and constraints and save your changes.

   You can skip this step to create an empty database.

5. Generate the create database script.
Right-click on the database meta-schema file name and select **Generate Database Creation Script**. At a minimum, select **SQLite** as the target database; select or de-select other options as desired.

6. Save the generated script in the `${ProjectDir}/databaseMaintenance` directory, and insert the file into the project in the **CreateDatabase** application node.
   
   At this time, you can also delete the `mydatabase_sqt_x.4gl` file, as it is no longer needed. To delete the file, right-click on the file name and select **Delete from Disk**. This option not only removes it from the project, but also removes it from the file system.

7. Execute the **CreateDatabase** application.
   
   Right-click the **CreateDatabase** application node and select **Execute**.
   
   The database is created. The status of the operation displays in the **Output** view.

8. Use **DBExplorer** to verify that the database was created properly.
   
   This step involves connecting to your new database and executing a query. For example, if you created a table named "account", you could enter the SQL statement:
   
   ```sql
   SELECT * FROM account;
   ```

**Related concepts**

**DB Explorer** on page 392

The DB Explorer plug-in is a tool that allows you to view, create and modify data stored in a relational database.

**Related tasks**

**Model the database** on page 216

From the database meta-schema file, you can graphically model a database by adding tables and constraints and generate a database creation script.

**Rename the database in a BAM Desktop project**

The BAM Desktop Project includes a meta-schema file named `mydatabase.4dbx`. Follow these steps to rename the meta-schema file and create a database using the same name.

1. Create a BAM Desktop project.
   
   Select **File > New > Genero BAM Desktop > BAM Desktop Project (.4pw)**. Provide a project name and location and click **OK**.
   
   A BAM Desktop Project is created. It includes a **Project** group that includes the following:

   - a **Database** library node with one file: a database meta-schema file

   It also includes a **DatabaseMaintenance** group containing a **CreateDatabase** application node.

   The `DBPATH` environment variable is set for the project, specifying the location on disk where the default database file can be found: `${ProjectDir}\database;${DBPATH}`

2. Rename the database meta-schema files to a name of your choosing.
   
   Under the **database** node, right-click on the file name and select **Rename**. Provide the new meta-schema file name. Ensure you keep the proper file extension during the renaming process.

   **Important**: When renaming, use lower case letters.
   
   For example, to create a database named "test", rename the meta-schema file `test.4dbx`.

3. If necessary, create an empty database file.
   
   With a desktop project, you are not provided with an empty database file; however it is required when creating databases such as SQLite. For example, for SQLite you would create an empty file named `test.db`.

4. Create an **fglprofile** to include the details needed to connect to your database.
   
   For example, for our SQLite example, you would create an fglprofile entry with the following lines:

   ```ini
   dbi.default.driver = "dbmsqt"
   ```
Save the file. A recommendation is to save the file in the config node of the project, and to $(ProjectDir)/config on disk.

Update the FGLPROFILE environment set to point to this fglprofile.

5. In the database meta-schema file (.4dbx), use the context menu to model the database by adding tables and constraints and save your changes.

You can skip this step to create an empty database.

6. Generate the create database script.

Right-click on the database meta-schema file name and select Generate Database Creation Script.

For this example, at a minimum, select SQLite as the target database; select or de-select other options as desired.

7. Save the generated script in the $(ProjectDir)/databaseMaintenance directory, and insert the file into the project in the CreateDatabase application node.

8. Execute the CreateDatabase application.

Right-click the CreateDatabase application node and select Execute.

The database is created. The status of the operation displays in the Output view.

9. Use DBExplorer to verify that the database was created properly.

This step involves connecting to your new database and executing a query. For example, if you created a table named "account", you could enter the SQL statement:

```
SELECT * FROM account;
```

**Related concepts**

- **DB Explorer** on page 392
  The DB Explorer plug-in is a tool that allows you to view, create and modify data stored in a relational database.

**Related tasks**

- **Model the database** on page 216
  From the database meta-schema file, you can graphically model a database by adding tables and constraints and generate a database creation script.

**Managing SERIALs in a generated application**

Primary keys you define as SERIAL are managed by the BAM using a table named seqreg in the meta-schema (.4dbx). BAM uses this table to provide a sequence of unique integer numbers for SERIAL columns keys.

**The seqreg table**

For generated applications, BAM automatically adds a seqreg table to a new meta-schema (.4dbx). It contains two columns; one with the name of the table having a SERIAL column and one with the last SERIAL column value for that table. Thus one seqreg table facilitates a sequence number management for all tables with SERIAL columns in the database.

When a new row is inserted in a table, the BAM provides the primary key value by auto-incrementing the last serial column value entered for the table.

**No configuration required**

Once you have created a table with a SERIAL defined as the primary key, BAM manages sequence number generation and generates all code necessary to handle inserting the new sequence number during an Add operation. No additional configuration or modeling is required.

You may initialize the seqreg table with the appropriate records otherwise SERIAL columns will start at number 1.

If the database schema does not contain SERIAL columns, the seqreg table is not required in the schema.
Informix databases

The `seqreg` table is required in a schema containing SERIAL columns except for Informix® databases where the native Informix® SERIAL management is used.

For further information on how the `seqreg` table is used, see Solution 2: Generate serial numbers from your own sequence table in the Genero Business Development Language User Guide.

Managing concurrency

The default template manages concurrent access.

Concurrent access occurs when two or more users work on the same data, such as when more than two users are working with the same application or with different applications using common data.

**Note:** With mobile apps, only one user accessing the app at any given time; data cannot be altered by another user. Concurrent access management is disabled for generated mobile apps.

The default template checks for concurrent access and notifies the user that the data being updated has been changed by another user (either deleted or updated). The concurrent access only applies to data of the current row. The check of the concurrent access is systematically done when the user:

- starts to modify the current row
- saves updates to the current row
- deletes the current row

If the user's data was not updated by another user the application follows its normal course, such as insert, update or delete of the current row.

If the user's data was updated by another user, a warning is raised and the user is asked to refresh the data. If the user rejects the request to refresh the data, the application stays in its current state and the application's process does not continue. If the user accepts the request to refresh the data, the current row is refreshed.

If the user's data was deleted by another user a warning is raised and the user is asked to refresh the data. If the user rejects the request to refresh the data, the application stays in its current state and the application's process does not continue. If the user accepts the request to refresh the data, the current row is deleted from the current data set.

Concurrent access management uses the optimistic locking strategy. The data that the user wants to update or delete is compared to what is stored in the database ensuring no change has occurred. A check is done when starting to modify the current row in the application to detect data modification at the earliest. See the topic Optimistic Locking in the Genero Business Development Language User Guide for more information.

**Note:** In the database layer, the default template uses physical row locking for update and delete operations. This ensures data cannot be changed during the database transaction. Physical row locking can only be done with databases that handle locks.

Cascade delete

A foreign key with a cascade delete specifies that if a row in the parent table is deleted, then the corresponding rows in the child tables are automatically deleted.

**Note:** An infinite delete cascade cycle can occur when table A references table B, table B references table C, table C references table A (and so on) and all delete cascade boxes are checked. While running the delete cascades, if a row is found that was a candidate for deletion during the cycle, an infinite delete cascade cycle occurs and results in a runtime error.

**Related concepts**

Add foreign keys on page 351
A foreign key constraint specifies that values in one table must also appear in another table. Foreign keys that are part of the database structure are displayed, but foreign keys can also be added.

### Working with forms

Special properties are used to specify what logic should be generated for a form and how the form should behave during the various states (DISPLAY, MODIFY, ADD, SEARCH).

- **Mobile forms** on page 285
- **Enable and disable CRUD logic** on page 291
- **Form behavior in CRUD states** on page 293
- **Control the row position in form** on page 296
- **Opening a form with a subset of data** on page 296
- **Field activation** on page 296
- **Define queries and data order** on page 297
- **Define a dynamically populated ComboBox** on page 297
- **Lookup fields** on page 299
- **Add buttons to form** on page 301
- **Add formonly (nondatabase) fields to a form** on page 302
- **Master-detail forms** on page 302

### Related concepts

- **Form Designer** on page 486

*Form Designer* is a visual editor that supports the creation, editing, and layout of Genero forms in Genero Studio.

### Mobile forms

This section includes topics on working with forms for mobile apps.

- **Mobile form patterns** on page 285
- **Display image with table row in mobile form** on page 288
- **Reuse a common form** on page 289
- **Rowbound actions** on page 289

### Mobile form patterns

Mobile platforms require special form patterns. These patterns can be modeled in BAM.

**Note:** Examples shown here can be found in the **MobilePatterns** demo from the Welcome Page.

#### Example: Table with 2 actions opens a common CRUD form

![Diagram showing a table with 2 actions opening a common CRUD form.]

**Figure 161: Model of table to grid form pattern**

This common mobile pattern can be modeled in BAM by creating a table container form to display the rows and setting a relation to a grid container form for CRUD operations on the individual record selected from the table display.
To see this example model, open the MobilePatterns project from the Welcome Page and open the MobilePatternsAppFlow.4ba file. To run this example, execute MobilePatternsApp to your emulator or device. Select Table Usage > 2 actions on the Table.

On mobile platforms, a table container displays as a list view, not as a grid with columns. As a list view, it only displays the two first columns’ content and any associated row image. There is no way to manipulate columns (hide, reorder, resize, or sort).

Depending on the alignment property of the two columns, a row is displayed as either

- On two lines for each row.
- On a single line with the first column is left-aligned and the second column is right-aligned. For example, if the first column is a text column (left-aligned) and the second column is a number (right-aligned), they will be rendered on a single line for the row.

If a table is the one and only element in a form, a standard table view displays.

![Image showing a table form to grid form pattern]

**Figure 162: Running app with table form to grid form pattern**

1. The table_2_actions form includes a table that has only the delete action code generated. (The record for this form specifies only canDelete functionality.) The icon or a swipe on the row triggers the delete action.
2. This table displays rows with two lines on each row. The first column is displayed above the second column. In the table container definition, two display only columns have been added to the table to concatenate the first and last name data in the first position and the address data in the second position. The logic to concatenate the data for these fields has been added as custom code in a POINT. See Finding the right place to customize on page 310. For example:

```
{<POINT Name="fct.record1_computeFields.user" Status="MODIFIED">}
   LET l_rec_BRComputedFields.recordfield1 = SFMT("%1 %2", p_br_fields.account_firstname CLIPPED, p_br_fields.account_lastname CLIPPED)
   LET l_rec_BRComputedFields.recordfield2 = SFMT("%1 - %2 - %3", p_br_fields.account_addr1 CLIPPED, p_br_fields.account_city CLIPPED, p_br_fields.account_lastName CLIPPED)
```

```
3. The user can browse the list, delete a selected row, or tap the row to go to a form populated with the selected record. The `doubleclick` property on the table specifies the action triggered when the row is selected. In this example the value of the `doubleclick` action is `open_common_form`. The code is automatically generated to open the form connected to the table form by a relation in the model.

See the Using tables on mobile devices topic in the Genero Business Development Language User Guide for more on table rendering and ergonomics in mobile apps.

4. The form opens and displays the correct record. This is achieved by setting some properties on the relation to the grid form on the BA diagram:
   - Position, Source Field property is set to `account.userid`.
   - UI Settings, Action is set to the `open_common_form` action.
   - UI Settings, Open Mode is set to DISPLAY.

5. All CRUD functionality is generated for the grid_common form (search (QBE), read and delete, create, and update). Available actions display in the order the ON ACTION statements appear in this dialog in the code. Only the first few actions appear in the menu bar at the top, depending on mobile device type.

6. All other available actions appear when the full menu is displayed by the user (by pressing a menu button on a phone, for example).

Example: Single CRUD forms or Common CRUD form

There are two options when modeling CRUD grid forms for mobile. Typically a grid form is used for all CRUD operations. You can model this by creating one common form and managing which operation(s) are available through the relation to the common form.

![Figure 163: Many vs one common form](image)

1. The `table_5_actions_v1` table form calls a different form depending on the action the user triggers. Each form handles a single operation such as (search (QBE), read and delete, create, and update).
2. The table_5_actions_v2 table form calls just one form, but models four different relations to the common form. Each relation opens the form for a different operation such as (search (QBE), read and delete, create, and update). The *Open Mode* and *Action* properties are uniquely set on each of the four relations. For example, one of the four relations has the *Action* property set to *append* and the *Open Mode* property set to *ADD*. When the user triggers the *append* action, the common form is opened and ready to accept input. Furthermore, the *ADD Behavior* properties on the relation overwrite the form defaults and specify to *ExitForm* when the user accepts or cancels the input. See *Overwriting a Form’s behavior with a Relation to the Form* on page 294.

**Display image with table row in mobile form**
You can display an image with each table row in a mobile form.  

**Note:** Example shown here is the *OrderLines* form in the *OfficeStoreMobile* demo.

**Example: Table with image**
This app displays a form that is made up of a table container. The table container has special features for mobile apps.

1. An image in the first position.

In this example, the database table includes a column (prodpic) that contains the name of the image to display for the selected row. During development, the images are found in the directory specified by the FGLIMAGEPATH environment variable. For apps that will be deployed, the packaging specifies where to find the images for inclusion in the app package.

![Figure 164: Running app with table form with image](image-url)

**Form structure**
To create this table view for your mobile app:

- The table structure for this form includes a picture column set to a *Phantom* type, and two fields (*masterInfo* and *detailInfo*) to hold the concatenated display data.
- The *masterInfo* field has the *IMAGECOLUMN* attribute (seen as the *image column* property in Genero Studio) set to the name of the field containing the name of the image.
When rendered in a mobile environment, the image defined by the Phantom field displays in the first position of each row in the table.

See also Using tables on mobile devices in the section of the Genero Business Development Language User Guide.

Reuse a common form
Writing applications often requires creating a lot of small forms, especially for mobile apps. To make app management easier, consider creating a common form that has all operations generated for it, then limiting the operations as it is opened in different scenarios in the app.

There are several ways to reuse a form in an application. For an example of setting many relations to a common form, see Example: Single CRUD forms or Common CRUD form on page 287.

Use it as is
In the BA diagram, set a relation to the form as is.

Reuse a subset of functionality
Create the form will all CRUD operations generated. Then, in the BA diagram relation to the form, set the properties to disable Add, disable Modify, disable Delete, and/or disable Search. When the form is opened, only a subset of the operations will be available to the user.

Display a subset of the data depending on the parent form
The Filter properties on a relation are used to apply a SQL filter to show a subset of data when opening a form. See Opening a form with a subset of data on page 296.

Position the cursor depending on the parent form
When a form is opened from another form, the current position is always set to the first row. With positioning, you can control the row that is preselected when the form is opened. The Position section Source Field property on the relation to the form controls which row will be pre-selected and displayed when the form is opened. See Control the row position in form on page 296.

Rowbound actions
Set the Row Bound property on a relation between forms to render a contextual action to the current row.

The Row Bound property specifies whether the action set in the Action property will appear as a contextual menu option on the row. If your form only has one record, BAM automatically uses that record as the Source Record for the Row Bound property. If your form has more than one record, you must set the Row Bound and the Source Record property to specify for which record the Row Bound property is to be applied.
To set an action other than the action being used to open a form to rowbound, customize the action in the code with `ATTRIBUTES(ROWBOUND)`. For example, this custom code adds a user action `print` and makes it rowbound.

```
{<POINT Name="dlg.recOrders.uiDisplay.userControlBlocks" Status="MODIFIED">}
  ON ACTION print ATTRIBUTES(TEXT = "Print", ROWBOUND)
  DISPLAY "print"
{</POINT>}
```

The `delete` action is always available via swipe left unless you have set the `canDelete` form functionality property to unchecked (false). See Enable and disable CRUD logic on page 291.

For more information on rowbound actions, see the Rowbound topics in the Genero Business Development Language User Guide.

**Android™ list view with rowbound actions**

On Android™, when rowbound actions are defined, each row of a list view shows the three-dot indicator. Tap this icon to bring up a row context menu with options to execute the corresponding rowbound actions. Swipe the row from the right to the left to fire the delete action, if defined.

**iOS list view with rowbound actions**

On iOS devices, when you swipe your finger from right to left, **More...** and/or **Delete** icons show up in the row. Tap **More...** to bring up a list of rowbound actions to execute. Tap **Delete** to fire the corresponding delete action code.
Enable and disable CRUD logic

Form functionality properties (canDisplay, canAdd, canModify, canDelete, canSearch, canEmpty) can be set on each record of a form to specify whether the program logic of display, add, update, delete, search and/or display empty should be generated. Generated functionality can be disabled on a relation to the form in the BA diagram.

Functionality properties on the form record

The program and user interface logic is generated when functionality properties are set. The state of the action (enabled/disabled) or the availability of the action in the Toolbar and/or Topmenu depends on the setting of the
**Functionality** properties. For example, if the `canSearch` property is checked, the form will allow for data queries and the Toolbar and Topmenu will include a button and menu option for searching.

- **canDisplay**
  This form can be used to display data in this record. Allows the user to browse records with next and previous.

- **canAdd**
  The form can be used to add data in this record.

- **canModify**
  The form can be used to modify data in this record.

- **canDelete**
  The form can be used to delete data in this record.

- **canSearch**
  The form can be used to search data in this record. `canSearch` is the only functionality property available to Zoom forms.

- **canEmpty**
  This form can be used to sit empty waiting for the user to trigger an action. Useful when a form is to be presented empty, prior to running a search query.

---

**Figure 166: Setting the record Functionality properties**

**Disabling Functionality properties on a relation to a form**

You can disable functionality generated for a form in a relation to the form on the BA diagram. For example you may have a form that generates code for all functionalities, but in some situation when the form is opened you do not want a functionality to be available. In this example, the form record has all functionalities selected, thus all CRUD logic will be generated. On a specific relation to this form, however, the Add and Search functionalities are disabled preventing the user from adding or searching from this form when opened.

---

**Figure 167: Disabling some of the generated functionality on the relation to the form**
Form behavior in CRUD states

The UI Settings properties specify the initial and default state of the selected form as well as the behavior of the form during each state (DISPLAY, MODIFY, ADD, and SEARCH). These settings can be set on the form entity in the BA diagram, but overwritten by a relation to the form.

**Figure 168: UI Setting properties**

**Initial and default states**

**Open Mode**

Open Mode is the initial state of the form when opened. The rendered form's default Toolbar allows the user to switch modes.

**Default Mode**

Default Mode is the mode in which you return after leaving another mode.

**Table 78: Form States**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAY</td>
<td>The default mode. Data is retrieved from the database and displayed as a record on the form, one at a time. The default Toolbar allows the user to scroll through the list of records.</td>
</tr>
<tr>
<td>MODIFY</td>
<td>The user can modify the currently displayed record.</td>
</tr>
<tr>
<td>ADD</td>
<td>The user can enter a new record.</td>
</tr>
</tbody>
</table>
### Option Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEARCH</td>
<td>The user can enter criteria for a search of the database, displaying the records that match that criteria.</td>
</tr>
<tr>
<td>EMPTY</td>
<td>An empty form is displayed. The user can select the search or add actions from the Toolbar to change the mode.</td>
</tr>
</tbody>
</table>

### Behavior properties

![Form Behavior properties](image)

**Figure 169: Form Behavior properties**

The **Behavior** properties control how the form should behave during each state: **DISPLAY**, **MODIFY**, **ADD**, **SEARCH** for CRUD forms and **DISPLAY**, **SEARCH** only for Zoom forms.

- **On accept**
  - What should happen when the `accept` action is triggered.
- **On cancel**
  - What should happen when the `cancel` action is triggered.
- **On close**
  - What should happen when `close` action is triggered.
- **On exit**
  - What should happen when the `exit` action is triggered.

**Table 79: Behavior properties**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExitForm</td>
<td>Close the form and close the application if it is the last form open.</td>
</tr>
<tr>
<td>ExitApp</td>
<td>Close the application.</td>
</tr>
<tr>
<td>ReturnToDefaultMode</td>
<td>Return to the mode specified as the default in the Default Mode property.</td>
</tr>
<tr>
<td>ReturnToCallerMode</td>
<td>Return to the previous mode.</td>
</tr>
<tr>
<td>StayInMode</td>
<td>Stay in the current mode.</td>
</tr>
</tbody>
</table>

**Overwriting a Form’s behavior with a Relation to the Form**

Property values on a relation to a form overwrite the form's default property values. This allows you to set default property values on the form, but change them on the relation to the form.

In the modeling of mobile apps, it is common to have multiple relations to a common form. This allows you to reuse a common form, opening it in a different state depending on the action triggered. In this example, there are 2 relations...
to the Order form. Each relation specifies how the form should behave depending on the action that opens it. One of those relations, Add an Order, specifies that when the form is opened with the append action, the form opens in the ADD state. It also changes the default ADD behavior by exiting the form (ExitForm) when the user either accepts or cancels the ADD operation.

Figure 170: Different Relations to a Common Form

**Data refresh**

Data refresh behavior is specified with the Data Refresh property on the relation between two forms. If the source form is in DISPLAY mode and opens a destination form, when the destination form closes, the data displayed in the source form refreshes.

**Table 80: Data Refresh property values**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Documents</td>
<td>All data are refreshed.</td>
</tr>
<tr>
<td>Current Document</td>
<td>The current document (current row of the master record) is refreshed. This is the default value.</td>
</tr>
<tr>
<td>All Rows</td>
<td>All rows of the current record defined by the Position, Source Field property values are refreshed.</td>
</tr>
<tr>
<td>Current Row</td>
<td>The current row defined by the Position, Source Field property values is refreshed.</td>
</tr>
<tr>
<td>None</td>
<td>No refresh is done.</td>
</tr>
</tbody>
</table>

The Current Row and the Current Record are defined by the Position, Source Field properties of the relation. If the Position, Source Field property values are the unique key of the record, then the Data Refresh uses the row defined by the returned values.

If the Data Refresh is for All or Current Document, then the Data Refresh uses the current document by following the record relations from the row of the record defined by the Position, Source Field property values to the master record (from unique key to unique key of the parent).
Note: If the canAdd property is enabled in the destination form, set the dataRefresh property to All Rows or All Documents, otherwise the added rows will not be displayed on return to the source form.

Control the row position in form

When a form is opened from another form, the current position is always set to the first row. With positioning, you can control the row that is preselected when the form is opened. The Position section Source Field property on the relation to the form controls which row will be pre-selected and displayed when the form is opened.

There are several ways that you may want to control the row position when a form is opened. The Source Field must match the target form field in number and type for the positioning to work.

- **Open to the same record as the parent form record**: Set the Source Field property to the primary key of the parent form record. When the form opens, it will open to the same record as the parent form.

- **Open to a position depending on a field value of the parent record**: Set the Source Field to the field of the parent record you want to use. For example, an account form and a country form may have a relation between them that specifies the account.country_id as the positioning Source Field. While in the account form, suppose the user is on a record with a country field value of "USA". If the user then opens the country form from the account form (through a Zoom form, for example), the country_id ("USA") will be preselected in the form instead of the first record. Often this will be a field that corresponds to a foreign key. (For example, the account table has a foreign key to the country table.)

Opening a form with a subset of data

The Filter properties on a relation are used to apply a SQL filter to show a subset of data when opening a form.

A form is typically opened with a set of data for the user to browse. This data set is controlled by the data set relationships and the where clause specified in the form record query property. In some instances, you may want to open the same form with a limited set of data. To do so, in the BA diagram, add a filter on the relation to the form.

The Filter properties consist of the Destination Field to filter on (for example, country.id) and the Source Field values to use for filtering (for example, the value in account.country_id). When the form is opened, it will filter the results displayed based on the filter.

Field activation

An active field is one that can be input by the user. Field activation occurs when the application is in an edit mode (MODIFY or ADD modes).

CRUD operations (create, read, update, delete) apply to fields belonging to the business record master table thus only the business record master table fields can be edited and saved.

These fields can be edited directly by user input or automatically by ascending lookups.

In edit mode, active fields are:

- fields belonging to the business record master table
- fields running ascending lookups

Automatically deactivated fields:

- Formonly fields
- Unique key fields when the form is in MODIFY mode
- Fields defined as a foreignField in a business record master-detail relationship
This default behavior can be bypassed by adding code in the predefined BLOCK/POINTs. Field activation sections are centralized to ease customization.

**Note:** When the user is in edit mode (property canModify is true) and no editable field is present, the program stops with an FGL error. In addition, if all fields of the business record master are primary keys, they can not be modified even if canModify is true. In the case of a master-detail relationship, a DISPLAY ARRAY / INPUT ARRAY may be generated instead of an INPUT ARRAY / INPUT ARRAY.

### Define queries and data order

Joins between tables referenced in the form are set up in the query property of the business record.

The *Edit query* dialog allows you to specify the joins between the tables in a record used for a form, report, or service.

**Related concepts**

Joins and Data order on page 539

The *Edit query* dialog allows you to specify the joins between the tables in a record used for a form, report, or service. Joins between tables referenced in the form are set up in the *query* property of the business record.

### Define a dynamically populated ComboBox

![ComboBox with Data](image)

**Figure 171: Combobox with Data**

**Populated ComboBox Example**

In this example, when the user is in an input mode and selects the ComboBox field, a list of the valid country descriptions are displayed. When the user selects a country description (country.codedesc) and accepts the selection, the account table is updated with the correct country information (account.country).

1. Modify the form. Find the master field that is to be replaced with a ComboBox of populated values and move it to another location on the form. You can convert this field to a Phantom widget as it will be needed for the code, but not needed to display to the user (account.country in this example).
2. On the Records tab, add the reference field to the master field in the same record. Right-click on the record and choose **Add Field**. (The `country.codedesc` field, the field representing the list of country descriptions, is added to the account record in this example.)

3. Right-click on the record and select **Edit Query**. Establish the join between the master table and reference table. (The `account.country` is joined to `country.code` in this example.)

4. Select the reference field in the record and set its **lookup** property. This name will be used as a function name in the generated code that is triggered to perform the ascending lookup.

![Figure 172: Lookup](image)

5. Return to the **Form** tab. Notice that the new field (`country.codedesc`) has been added to the form in the upper left. Move it to its correct location and convert it to a ComboBox widget.

6. Set the **initializer** property on the ComboBox field to a unique name. This name will be used as a function name for the generated code that populates the ComboBox when it is built in the user interface.
7. Save the form. **Build** and **Execute** the program.

**Lookup fields**

A lookup field is a field linked to another one inside a record. Lookup fields contain a value that is retrieved from a reference table instead of being input by a user.

There must be a join relationship between the master and reference tables inside a record. There are two types of lookup fields: descending and ascending.

**descending**

Descending lookup fields are automatically managed. A descending lookup field specifies that when a master field is updated, the related non-master fields will also be updated. The fields that are automatically updated are determined from the joins in the Query Editor. For this reason, the only type of lookup that is set up in the form is ascending.

**ascending**

Ascending lookup fields specify that when a field of a non master table is updated, the related master table field will be updated. An ascending lookup is implemented by entering a value in the `initializer` property on the field (that has sense only for fields of a non master table in the record) in the form record.

**Composite fields in a lookup**

If the lookup requires more than one field, set the `initializer` property with the same value on all these fields. The lookup is resolved with the set of values of these fields.
**Descending lookup example**

In this example, the form is primarily for the **account** table (master), but when a customer's account record is displayed or input, it also displays the corresponding country name from the **country** table (descending lookup).

![Form showing lookup field](image)

**Figure 174: Form showing lookup field**

**Define an ascending lookup field**

Ascending lookup fields specify that when a field of a non master table is updated, the related master table field will be updated.

1. Open the form to be modified. A record of the form must include fields from the master table as well as the lookup field(s) from the reference (non master) table.
2. Confirm the join between the tables. Right-click on the record and select **Edit Query** to confirm the join between the master and reference (non master) table.
3. Select the Record tab and then the lookup field column to see its properties. Set the **lookup** property to a unique name (*lkp_acct_ctry_desc* in this example). To implement a composite lookup, set the lookup property of both lookup fields to the same value.
Figure 175: lookup Property

The generator will use this name in the function call that triggers the ascending lookup when the user modifies the reference (non master) field data. If the value is found in the reference table, it is used to update the master field. If it is not found in the reference table, the original values on the form remain intact.

4. Save the form. Build and Execute the program.

Related concepts
Lookup fields on page 299
A lookup field is a field linked to another one inside a record. Lookup fields contain a value that is retrieved from a reference table instead of being input by a user.

Add buttons to form

If you add a widget to your form that can trigger an action, such as a Button, you can associate an action from the generated program to the widget.

1. Open the form.
2. Select Widget > Button from the menu.
3. Draw out a Button in a container on the form.
4. Provide the name of the action in the name property for the Button.
Add formonly (nondatabase) fields to a form

Formonly fields are added by creating a non database field and making the record inactive. You must supply the business logic for the formonly field - it is not generated.

Master-detail forms

A Genero application can be modeled to a master-detail relationship between two tables.

Desktop applications

A Genero desktop application can display a form that contains a master-detail relationship between two tables.

The user can search for a row in the master table, and the corresponding rows in the detail table will also be displayed. The values in rows from both tables can be added, deleted, or modified.

The form must contain fields from both tables and the table relationships must be set.

Mobile applications

A Genero mobile application requires two forms to manage the master-detail relationship: one form for the master, one form for the detail.

Related concepts

Business records (data sets) on page 493
Business records model the data definition, structure and table relationships of the data used in a form, report, and/or web service. Business records are designed and modified in the Business Records diagram.

**Related tasks**
- Add a detail list to the form on page 211
- Add a list to your form and create the master-detail relationship.

### Adding custom code

This section includes topics on how to add custom code to the generated code.

- **Understanding what gets generated** on page 303
- **Finding the right place to customize** on page 310
- **Using POINTs and BLOCKs** on page 313

**Related concepts**
- **BAM Template Developer Guide** on page 1094

Genero Studio comes with a set of templates used with the Business Application Modeler (BAM) for the code generation. You are free to customize these templates or create your own templates so that the generated application meets your requirements.

### Understanding what gets generated

4gl files are generated for each diagram entity.

When a generated program is built, the generated 4gl files are compiled and linked with the other program files to create the executable application. The compiled binary versions are stored in the project's `/bin` directory.

The generated 4gl and XML files for each diagram entity are shown in the Intermediate Files folders in the project. This figure shows two generated files for the Program entity, Account.4prg.

**Note:** The XML file listed in the Intermediate Files folder are temporary files that consolidate all data from diagrams and are used to generate the 4gl code. They are not places to customize code, but can be useful to a template developer as these XML files are used as input to the code that is generated. See How code is generated on page 225.

![Figure 177: Intermediate Files](image)

The default template set is designed to generate organized and functional code for a data-driven business application. See The default template features on page 230.

Each generated 4gl file has many functions. Each function has specific places called POINTs and BLOCKs where you can add your own code. Code Files and Code Link Properties can be used to directly access generated files and specific POINTs.
Table 81: Generated files for the program (4prg)

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program.4gl on page 305</td>
<td>This file contains the MAIN function. It defines global variables, connects to the data source with the CONNECT TO statement, loads styles and action defaults files, and performs some initialization tasks for the application.</td>
</tr>
</tbody>
</table>

Table 82: Generated files for the database (4dbx)

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema_dbxdata.4gl on page 309</td>
<td>This file manages the database SELECT, INSERT, UPDATE, and DELETE statements.</td>
</tr>
<tr>
<td>Schema_dbxconstraints.4gl on page 310</td>
<td>This file manages table and column constraints.</td>
</tr>
</tbody>
</table>

Table 83: Generated files for form entities (4fdm, 4fdz)

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form_ui.4gl on page 306</td>
<td>This file contains functions called from MAIN. It opens the form based on the open mode, loads the Toolbar and Topmenu, launches functions to manage application states, calls MENU or DIALOG statements for the various states and calls functions in the Form_ui_dialog.4gl to retrieve and manage subdialogs.</td>
</tr>
<tr>
<td>Form_uidialog.4gl on page 307</td>
<td>This file defines the subdialogs and modular variables. It manages the states of actions, fields and subdialog and calls functions in the Form_uidialogdata.4gl.</td>
</tr>
<tr>
<td>Form_uidialogdata.4gl on page 308</td>
<td>This file defines the modular variables, manages UI data (clear, fetch, set default values), and calls functions in the Form_uidata.4gl.</td>
</tr>
<tr>
<td>Form_uidata.4gl on page 309</td>
<td>This file includes functions to retrieve and manage the database data. It defines the records for the database columns and tables, creates and fills a dynamic row of data, creates and fills a dynamic row of record keys, contains the SQL statements to SELECT, UPDATE, and DELETE rows in the database, and handles the SQL transactions.</td>
</tr>
</tbody>
</table>
Important: Reports and Web Services are not supported on mobile platforms.

Table 84: Generated files for the reports

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report_report.4gl</td>
<td>This file contains the report driver for the report.</td>
</tr>
<tr>
<td>Report_reportdata.4gl</td>
<td>This file contains the SQL and BDL statements to fetch the report data.</td>
</tr>
<tr>
<td>reportdata_report.rdd</td>
<td>This file is a data file generated for use when the developer creates the report definition file (4rp).</td>
</tr>
</tbody>
</table>

Table 85: Generated files for the web services

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebService_service.4gl</td>
<td>This file handles business record type definition (only take public fields), defines the module variables (input and output variable for service's CRUD operations), and sets up the service and publishes the CRUD operations.</td>
</tr>
<tr>
<td>WebService_serviceclient.4gl</td>
<td>This file is the Client API of a JSON Web Service. It contains a set of function wrappers which allow communication with a JSON Web Server.</td>
</tr>
<tr>
<td>WebService_uidata.4gl</td>
<td>This file handles business record type definition, contains operations to insert / update / delete rows in the database, contains ascending lookup management.</td>
</tr>
<tr>
<td>WebServiceServer_server.4gl</td>
<td>This file contains the MAIN function, sets up the web service engine, contains the CONNECT TO BDL statement, registers services, and listens for incoming requests.</td>
</tr>
<tr>
<td>WebService.xml</td>
<td>The XML file contains the information about the web service needed to generate the logic in the 4gl files.</td>
</tr>
<tr>
<td>WebServiceServer.xml</td>
<td>This file contains the information about the Webservice Server needed to generate the logic in the 4gl files.</td>
</tr>
</tbody>
</table>

Table 86: Library files used by a generated app

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>libdbappCore.4gl</td>
<td>This file includes common functions for managing errors and strings.</td>
</tr>
<tr>
<td>libdbappExt.4gl</td>
<td>This file includes common functions for managing the interaction with mobile device peripherals. See Add mobile device features (Photo, Gallery, Phone, Mail, SMS, Contact, Maps, Barcode) on page 272.</td>
</tr>
<tr>
<td>libdbappFormUI.4gl</td>
<td>This file includes common functions related to form transitions and states.</td>
</tr>
<tr>
<td>libdbappSQL.4gl</td>
<td>This file includes common functions related to SQL transactions, errors, and database statements.</td>
</tr>
<tr>
<td>libdbappUI.4gl</td>
<td>This file includes common functions related to the user interface such as Topmenus, Toolbars, and front end type.</td>
</tr>
</tbody>
</table>

Program.4gl

This file contains the MAIN function. It defines global variables, connects to the data source with the CONNECT TO statement, loads styles and action defaults files, and performs some initialization tasks for the application.

The function(s) in this file include POINT and BLOCK sections where you can add your own code.

Table 87: The Program.4gl file

<table>
<thead>
<tr>
<th>Generated Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>The MAIN function of the program.</td>
</tr>
</tbody>
</table>
**Form_ui.4gl**
This file contains functions called from MAIN. It opens the form based on the open mode, loads the Toolbar and Topmenu, launches functions to manage application states, calls MENU or DIALOG statements for the various states and calls functions in the Form_uidialog.4gl to retrieve and manage subdialogs.

**Table 88: The Form_ui.4gl file**

<table>
<thead>
<tr>
<th>Generated Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uiOpenForm</td>
<td>Opens form according to the open mode.</td>
</tr>
<tr>
<td>uiOpenFormByKey</td>
<td>Opens form according to the open mode and position to the key.</td>
</tr>
<tr>
<td>uiAutomaton</td>
<td>The automaton function allowing the switch between modes (DISPLAY, MODIFY, ADD, SEARCH, EMPTY)</td>
</tr>
<tr>
<td>uiDisplay</td>
<td>Manages the DIALOG block for the DISPLAY mode.</td>
</tr>
<tr>
<td>uiInput</td>
<td>Manages the DIALOG block for the MODIFY and ADD modes.</td>
</tr>
<tr>
<td>uiConstruct</td>
<td>Manages the DIALOG block for the CONSTRUCT mode.</td>
</tr>
<tr>
<td>uiSearch</td>
<td>Manages the DIALOG block for the SEARCH mode.</td>
</tr>
<tr>
<td>uiEmpty</td>
<td>Manages the DIALOG block for the EMPTY mode.</td>
</tr>
<tr>
<td>initializeDefaultActions</td>
<td>Initializes the list of default actions.</td>
</tr>
<tr>
<td>initializeDefaultUISettings</td>
<td>Initializes the UI settings of the form.</td>
</tr>
<tr>
<td>action_</td>
<td>Launch a module.</td>
</tr>
<tr>
<td>validateCRUDOperationWrapper</td>
<td>Validate a CRUD operation for a record (INSERT, UPDATE, DELETE).</td>
</tr>
<tr>
<td>initializeReports</td>
<td>Initializes the list of available reports.</td>
</tr>
<tr>
<td>processReport</td>
<td>Load, configure and run the current report.</td>
</tr>
</tbody>
</table>
**Form_uidialog.4gl**

This file defines the subdialogs and modular variables. It manages the states of actions, fields and subdialog and calls functions in the `Form_uidialogdata.4gl`.

**Table 89: The Form_uidialog.4gl**

<table>
<thead>
<tr>
<th>Generated Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getDefaultField</td>
<td>Get the default DIALOG field name.</td>
</tr>
<tr>
<td>validateCRUDOperation</td>
<td>Validate a CRUD operation (INSERT, UPDATE, DELETE).</td>
</tr>
<tr>
<td>processCRUDOperation</td>
<td>Process a CRUD operation (INSERT, UPDATE, DELETE). Check the concurrent access.</td>
</tr>
<tr>
<td>synchronizeUI</td>
<td>Synchronize the UI according an internal action returned by a CRUD operation.</td>
</tr>
<tr>
<td>setAllCurrentRow</td>
<td>Set the current row for each subdialog in the case of master-details to restore the context when the mode switches.</td>
</tr>
<tr>
<td>uiDisplay</td>
<td>DISPLAY ARRAY dialog.</td>
</tr>
<tr>
<td>uiInput</td>
<td>INPUT dialog.</td>
</tr>
<tr>
<td>uiConstruct</td>
<td>CONSTRUCT dialog.</td>
</tr>
<tr>
<td>setActionStates</td>
<td>Sets the state of the dialog actions to active or inactive according to the mode.</td>
</tr>
<tr>
<td>setFieldActive</td>
<td>Initialize fields state according to startup state.</td>
</tr>
<tr>
<td>getRecordFieldList</td>
<td>Gets the field list of a subdialog.</td>
</tr>
<tr>
<td>navigate</td>
<td>Navigation for current record.</td>
</tr>
<tr>
<td>setAllCurrentRow</td>
<td>Set the current row on all subdialogs whose container is a 'table' to restore the context.</td>
</tr>
<tr>
<td>init</td>
<td>Populate the referenced ComboBox.</td>
</tr>
</tbody>
</table>
**Form uidialogdata.4gl**
This file defines the modular variables, manages UI data (clear, fetch, set default values), and calls functions in the `Form uidata.4gl`.

**Table 90: Form uidialogdata.4gl**

<table>
<thead>
<tr>
<th>Generated function</th>
<th>Function features</th>
</tr>
</thead>
<tbody>
<tr>
<td>seek</td>
<td>Gets the index of an item matching the given criteria.</td>
</tr>
<tr>
<td>fetchRowAndDetails</td>
<td>Fetch row and details data for record.</td>
</tr>
<tr>
<td>fetchAll</td>
<td>Fetch the data set for a record.</td>
</tr>
<tr>
<td>fetchRow</td>
<td>Fetch row data for record.</td>
</tr>
<tr>
<td>fetchDetails</td>
<td>Fetch details data for record.</td>
</tr>
<tr>
<td>clear</td>
<td>Clear record data.</td>
</tr>
<tr>
<td>clearRowAndDetails</td>
<td>Clear row and details data for record.</td>
</tr>
<tr>
<td>setDefaultValues</td>
<td>Initialize record with default values.</td>
</tr>
<tr>
<td>runUpdates</td>
<td>Run the treatments after the update of the field (like the descending, ascending lookup, ...). This function is especially called in the control blocks ON CHANGE, ON ACTION zoom.</td>
</tr>
<tr>
<td>runDescLookup</td>
<td>Generate descending lookup functions triggered by fields of the master table of the record. It calls 'DescLookup' defined in data then update the fields if lookup successes.</td>
</tr>
</tbody>
</table>
**Form_uidata.4gl**
This file includes functions to retrieve and manage the database data. It defines the records for the database columns and tables, creates and fills a dynamic row of data, creates and fills a dynamic row of record keys, contains the SQL statements to SELECT, UPDATE, and DELETE rows in the database, and handles the SQL transactions.

**Table 91: The Form_uidata.4gl**

<table>
<thead>
<tr>
<th>Generated Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getKeys</td>
<td>Create and fill a dynamic array of Business Record (BR) Unique Keys (UK).</td>
</tr>
<tr>
<td>getDataByKey</td>
<td>Get data from a given Business Record (BR) Unique Key (UK) (may be a composite key).</td>
</tr>
<tr>
<td>getDataArray</td>
<td>Create and fill a dynamic array of Business Record (BR) fields.</td>
</tr>
<tr>
<td>insertRow</td>
<td>Insert a row in the database.</td>
</tr>
<tr>
<td>updateRow</td>
<td>Update a row in the database.</td>
</tr>
<tr>
<td>deleteRow</td>
<td>Delete a row in the database.</td>
</tr>
<tr>
<td>deleteRowWithConcurrentAccess</td>
<td>Delete a row in the database.</td>
</tr>
<tr>
<td>checkRow</td>
<td>Check a row in the database.</td>
</tr>
<tr>
<td>checkRowConcurrentAccess</td>
<td>Check a row in the database.</td>
</tr>
<tr>
<td>DescLookup</td>
<td>Retrieve data according the descending lookup after update of the field.</td>
</tr>
<tr>
<td>AscLookup</td>
<td>Retrieve data according the ascending lookup after update of the field(s).</td>
</tr>
<tr>
<td>fillArray</td>
<td>Populate an array of key/value pairs.</td>
</tr>
<tr>
<td>computeFields</td>
<td>Compute FORMONLY fields of the Business Record (BR).</td>
</tr>
</tbody>
</table>

**Schema_dbxdata.4gl**
This file manages the database SELECT, INSERT, UPDATE, and DELETE statements.

Find this file in the Intermediate Files folder listed with your database schema in your project.
### Table 92: Schema_dbxdata.4gl

<table>
<thead>
<tr>
<th>Generated function</th>
<th>Function features</th>
</tr>
</thead>
<tbody>
<tr>
<td>selectRowByKey</td>
<td>Select a row identified by the primary key in the table.</td>
</tr>
<tr>
<td>insertRowByKey</td>
<td>Insert a row in the table and return the table keys.</td>
</tr>
<tr>
<td>updateRowByKey</td>
<td>Update a row identified by the primary key in the table.</td>
</tr>
<tr>
<td>deleteRowByKey</td>
<td>Delete a row identified by the primary key in the table.</td>
</tr>
<tr>
<td>deleteRowByKeyWithConcurrentAccess</td>
<td>Delete a row identified by the primary key in the table if the concurrent access is successful.</td>
</tr>
<tr>
<td>deleteReferencingRowsByKey</td>
<td>Delete rows referencing the primary key of the table.</td>
</tr>
<tr>
<td>checkRowByKeyWithConcurrentAccess</td>
<td>Check if a row identified by the primary key in the table has been modified or deleted.</td>
</tr>
<tr>
<td>setDefaultValuesFromDBSchema</td>
<td>Set data with the default values coming from the DB schema.</td>
</tr>
</tbody>
</table>

### Schema_dbxconstraints.4gl

This file manages table and column constraints.

Find this file in the Intermediate Files folder listed with your database schema in your project.

### Table 93: Schema_dbxconstraints.4gl

<table>
<thead>
<tr>
<th>Generated function</th>
<th>Function features</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkTableConstraints</td>
<td>Check constraints on the table.</td>
</tr>
<tr>
<td>checkUniqueConstraint</td>
<td>Check the primary key uniqueness constraint on the table.</td>
</tr>
<tr>
<td>checkColumnConstraints</td>
<td>Check constraints on the column.</td>
</tr>
<tr>
<td>checkFKConstraint</td>
<td>Check the foreign key existence constraint on the table.</td>
</tr>
</tbody>
</table>

### Finding the right place to customize

To determine where to add code to make your customization, use Code Link or Code File properties or consider these questions.

**Direct access with Code Links**

The most common places to customize the generated code are accessible from Code Link properties. Code Link properties provide direct access to the POINT in the appropriate generated file. For example, the Compute Fields
property on a **Form** record opens the correct generated 4GL file to the POINT location logically used for computing fields.

```xml
{<POINT Name="fct.recExample_computeFields.user">} {</POINT>}
```

You will find **Code Link** properties available on:

**Meta-schema** - Select the background of the Meta-schema diagram.
- **table**
  - Constraints
  - Key Uniqueness
  - Key Exists
  - Select
  - Insert (before)
  - Insert (after)
  - Update
  - Delete
  - Delete (Concurrent)
  - Cascade Delete
  - Optimistic Locking
  - Defaults
- **column**
  - Constraints (Column)

**Form**

- **Records tab**
  - **ManagedForm** - Select the background of the Form Records tab.
    - Open Window
    - UI Automaton
    - Display Mode Events
    - Input Mode Events
    - Search Mode Events
    - Empty Mode Events
    - Action State
- **Record**
  - Select rows
  - Select row
  - Computed fields
  - Insert
  - Update
  - Delete
  - Delete (Concurrent)
  - Lookups (Descending)
  - Combobox initializer (Data)
  - Defaults (Dialog)
  - DISPLAY attributes
  - DISPLAY events
  - INPUT events
  - CONSTRUCT events
  - Action state
• Field Activation
• Record field
• Field Activation

• Form tab
• Combobox widget
  • Combobox initializer (Data)
  • Combobox initializer (UI)

Direct access with Code Files

Code Files properties are available for direct access to the appropriate generated code file.

You will find Code Files properties available on:

Meta-schema  - Select the background of the Meta-schema diagram.
  • Data opens  Schema_dbxdata.4gl on page 309
  • Constraints opens  Schema_dbxconstraints.4gl on page 310

Form  - Select the background of the Form Records tab.
  • UI opens  Form_ui.4gl on page 306
  • Dialogs opens  Form_uidialog.4gl on page 307
  • Dialog Data opens  Form_uidialogdata.4gl on page 308
  • Data opens  Form_uidata.4gl on page 309

Manually finding the right location

If a Code Link is not available to help guide you to a location for your custom code, these questions can help you determine where to customize.

Can the modification be done in a model?  For example, can you make a change to a property on an entity in the BA diagram to get the desired effect? This is the best first choice because the models drive the code generation. When you rebuild, the generated code reflects your changes. See Modeling the application on page 232.

Can the modification be done in an external resource file?  For example, you can modify the default Toolbar file that is loaded by the program. Modifying an external resource file does not affect the code generation if you use the default naming convention (dbapp.4tb, for example). When you rebuild, the modified resource file is loaded instead of the default one. See Modify action defaults on page 318, Modify styles on page 318, Modify the Topmenu on page 319, Modify the Toolbar on page 320.

Is the modification database related?  Look at the functions in the  Schema_dbxdata.4gl on page 309 and  Schema_dbxconstraints.4gl on page 310 files. Look for a POINT in the function to place your custom logic. If a POINT is not available, modify the BLOCK of code. See Using POINTs and BLOCKs on page 313.

Is the modification for when the program begins or ends?  Look at the MAIN function in the  Program.4gl on page 305 file. Look for a POINT in the function to place your custom logic. If a POINT is not available, modify
Is the modification related to the interaction with the user?

Look at the functions in the Form_ui.4gl on page 306, Form_uidialog.4gl on page 307, Form_uidialogdata.4gl on page 308, and Form_uidata.4gl on page 309. Look for a POINT in the function to place your custom logic. If a POINT is not available, modify the BLOCK of code. See Using POINTs and BLOCKs on page 313.

Is the modification related to reporting?

Look at the functions in the _report.4gl and _reportdata.4gl files. Look for a POINT in the function to place your custom logic. If a POINT is not available, modify the BLOCK of code. See Using POINTs and BLOCKs on page 313.

Is the modification related to web services?

Look at the functions in the _service.4gl and _server.4gl, and _uidata.4gl files. Look for a POINT in the function to place your custom logic. If a POINT is not available, modify the BLOCK of code. See Using POINTs and BLOCKs on page 313.

Is the modification repeating many times?

Consider modifying the template files. See BAM Template Developer Guide on page 1094.

Using POINTs and BLOCKs

POINT and BLOCK sections are the areas in the generated code where you can add your own code. Any code added in a POINT or BLOCK is preserved in the application even when the application is rebuilt.

Code added to a POINT or BLOCK section is preserved in a .code file that is used each time the application is compiled. If you are using Source Code Management, the .code file must be committed with the project. It is not necessary to commit the generated 4gl files.

Tip: To remove all changes you have made, you can simply remove the .code and all generated 4gl files.

POINT

POINT sections are located within each function BLOCK. The more granular POINT sections are located in all relevant locations for adding business logic such as in all control blocks (BEFORE ROW, AFTER INPUT,...). Common uses of a POINT include defining your own variables, setting conditions on SELECT statements, adding or modifying actions in control blocks of interactive dialogs such as CONSTRUCT, INPUT, MENU, and changing the program flow.

Note: You are responsible for the validity of the code in a POINT.
**Table 94: POINT examples**

<table>
<thead>
<tr>
<th>Description</th>
<th>POINT name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import an additional module(s).</td>
<td>import</td>
<td>`{&lt;POINT Name=&quot;import&quot; Status=&quot;MODIFIED&quot;&gt;}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMPORT FGL mylibrary {&lt;/POINT}&gt;</td>
</tr>
<tr>
<td>Add comments in the code.</td>
<td>comment</td>
<td>`{&lt;POINT Name=&quot;user.comments&quot; Status=&quot;MODIFIED&quot;&gt;}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--Additional information about this module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{&lt;/POINT}&gt;</td>
</tr>
<tr>
<td>Define modular scope variables.</td>
<td>define</td>
<td>`{&lt;POINT Name=&quot;define&quot; Status=&quot;MODIFIED&quot;&gt;}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFINE myvar STRING {&lt;/POINT}&gt;</td>
</tr>
<tr>
<td>Define local scope variables.</td>
<td>function</td>
<td>`{&lt;POINT Name=&quot;fct.uiOpenForm.define&quot; Status=&quot;MODIFIED&quot;&gt;}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFINE myvar STRING {&lt;/POINT}&gt;</td>
</tr>
<tr>
<td>Add additional function(s) to the module.</td>
<td>user.functions</td>
<td><code>--Add user functions</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>`{&lt;POINT Name=&quot;user.functions&quot; Status=&quot;MODIFIED&quot;&gt;}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FUNCTION dispmsg()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESSAGE &quot;Program ending&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END FUNCTION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{&lt;/POINT}&gt;</td>
</tr>
<tr>
<td>Add additional actions for each of the interactive dialog statements (DISPLAY, DISPLAY ARRAY, CONSTRUCT, INPUT, INPUT ARRAY and MENU).</td>
<td>userControlBlock</td>
<td>`{&lt;POINT Name=&quot;fct.uiInput.dlg.userControlBlocks&quot; Status=&quot;MODIFIED&quot;&gt;}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ON ACTION myaction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CALL dispmsg()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{&lt;/POINT}&gt;</td>
</tr>
</tbody>
</table>

**BLOCK**

Each function in the generated code is nested within a BLOCK section.

**Note:** You can change the behavior of the generated function, however once new code has been added to a BLOCK, you take responsibility for the validity of all the code in that BLOCK.

**Status attribute**

During the first application generation, POINT and BLOCK sections will only contain the Name attribute.
When the code is changed, the POINT and BLOCK will include a Status attribute set to MODIFIED.

```xml
{<POINT Name="fct.uiDisplay.dlg.userControlBlocks" Status="MODIFIED">
  ON ACTION test
  MESSAGE "testing..." {
  </POINT>}
```

**Revert a change to a POINT or BLOCK**

Custom changes made to the source code in a POINT or BLOCK can be reverted.

1. Open your source code in Code Editor.
2. Select the area you want to revert.
3. Right-click on the selected POINT or BLOCK heading and use the Point/Block > Revert Point/Block menu option. This will add a new attribute Action="REVERT".
4. Compile the application and the changes made in the source will be removed.

**Lost POINT or BLOCK**

If a POINT / BLOCK is present in your source code but is no longer defined in the template file, the POINT / BLOCK will be considered "LOST". When the application is rebuilt, the Status attribute for the missing POINT / BLOCK will be set to LOST.

Lost sections, with their complete content intact, are commented out and put at the end of the regenerated source file. For example:

```xml
{<BLOCK Name="myBlockName" Status="LOST">
  -- DEFINE i INT -- DEFINE s STRING
  -- {<POINT Name="myPointName" Status="MODIFIED">
    LET s = "Hello World"
  </POINT>}
  LET i = 10
  {<POINT Name="myPointName2">}
</BLOCK>}
```

**Note:** The LOST status is not set on the POINT nested within the lost BLOCK, however the entire lost BLOCK is commented out including all nested POINT / BLOCK sections.

**Modifying the look and feel**

This section includes topics on how to modify the look and feel of the application.

- Default actions on page 316
- Modify action defaults on page 318
- Modify styles on page 318
- Default Topmenu and Toolbar on page 319
- Modify the Topmenu on page 319
- Modify the Toolbar on page 320

**Related concepts**

BAM Template Developer Guide on page 1094
Genero Studio comes with a set of templates used with the Business Application Modeler (BAM) for the code generation. You are free to customize these templates or create your own templates so that the generated application meets your requirements.

### Default actions

Generated applications have default actions, which are triggered when the user clicks on an action view on the form (such as a Toolbar icon). The actions are enabled appropriately as the BDL interactive statements in the generated application are executed.

### Default actions

The user interface for your BAM program is based on the form created for the program. When you set the **Functionality** properties on form records, you are specifying which default actions should be available to the user. For example, if the canSearch property is checked, the default actions needed to input criteria and search the database would be generated.

Actions can be programmatically enabled and disabled, hidden and shown, with methods such as `ui.Dialog.setActionActive()` and `ui.Dialog.setActionHidden()`. The text, image and other attributes of the action can be controlled with an action default file (4ad). See the *Genero Business Development Language User Guide*.

<table>
<thead>
<tr>
<th>Action Text</th>
<th>Action Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>new</td>
<td>Adds a new record in the master, whether or not the focus is in the master or a detail.</td>
</tr>
<tr>
<td>Insert</td>
<td>insert</td>
<td>Appends a new record at the end of the list of the currently selected table - either master or detail.</td>
</tr>
<tr>
<td>Append</td>
<td>append</td>
<td>Adds a new record at the location of the current selection - either in a master or a detail. The append and new actions are equivalent if the focus is in the master.</td>
</tr>
<tr>
<td>Modify</td>
<td>modify</td>
<td>Update a record.</td>
</tr>
<tr>
<td>Search</td>
<td>query</td>
<td>Search the database table; enter criteria in the relevant fields and click Accept; or click Accept on an empty form to retrieve all the records in the database table.</td>
</tr>
<tr>
<td>Delete</td>
<td>delete</td>
<td>Delete a database record.</td>
</tr>
<tr>
<td>Zoom</td>
<td>zoom</td>
<td>Activate the zoom form.</td>
</tr>
<tr>
<td>First</td>
<td>firstrow</td>
<td>Navigate to the first record.</td>
</tr>
<tr>
<td>Last</td>
<td>lastrow</td>
<td>Navigate to the last record.</td>
</tr>
<tr>
<td>Next</td>
<td>nextrow</td>
<td>Navigate to the next record.</td>
</tr>
<tr>
<td>Previous</td>
<td>prevrow</td>
<td>Navigate to the previous record.</td>
</tr>
</tbody>
</table>
About the new, append, and insert actions

The new, append, and insert actions always create a new record. The new action only creates a master record. The append and insert actions create either a master record or a detail record. The new action is a global action. The append and insert actions are contextual according to the container. The new and append actions will always appear in the Toolbar/Topmenu. The insert action will appear in the Toolbar/Topmenu if there is at least one list on the CRUD form.

See the Action rendering topic in the Mobile applications section of the Genero Business Development Language User Guide for information on the default action rendering in Android™ and iOS mobile apps.

Figure 179: Actions displayed in a desktop vs. mobile app

Report actions

Important: This feature is not supported on mobile platforms.

If a Report entity is implemented in the BA diagram, additional actions are enabled for the user. Which actions appear is controlled by the Report Options properties of the Form entity.

Figure 180: Default Toolbar with Report options

Table 96: Additional Actions for Reports

<table>
<thead>
<tr>
<th>Action Text</th>
<th>Action Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print</td>
<td>reportprint</td>
<td>Print a defined report.</td>
</tr>
<tr>
<td>Preview</td>
<td>reportpreview</td>
<td>Preview a defined report.</td>
</tr>
<tr>
<td>Print...</td>
<td>reportsetup</td>
<td>Select report and printer or export settings in Report Print Settings window. If no report design template (4rp) is available for the report, the Select Fields button is enabled to select fields for the report.</td>
</tr>
<tr>
<td>PDF</td>
<td>reportexportpdf</td>
<td>Export report to PDF format.</td>
</tr>
<tr>
<td>HTML</td>
<td>reporttextporthtml</td>
<td>Export report to HTML format.</td>
</tr>
<tr>
<td>XLS</td>
<td>reportexportxls</td>
<td>Export report to XLS (Excel) format.</td>
</tr>
<tr>
<td>RTF</td>
<td>reportexportrtf</td>
<td>Export report to RTF (MS-Word) format.</td>
</tr>
</tbody>
</table>
Modify action defaults

Create an action defaults file (.4ad) to modify the default text, image, and other attributes for the actions of the generated application.

The action defaults are initially defined by a default Genero Action Defaults file (.4ad) located in the styles sub-directory of the template set directory. Two default files are provided, dbapp.4ad for desktop applications and mobile_dbapp.4ad for mobile applications.

For details regarding the syntax of a Genero Action Defaults file, see the Genero Business Development Language User Guide.

1. Select File > New.
   The Select an item dialog opens.
2. Select either Genero BAM Desktop or Genero BAM Mobile, depending on what type of BAM application you are working with.
3. Under the Resources section, select Action Defaults (.4ad).
   An untitled Genero Action Defaults file with a .4ad extension displays in the central workspace. This file is based on an action defaults template file.
4. Modify the action defaults file by adding, modifying, or deleting actions and action attributes.
5. Save the modified file to your project.
   The default name expected for an action defaults file is either dbapp.4ad for desktop applications or mobile_dbapp.4ad for mobile applications. Saving your file using this name ensures that Genero Studio will use your file at runtime, replacing the template default. If you save your file using a different name, you must find the BLOCK in the generated code that calls the ui.Interface.loadActionDefaults() method and change the parameter name to the name of your new file.

Modify styles

Create an Genero Styles file (.4ad) to modify the styles of the generated application.

The presentation styles are initially defined by a default Genero Styles file (dbapp.4st) located in the styles sub-directory of the template set directory.

For details regarding the syntax of a Genero Styles file, see the Genero Business Development Language User Guide.

1. Select File > New.
   The Select an item dialog opens.
2. Select either Genero BAM Desktop or Genero BAM Mobile, depending on what type of BAM application you are working with.
3. Under the Resources section, select Styles (.4st).
   An untitled Genero Styles file with a .4st extension displays in the central workspace. This file is based on the default styles file for the selected template set.
4. Modify the styles file by adding, modifying, or deleting style and style attributes.
5. Save the modified file to your project.
   The default name expected for the styles file is dbapp.4st. Saving your file using this name ensures that Genero Studio will use your file at runtime, replacing the template default. If you save your file using a different name, you must find the BLOCK in the generated code that calls the ui.Interface.loadStyles() method and change the parameter name to the name of your new file.
**Default Topmenu and Toolbar**

The form for your generated program contains default action views (in a Topmenu and Toolbar) allowing the user to trigger the program actions.

**Desktop applications**

![Form1](image)

**Figure 181: Application generated by the default template**

**Mobile apps**

See the Toolbar rendering and Topmenu rendering topics in the Mobile applications section of the Genero Business Development Language User Guide for information regarding the rendering of Toolbars and Topmenus on Android™ and iOS mobile apps.

**Related tasks**

- Modify the Topmenu on page 319
  
  Create an Genero Topmenu file (.4tm) to modify the Topmenu of the generated application.

- Modify the Toolbar on page 320
  
  Create an Genero Toolbar file (.4tb) to modify the Toolbar of the generated application.

**Modify the Topmenu**

Create an Genero Topmenu file (.4tm) to modify the Topmenu of the generated application.

The Topmenu is initially defined by a default Genero Topmenu file (dbapp.4tm) located in the styles sub-directory of the template set directory.

For details regarding the syntax of a Genero Topmenu file, see the Genero Business Development Language User Guide.

1. Select **File > New**.
   
   The Select an item dialog opens.

2. Select **Genero BAM Desktop**.

3. Under the Resources section, select **Top Menu (.4tm)**.
   
   An untitled Genero Topmenu file with a .4tm extension displays in the central workspace. This file is based on the default Topmenu file for the selected template set.

4. Modify the Topmenu file by adding, modifying, or deleting actions.

5. Save the modified file to your project.

   The default name expected for the Topmenu file is dbapp.4tm. Saving your file using this name ensures that Genero Studio will use your file at runtime, replacing the template default. If you save your file using a different name, you must find the BLOCK in the generated code that calls the ui.Interface.loadTopMenu() method and change the parameter name to the name of your new file.
Modify the Toolbar

Create a Genero Toolbar file (.4tb) to modify the Toolbar of the generated application.

The Toolbar is initially defined by a default Genero Toolbar file (dbapp.4tb) located in the styles sub-directory of the template set directory.

For details regarding the syntax of a Genero Toolbar file, see the Genero Business Development Language User Guide.

1. Select File > New.
   The Select an item dialog opens.
2. Select Genero BAM Desktop.
3. Under the Resources section, select Toolbar (.4tb).
   An untitled Genero Toolbar file with a .4tb extension displays in the central workspace. This file is based on the default Toolbar file for the selected template set.
4. Modify the Toolbar file by adding, modifying, or deleting actions.
5. Save the modified file to your project.

The default name expected for the Toolbar file is dbapp.4tb. Saving your file using this name ensures that Genero Studio will use your file at runtime, replacing the template default. If you save your file using a different name, you must find the BLOCK in the generated code that calls the ui.Interface.loadToolBar() method and change the parameter name to the name of your new file.

BAM Reference

- GSTSETUPDIR on page 168
- $(generate) on page 321
- tclsh on page 322
- $(tcl) - deprecated on page 322
- $(blockpoint) on page 323
- Business Application Modeling error messages on page 325
- Business Records error messages on page 336
- Business Application Diagram error messages on page 340

BAM-specific environment variables

A subset of environment variables configure your Business Application Modeler (BAM) environment.

**DBAPP_MOBILE**

Defines whether the generated application is for a mobile device.

Set to 1 (TRUE) if the generated application is for a mobile device.

The DBAPP_MOBILE environment variable is used at the application generation level, to perform checks for unsupported functionality on mobile devices. The setting of this environment variable does not influence the generated code. The generated code will be the same regardless of the front end and regardless of the setting of the DBAPP_MOBILE environment variable.

For example, if the generated code includes an INPUT ARRAY statement, setting DBAPP_MOBILE results in warning GS-13145; otherwise there is no warning. Likewise, if the generated code includes a TreeView or ScrollGrid container, warning GS-13149 will be raised when DBAPP_MOBILE is set. The application, however, is still created, and in both cases the program will fail at runtime as the feature or container is not supported on mobile devices.

Related concepts

Environment sets on page 164
Define sets of environment variables to add to or overwrite the default environment. These environment settings are also usable by each Genero Studio client that adds a remote Genero Studio Server to its list.

**GSTSETUPDIR**

The **GSTSETUPDIR** environment variable defines the BAM application generator template directory. Changing this variable launches synchronization from the server and reloads the templates.

Select the default environment set or create a new one that includes the **GSTSETUPDIR** specifying the location of the template directory to be used.

**Related concepts**

[Environment sets on page 164](#)

Define sets of environment variables to add to or overwrite the default environment. These environment settings are also usable by each Genero Studio client that adds a remote Genero Studio Server to its list.

---

**(generate)**

The $(generate) command creates an intermediary XML file from modeled entities.

**Syntax**

\[
$(generate) \ [\text{options}] \\
\]

1. **options** are described in Table 97: $(generate) options on page 321.

**Options**

**Table 97: $(generate) options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-oldModelFormat</td>
<td>Generate XML file with prior format.</td>
</tr>
<tr>
<td></td>
<td>This feature is deprecated. Use depth attribute on Item elements in settings.agconf instead.</td>
</tr>
<tr>
<td>-depth depthNumber</td>
<td>For example, in a BA diagram specify the number of relations to traverse while generating the model. Outgoing relations are traversed. The model always contains the incoming and outgoing relation for the current item, but if the depth limit is reached, the target item definition is not generated. The depth is an integer starting from 0, or unlimited to traverse the complete application.</td>
</tr>
<tr>
<td>-ba baFilePath</td>
<td>Path to the BA diagram (4ba). The path of the 4ba file in the current project workspace is stored in the $(BAFilePath) project manager variable.</td>
</tr>
<tr>
<td>-o outputFilePathPath</td>
<td>Path of the generated XML file. <em>filename</em> is the generated XML file name. Uses the filename, replacing the extension with xml.</td>
</tr>
</tbody>
</table>

**Usage**

The $(generate) command is used in build rules for generated programs. [Predefined node variables on page 436](#) can be used in the command.

\[
$(generate) -ba "$(BAFilePath)" "$(InputPath)"
\]
Related concepts

$(generate) model on page 1167
The Business Application Modeler generates intermediate XML files. Examine the XML schema associated with an intermediate file to understand the content of the generated file.

Command line options for build, link, execution rules on page 416
Special command line options can be used for build, link, and execution rules.

Predefined node variables on page 436
Each Project Manager node has a defined set of variables containing values determined at runtime.

How code is generated on page 225
When you build an application from a Business Application diagram, the build rules define the various files that are input into the Code Generation Engine and the application code files that are output.

tclsh
The tclsh executable generates the final file by using both a Tcl template file and the intermediary XML file created by the $(generate) command.

The executable is GSTDIR/bin/tclsh (for UNIX™) and GSTDIR/bin/tclsh.exe (for Windows®).

Syntax

tclsh script.tcl [arguments]

1. Name of the intermediary XML file created by the $(generate) on page 321 command for the current item.

Usage
tclsh is used in build rules for generated programs. Predefined node variables on page 436 can be used in the command.

tclsh "$(TemplateDir)/tpl/genprg.tcl" "$(InputDir)/$(InputBaseName).xml"

Related concepts

Predefined node variables on page 436
Each Project Manager node has a defined set of variables containing values determined at runtime.

How code is generated on page 225
When you build an application from a Business Application diagram, the build rules define the various files that are input into the Code Generation Engine and the application code files that are output.

$(tcl) - deprecated
$(tcl) is deprecated, tclsh is used instead of $(tcl) to generate files. The $(tcl) command launches the TCL interpreter and generates the final file by using both a Tcl template file and the intermediary XML file created by the $(generate) command.

Syntax

$tcl [options] xml_filename

1. options are described in Table 98: $(tcl) options on page 323.
2. Name of the intermediary XML file created by the $(generate) on page 321 command for the current item.
Options

Table 98: $(tcl) options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-tpl templatefilename</td>
<td>Tcl template file name.</td>
</tr>
<tr>
<td>-o</td>
<td>Path of the file to which to write output.</td>
</tr>
</tbody>
</table>

Alternative syntax

Syntax similar to the command line Tcl Interpreter is available from version 2.41.

$(tcl) templatename templatearguments

Usage

The $(tcl) is used in build rules for generated programs. Predefined node variables on page 436 can be used in the command.

$(tcl) "$(TemplateDir)/tpl/main.tcl" "$(InputDir)/$(InputBaseName).xml"

Related concepts

tclsh on page 322
The tclsh executable generates the final file by using both a Tcl template file and the intermediary XML file created by the $(generate) command.

Predefined node variables on page 436
Each Project Manager node has a defined set of variables containing values determined at runtime.

How code is generated on page 225
When you build an application from a Business Application diagram, the build rules define the various files that are input into the Code Generation Engine and the application code files that are output.

$(blockpoint)

The $(blockpoint) command manages user added code by extracting or injecting code between BLOCK and POINT tags in a generated 4gl file.

Syntax

$(blockpoint) [options] "filename(s)"

1. options are described in Table 99: $(blockpoint) options on page 323.
2. filename is the generated 4gl file(s) separated by a space.

Options

Table 99: $(blockpoint) options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-extract</td>
<td>Extract diff between filename and generated part of filename.code file.</td>
</tr>
<tr>
<td>-storeGenerated</td>
<td>Store the generated part of filename.code with the content of filename.</td>
</tr>
</tbody>
</table>
Option | Description
--- | ---
-inject | Inject the diff part of `filename.code` in the `filename`.
-commentStart | Comment start pattern.
-commentStart2 | Line comment pattern.
-commentEnd | Comment end pattern.
-code \[.code file path\] | Specifies the name of the `.code` file. If there is only one generated source file (4gl), the `.code` file uses (by default) the same name as the source file, otherwise - code is mandatory.

**Usage**

The `$(blockpoint)` command is used in build rules for generated programs. Predefined node variables on page 436 can be used in the command.

```bash
$(blockpoint) -code "$(InputDir)/$(InputBaseName).code"
-extract "$(InputDir)/$(InputBaseName).4gl"
```

```bash
$(blockpoint) -storeGenerated -code "$(InputDir)/$(InputBaseName).code"
-inject "$(InputDir)/$(InputBaseName).4gl"
```

**Related concepts**

Using POINTs and BLOCKs on page 313

POINT and BLOCK sections are the areas in the generated code where you can add your own code.

Predefined node variables on page 436

Each Project Manager node has a defined set of variables containing values determined at runtime.

How code is generated on page 225

When you build an application from a Business Application diagram, the build rules define the various files that are input into the Code Generation Engine and the application code files that are output.

**File extensions and BAM**

Files involved in Business Application Modeling use specific file extensions. Knowing what the acronyms within a file extension reference can help you interpret each of the BAM-related file extension.

A file extension is comprised of acronyms that represent the different aspects of a file.

**Table 100: Acronyms within a file extension**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>What it represents</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fd</code></td>
<td>Form Designer</td>
</tr>
<tr>
<td><code>m</code></td>
<td>Managed Form (CRUD)</td>
</tr>
<tr>
<td><code>ws</code></td>
<td>Web Service</td>
</tr>
<tr>
<td><code>j</code></td>
<td>JSON</td>
</tr>
<tr>
<td><code>prg</code></td>
<td>Program file that includes a MAIN function</td>
</tr>
<tr>
<td><code>z</code></td>
<td>Zoom Form</td>
</tr>
<tr>
<td><code>c</code></td>
<td>Custom Form</td>
</tr>
</tbody>
</table>
The following is a list of BAM-related file extensions. The category column relates to the categories used to organize the options when creating a new file with File > New.

**Table 101: BAM-related file extensions**

<table>
<thead>
<tr>
<th>Category</th>
<th>File Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>.4pw</td>
<td>Project file. This extension is not BAM-specific.</td>
</tr>
<tr>
<td>Application Modeling</td>
<td>.4ba</td>
<td>Business Application Diagram file</td>
</tr>
<tr>
<td>Database</td>
<td>.4dbx</td>
<td>Database meta-schema.</td>
</tr>
<tr>
<td>Sources</td>
<td>.4fdm</td>
<td>CRUD Form file</td>
</tr>
<tr>
<td>Sources</td>
<td>.4fdz</td>
<td>Zoom form file</td>
</tr>
<tr>
<td>Sources</td>
<td>.4rd</td>
<td>Report Data file</td>
</tr>
<tr>
<td>Sources</td>
<td>.4ws</td>
<td>SOAP Web Service file</td>
</tr>
<tr>
<td>Sources</td>
<td>.4fdmws</td>
<td>CRUD Form and SOAP Web Service file</td>
</tr>
<tr>
<td>Sources</td>
<td>.4fdzws</td>
<td>Zoom Form and SOAP Web Service file</td>
</tr>
<tr>
<td>Sources</td>
<td>.4wsj</td>
<td>JSON Web Service file</td>
</tr>
<tr>
<td>Sources</td>
<td>.4fdmwsj</td>
<td>CRUD Form and JSON Web Service file</td>
</tr>
<tr>
<td>Sources</td>
<td>.4fdzwsj</td>
<td>Zoom Form and JSON Web Service file</td>
</tr>
<tr>
<td>Sources</td>
<td>.4fdc</td>
<td>Custom form file</td>
</tr>
<tr>
<td>Sources</td>
<td>.4wsprg</td>
<td>SOAP Web Service Server file</td>
</tr>
<tr>
<td>Sources</td>
<td>.4wsjprg</td>
<td>JSON Web Service Server file</td>
</tr>
<tr>
<td>Resources</td>
<td>.4ad</td>
<td>Action Defaults file. This extension is not BAM-specific.</td>
</tr>
<tr>
<td>Resources</td>
<td>.4tm</td>
<td>Top Menu file. This extension is not BAM-specific.</td>
</tr>
<tr>
<td>Resources</td>
<td>.4tb</td>
<td>Toolbar file. This extension is not BAM-specific.</td>
</tr>
<tr>
<td>Resources</td>
<td>.4st</td>
<td>Presentation Style file. This extension is not BAM-specific.</td>
</tr>
</tbody>
</table>

**Business Application Modeling error messages**

A list of BAM error messages. For messages that are not self-explanatory, additional information is provided.

**Table 102: Business Application Modeling Error Messages**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-13001</td>
<td>Cannot load file.</td>
</tr>
<tr>
<td></td>
<td>The file cannot be loaded; depending on the error, the message can change:</td>
</tr>
<tr>
<td></td>
<td>• Unknown node: an unknown node is present in the settings file</td>
</tr>
<tr>
<td></td>
<td>• Cannot register node: a node cannot be added to the file format (it's invalid or already present)</td>
</tr>
<tr>
<td></td>
<td>• Empty extension: a required file extension is empty.</td>
</tr>
<tr>
<td></td>
<td>Check file path, format and permissions.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-13002</td>
<td>Cannot save file.</td>
</tr>
<tr>
<td></td>
<td>Check file path, format and permissions.</td>
</tr>
<tr>
<td>GS-13003</td>
<td>Template not found.</td>
</tr>
<tr>
<td></td>
<td>Check template directory path.</td>
</tr>
<tr>
<td>GS-13004</td>
<td>Unknown item %1.</td>
</tr>
<tr>
<td></td>
<td>The settings.agconf template file format is incorrect. Validate it using the XML schema (or open it in Code Editor) and fix the errors.</td>
</tr>
<tr>
<td>GS-13005</td>
<td>Missing version in settings.agconf</td>
</tr>
<tr>
<td></td>
<td>The settings.agconf file version attribute is not present.</td>
</tr>
<tr>
<td></td>
<td>Add a version attribute corresponding to the product.</td>
</tr>
<tr>
<td>GS-13006</td>
<td>settings.agconf version %1 is not supported, use version %2.</td>
</tr>
<tr>
<td></td>
<td>Modify settings.agconf to match version 2 XML schema.</td>
</tr>
<tr>
<td>GS-13007</td>
<td>%1 BLOCK(S) or POINT(S) end tag is(are) missing.</td>
</tr>
<tr>
<td></td>
<td>Fix the BLOCK / POINT.</td>
</tr>
<tr>
<td>GS-13008</td>
<td>Incorrect end BLOCK or POINT tag type.</td>
</tr>
<tr>
<td></td>
<td>Fix the BLOCK / POINT.</td>
</tr>
<tr>
<td>GS-13009</td>
<td>End BLOCK or POINT tag does not correspond to an open tag.</td>
</tr>
<tr>
<td></td>
<td>Fix the BLOCK / POINT.</td>
</tr>
<tr>
<td>GS-13010</td>
<td>%1 with name %2 is already defined.</td>
</tr>
<tr>
<td></td>
<td>Rename the BLOCK / POINT.</td>
</tr>
<tr>
<td>GS-13011</td>
<td>A BLOCK or POINT cannot be a child of a POINT tag.</td>
</tr>
<tr>
<td></td>
<td>Rename the BLOCK / POINT.</td>
</tr>
<tr>
<td>GS-13012</td>
<td>Renamed %1: %2 to %3.</td>
</tr>
<tr>
<td></td>
<td>Preprocessor message that a BLOCK / POINT has been renamed:</td>
</tr>
<tr>
<td></td>
<td>%1 = BLOCK or POINT</td>
</tr>
<tr>
<td></td>
<td>%2 = old BLOCK / POINT name</td>
</tr>
<tr>
<td></td>
<td>%3 = new BLOCK / POINT name</td>
</tr>
<tr>
<td>GS-13013</td>
<td>Lost %1: %2.</td>
</tr>
<tr>
<td></td>
<td>%1 = BLOCK or POINT</td>
</tr>
<tr>
<td></td>
<td>%2 = BLOCK / POINT name</td>
</tr>
<tr>
<td>GS-13014</td>
<td>Adding entries outside BLOCK or POINT not allowed.</td>
</tr>
<tr>
<td></td>
<td>Add a BLOCK / POINT in the templates or remove your code.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-13015</td>
<td>$(agcomp) is deprecated, please prefer using $(generate), tclsh and $(blockpoint) commands. Modify the build rule to use the new commands.</td>
</tr>
</tbody>
</table>
| GS-13016 | Modified %1: %2.  
%1 = BLOCK or POINT  
%2 = BLOCK / POINT name |
| GS-13017 | Unknown property.  
An unknown property has been set in the settings file. Remove the property or change its name. |
| GS-13019 | Cannot open file: extraction load failed.  
Check the rights on the generated source file. |
| GS-13020 | Cannot open file: injection save failed.  
Check the rights on the generated source file. |
| GS-13021 | Cannot open file: update load failed.  
Check the rights on the generated source file. |
| GS-13022 | Cannot open file: update save failed.  
Check the rights on the code file. |
| GS-13023 | Code file load failed.  
Check the rights on the code file or check that the code file contains valid XML content. |
| GS-13024 | Unknown argument %1 to Application Generator Block & Point task.  
The command $(blockpoint) contains an unknown argument in the build rule. Check the argument %1 of $(blockpoint) command in the build rule. |
| GS-13025 | Missing -depth int argument to Application Generator compilation task.  
The Application Generator build rule requires the -depth argument in the $generate command. Add the -depth argument to the $(generate) command in Application Generator build rules. |
| GS-13026 | Invalid -depth argument for Application Generator compilation task, required positive integer or 'unlimited'. The $(generate) command requires a positive number (or keyword unlimited). Fix the argument in the build rule command. |
| GS-13027 | Unknown argument %1 to Application Generator compilation task.  
Command or tclsh or $(agcomp) contains an unknown argument in the build rule. Check the argument %1 of $(generate) or tclsh or $(agcomp) command in the build rules. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
</table>
| GS-13028| Invalid `-endComment string argument, it cannot be used without `-startComment string argument.  
The command $(blockpoint) contains an `-endComment argument without `-startComment argument in the build rules. Remove the `-endComment argument or add a `-startComment argument in the build rule. |
| GS-13029| Error decoding `%1 using codec %2 (encoding=%3)  
The file contents cannot be decoded with the codec, the encoding specified in the environment does not correspond to the file and it cannot be read. Change the encoding so that it supports all the file characters. |
| GS-13030| Error encoding `%1 using codec %2 (encoding=%3)  
The file contents cannot be encoded with the codec, the encoding specified in the environment does not correspond to the file and it cannot be written to. Change the encoding so that it supports all the file characters. |
| GS-13031| Missing codec for encoding `%1  
There is no default text codec for the specified encoding. Use another alias for this encoding, update encodingMap.xml, or add a new POSIX charmap. |
| GS-13032| Malformed BLOCK/POINT start tag  
The BLOCK/POINT start tag has a wrong syntax. Restore the right syntax. |
| GS-13033| Unexpected characters outside text blocks : `%1  
Some characters have changed outside the topmost block, which is not supported. Insert a new toplevel block around the code you want to modify and regenerate the code. |
| GS-13034| Missing argument `-code in the $(BlockPoint) command  
The $(BlockPoint) command is used with multiple generated files, the `-code argument is mandatory. Add the `-code argument. |
| GS-13035| Missing files in the $(BlockPoint) command  
The $(BlockPoint) command is used without generated files arguments. Add the generated files list. |
| GS-13036| `%1 contains code not managed in the $(BlockPoint) command  
Some file managed by the .code were not updated by the $(BlockPoint) command, the .code is not completely up to date. Add the missing files to the command line in the build rule or remove the old file from the .code if they are no longer useful. |
| GS-13037| .code file version `%1 is not supported  
The current version of Genero Studio does not support this version of .code file. Upgrade Genero Studio. |
| GS-13038| .code file encoding `%1 differs with current one  
The .code file encoding is not the same as the build encoding, the resulting files may be incorrect.  
Change the build encoding (the LANG variable for example) to match the .code file encoding or rewrite the .code file with the current encoding. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-13039</td>
<td>Resolve conflict before compiling</td>
</tr>
<tr>
<td></td>
<td>The code file id does not match the source file id, thus a resulting</td>
</tr>
<tr>
<td></td>
<td>conflict file. You must merge the differences manually. This may occur</td>
</tr>
<tr>
<td></td>
<td>after updating from the SVN repository.</td>
</tr>
<tr>
<td></td>
<td>Merge the .conflict file and the source file, then delete the .conflict</td>
</tr>
<tr>
<td></td>
<td>file.</td>
</tr>
<tr>
<td>GS-13040</td>
<td>Cannot write conflict file</td>
</tr>
<tr>
<td></td>
<td>The code file id does not match the source file id, thus a resulting</td>
</tr>
<tr>
<td></td>
<td>conflict file which cannot be written to the disk.</td>
</tr>
<tr>
<td></td>
<td>Check the directory permissions.</td>
</tr>
<tr>
<td>GS-13041</td>
<td>Ignored unknown attribute '%1' on element '%2'</td>
</tr>
<tr>
<td></td>
<td>In the settings.agconf file, an attribute has been defined where it is</td>
</tr>
<tr>
<td></td>
<td>not expected. If the attribute is not required, remove it. If the</td>
</tr>
<tr>
<td></td>
<td>attribute is required but has an error (a spelling error, for example),</td>
</tr>
<tr>
<td></td>
<td>fix the error.</td>
</tr>
<tr>
<td>GS-13042</td>
<td>'-depth' argument of $(generate) command is deprecated, use 'depth'</td>
</tr>
<tr>
<td></td>
<td>attribute on 'Item' elements in settings.agconf.</td>
</tr>
<tr>
<td></td>
<td>Some Business Application deprecated feature has been used.</td>
</tr>
<tr>
<td></td>
<td>Refer to the migration notes in the manual to update the project. In this</td>
</tr>
<tr>
<td></td>
<td>example, modify the settings.agcong to add the depth, and edit the build</td>
</tr>
<tr>
<td></td>
<td>rules to remove the depth.</td>
</tr>
<tr>
<td>GS-13043</td>
<td>Invalid Template directory '%1'</td>
</tr>
<tr>
<td></td>
<td>The Business Application template directory setting is set incorrect.</td>
</tr>
<tr>
<td></td>
<td>Solution:</td>
</tr>
<tr>
<td></td>
<td>• Remove the template directory setting if useless.</td>
</tr>
<tr>
<td></td>
<td>• Change the path for a valid directory containing Business Application</td>
</tr>
<tr>
<td></td>
<td>settings.</td>
</tr>
<tr>
<td>GS-13044</td>
<td>%1 %2 index out of range (%3) current index is %4</td>
</tr>
<tr>
<td></td>
<td>Business Application build command $(BlockPoint) failed, there are</td>
</tr>
<tr>
<td></td>
<td>inconsistencies between the source files and the code files. This error</td>
</tr>
<tr>
<td></td>
<td>should not happen during normal operations, but may appear in case of</td>
</tr>
<tr>
<td></td>
<td>errors during the process of migration between different versions.</td>
</tr>
<tr>
<td></td>
<td>If you receive this message, contact support for help during the</td>
</tr>
<tr>
<td></td>
<td>migration.</td>
</tr>
<tr>
<td>GS-13100</td>
<td>The 'Open Mode' property is not set.</td>
</tr>
<tr>
<td></td>
<td>The Open Mode property value is empty. The available values are: DISPLAY,</td>
</tr>
<tr>
<td></td>
<td>MODIFY, ADD, SEARCH and EMPTY. Check settings.agconf and ensure that</td>
</tr>
<tr>
<td></td>
<td>the initialValue attribute of the &lt;DynamicProperty&gt; node having name =</td>
</tr>
<tr>
<td></td>
<td>&quot;openMode&quot; contains one of the available values.</td>
</tr>
<tr>
<td>GS-13101</td>
<td>The 'Action' property is mandatory.</td>
</tr>
<tr>
<td></td>
<td>The Action property is mandatory when a relation is defined between</td>
</tr>
<tr>
<td></td>
<td>modules (4fdm to 4fdm) or between module and zoom (4fdm to 4fdz) and the</td>
</tr>
<tr>
<td></td>
<td>relation type=Relation. Check that the Type property of the relation is</td>
</tr>
<tr>
<td></td>
<td>Relation. Check that the Action property of the relation is not empty.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-13102</td>
<td>No report file is defined. Default layout will be used.</td>
</tr>
<tr>
<td></td>
<td>If the Report File property is empty, the default layout (ASCII mode) will be used when a relation is defined between a module and a report (4fdm to 4rd) and the relation type=ReportRelation. This is a warning message. Check that the Type property of the relation is ReportRelation. Check that the Report File property of the relation is not empty.</td>
</tr>
<tr>
<td>GS-13103</td>
<td>The 'Type' property of the relation is invalid.</td>
</tr>
<tr>
<td></td>
<td>The type of relation is not valid between entities: module (4fdm), program (4prg), zoom (4fdz), report (4rd). Check the Type property of the supported relation and ensure that the relation is supported.</td>
</tr>
<tr>
<td>GS-13104</td>
<td>The relation is not supported.</td>
</tr>
<tr>
<td></td>
<td>Check the entity relations.</td>
</tr>
<tr>
<td>GS-13105</td>
<td>Any relation to a program is not supported.</td>
</tr>
<tr>
<td></td>
<td>Check the entity relations.</td>
</tr>
<tr>
<td>GS-13106</td>
<td>The 'Report file' property contains an absolute path. Prefer a relative path. Absolute paths are not recommended because the project will not be portable. Use a relative path. Be sure that your Genero environment variables are correctly set to search for your resource files.</td>
</tr>
<tr>
<td>GS-13107</td>
<td>A unique key field can only be defined on the master table.</td>
</tr>
<tr>
<td></td>
<td>All fields making up the unique key of a record must be fields of the master table of the record. Uncheck the Unique Key property of fields which are not of the master table of the record.</td>
</tr>
<tr>
<td>GS-13108</td>
<td>Duplicate table name in the FROM clause.</td>
</tr>
<tr>
<td></td>
<td>The joins between a pair of tables must have the same join operator. Check that there are not multiple join operators for the same pair of tables.</td>
</tr>
<tr>
<td>GS-13109</td>
<td>Field updated via ascending lookup is mandatory in the record.</td>
</tr>
<tr>
<td></td>
<td>The field that is updated via an ascending lookup is mandatory in the master table of the record. Add the field to the master table of the record. Currently, the field is only defined in the Query clause of the record. If you don't want to see this field on your form, use the phantom widget. Or, delete the lookup property of the field which triggers the ascending lookup.</td>
</tr>
<tr>
<td>GS-13110</td>
<td>Multiple master records are not supported. Define a relation between records.</td>
</tr>
<tr>
<td></td>
<td>When there are several records in a managed Form, only one can be the master. This error occurs when at least one record is not linked by a relation. In the Records view, check that all records are linked by a relation.</td>
</tr>
<tr>
<td>GS-13111</td>
<td>Invalid relation, primaryField and foreignField must have same number of fields.</td>
</tr>
<tr>
<td></td>
<td>A relation must have the same number of primary fields and foreign fields. Modify the relation fields so that a primary field corresponds to each foreign field.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>GS-13112</td>
<td>The 'zoom' action in the Toolbar/Topmenu won't be generated, an action is already named 'zoom'. Rename your action to a name other than zoom.</td>
</tr>
<tr>
<td>GS-13113</td>
<td>Several relations ($1) use the same action name ($2 - $3). Several relations use the same combination of Action/Source Record property values. $1 - Number of relations having the same combination of Action/Source Record. $2 - Action name. $3 - List of record names. Rename Action or change the Source Record to have a unique couple.</td>
</tr>
<tr>
<td>GS-13114</td>
<td>SQL statement 'FULL OUTER JOIN' is vendor proprietary SQL syntax.</td>
</tr>
<tr>
<td>GS-13115</td>
<td>Invalid filename. A filename must be a BDL identifier. The filename contains invalid characters. Identifiers must confirm to these rules: • It must include at least one character, without any limitation in size. • Only ASCII letters, digits, and underscore (_) symbols are valid. • No blanks, hyphens, and other non-alphanumeric characters. • The initial character must be a letter or an underscore. • It is recommended to always write identifiers in lower case. Rename the file to a valid file name. If the file was generated, renaming the object may not be enough; you will have to rename the file itself to match the new object name in Project Manager.</td>
</tr>
<tr>
<td>GS-13116</td>
<td>Several comboboxes use the same initializer name. Make all initializer names unique.</td>
</tr>
<tr>
<td>GS-13117</td>
<td>A ComboBox is possibly not initialized. Set the initializer or the items property of the ComboBox.</td>
</tr>
<tr>
<td>GS-13118</td>
<td>Table $1 unused in query. A database table is used in the record, but there is no join for it in the query. Add a join for the table in the record query.</td>
</tr>
<tr>
<td>GS-13119</td>
<td>Duplicated joins are forbidden in the query. Remove the duplicated joins.</td>
</tr>
<tr>
<td>GS-13120</td>
<td>Building $1 generates an XML file (version=$2) which cannot be handled by the templates (version=$3). The build process of the file ($1) has been aborted because the intermediate XML file has a version ($2) which mismatches the template version ($3). Use a valid template set defined in Application Generator Preferences. Verify build rules for the appropriate file type.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>GS-13121</td>
<td>The $1 property is not supported. In the Business Application diagram, the openMode or defaultMode property defined on a relation between a CRUD Form and a Zoom Form is not supported. Only DISPLAY and SEARCH values are currently supported on a Zoom. Change the openMode or defaultMode property to a supported value.</td>
</tr>
</tbody>
</table>
| GS-13122 | The $1 property is missing in settings.agconf. The build process has been aborted because the $1 dynamicProperty is missing in the settings.agconf configuration file. Check settings.agconf and confirm:  
• The `<DynamicProperty>` node with name="$1" is defined in the `<BusinessApplication>` section.  
• The dynamicProperties attribute of the item raising the error contains $1. |
| GS-13123 | No CRUD function will be generated for the $1 table because no primary key has been defined. A primary key needs to be defined for the given table in order to have its CRUD functions generated when compiling the database schema. |
| GS-13124 | Record functionalities are incompatible with the $1 property value ($2).  
On a CRUD Form, if the openMode or defaultMode property is:  
• ADD, the master record must have the functionality canAdd activated.  
• EMPTY, the master record must have the functionality canEmpty activated.  
• DISPLAY, all records must have the functionality canDisplay activated.  
• MODIFY, at least one record must have the functionality canModify activated.  
On a CRUD Form or Zoom Form, if the openMode or defaultMode property is SEARCH, at least one record must have the functionality canSearch activated.  
On incoming relations of a CRUD Form or Zoom Form, the openMode and defaultMode property must follow the same rules as those forms.  
To fix the error, change the openMode or defaultMode property value or change the functionality of the record. |
<p>| GS-13125 | $1 doesn't exist or is not unique$2, please have a look in the XML intermediate file. An XPath cannot be resolved. Check the XML intermediate file. |
| GS-13126 | The number of source fields defined on the relation doesn't match the number of unique key fields defined in the destination item. Change the number of source fields defined on the relation. |
| GS-13127 | Destination fields defined on a relation to a zoom are unused in the code generation. The destination fields defined on a Zoom relation will not appear in the generated 4gl code. |</p>
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-13128</td>
<td>The table 'seqreg' is missing in your database schema, therefore SERIAL fields cannot be handled by the templates. A database schema field of datatype SERIAL can only be handled by using the seqreg table. This table contains a list of table names and values, a value is the last SERIAL created for a given table name. In case the couple (table name, last value) does not exist, a new record will be created and the SERIAL value will start at 1. Create the table seqreg in your database schema file.</td>
</tr>
<tr>
<td>GS-13129</td>
<td>Multiple SERIAL fields for the table $1 cannot be handled by the templates. Only one SERIAL field can be managed per database table. This is inherent to the seqreg table. Modify the table in your database schema file.</td>
</tr>
<tr>
<td>GS-13130</td>
<td>Business Application diagram is missing in the project. Add a Business Application Diagram to the project.</td>
</tr>
<tr>
<td>GS-13131</td>
<td>Unique key must be a database primary or secondary key of the master table. All fields making up the unique key of a record must be either a database primary key or secondary key of the master table (unique constraint). Change the unique key property of fields which are not a database primary key or secondary key of the master table or update the 4dbx schema.</td>
</tr>
<tr>
<td>GS-13132</td>
<td>Missing master table for $1 record. Select a table in the master table property.</td>
</tr>
<tr>
<td>GS-13133</td>
<td>The field $1 defined on the relation does not exist in Records view. In the Business Application diagram, when a relation is defined between items, the Source Field and the Destination Field must exist in the Records view. Check the Source Field and/or Destination Field property of the relation or open the appropriate item and add the missing fields.</td>
</tr>
<tr>
<td>GS-13134</td>
<td>Invalid relation, Source and Destination must have same number of fields. In the Business Application diagram, a relation between Forms must have the same number of fields in the Source Field and Destination Field. Check the Source Field and/or Destination Field property of the relation.</td>
</tr>
<tr>
<td>GS-13136</td>
<td>The Service Name property is mandatory. The property Service Name of a WebService item must have a value in the Business Application Diagram. Set the Service Name property value.</td>
</tr>
<tr>
<td>GS-13137</td>
<td>The Namespace property is mandatory. The property Namespace of a WebService Server item in a Business Application diagram must have a value. Set the Namespace property value.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-13138</td>
<td>Several web services use the same name.</td>
</tr>
<tr>
<td></td>
<td>Any web service registered to the same Web Service Server must have an unique name.</td>
</tr>
<tr>
<td></td>
<td>Ensure that the property Name of a web service is unique.</td>
</tr>
<tr>
<td>GS-13139</td>
<td>The $1 action is a reserved action name.</td>
</tr>
<tr>
<td></td>
<td>In the Business Application Diagram, when a relation is defined between items, the Action property can not be a reserved action name. The list of reserved action names is defined in the action defaults file (dbapp.4ad).</td>
</tr>
<tr>
<td></td>
<td>Set the value of the Action property to a non-reserved action name.</td>
</tr>
<tr>
<td>GS-13140</td>
<td>Unexpected TCL script error: $1</td>
</tr>
<tr>
<td></td>
<td>An unexpected internal error occured during code generation. Contact your local Four Js support center.</td>
</tr>
<tr>
<td>GS-13141</td>
<td>$1 - Check LANG variable or add entry in file encoding.tcl where $1 is the value of the unsupported encoding</td>
</tr>
<tr>
<td></td>
<td>The LANG environment variable is set, but the encoding part is either invalid (change the value of the LANG environment variable) or not present in the supported encoding array (add an entry in encoding.tcl).</td>
</tr>
<tr>
<td>GS-13142</td>
<td>All fields must be part of the same record.</td>
</tr>
<tr>
<td></td>
<td>Choose fields from the same record.</td>
</tr>
<tr>
<td>GS-13143</td>
<td>The record $1 defined on the relation does not exist in Records view.</td>
</tr>
<tr>
<td></td>
<td>In the Business Application diagram, when a relation is defined between items, the Record Source must exist in the Records view. Check the Source Record property of the relation or open the source entity and add the missing record.</td>
</tr>
<tr>
<td>GS-13144</td>
<td>The 'Source Record' property is missing.</td>
</tr>
<tr>
<td></td>
<td>The Source Record property is missing in the Business Application diagram. When a relation is defined between items, if the Row Bound property is checked on an outgoing relation, the Record Source must be defined if there is more than one record in the form. It is implicit if there is only one record.</td>
</tr>
<tr>
<td></td>
<td>Check the Source Record property of the relation or uncheck the Row Bound property.</td>
</tr>
<tr>
<td>GS-13145</td>
<td>The $1 functionality is not supported for $2 container on mobile devices.</td>
</tr>
<tr>
<td></td>
<td>Some functionality is not supported by the template. For example canAdd, canModify, and canSearch are not supported for tables.</td>
</tr>
<tr>
<td></td>
<td>Uncheck the functionality.</td>
</tr>
<tr>
<td>GS-13146</td>
<td>Several records $1 are candidate for the 'Source Record' property.</td>
</tr>
<tr>
<td></td>
<td>Remove the ambiguity and choose a record.</td>
</tr>
<tr>
<td>GS-13147</td>
<td>The 'Row Bound' property is not supported for GRID container.</td>
</tr>
<tr>
<td></td>
<td>Uncheck the Row Bound property or choose the table container in the form.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-13148</td>
<td>The 'Source Field' property is missing. The position Source Field is required when the destination of the relation is a zoom.</td>
</tr>
<tr>
<td>GS-13149</td>
<td>The '$1' container is not supported on mobile devices. Some containers such Tree and ScrollGrid are not supported on mobile devices. Change container to a supported container.</td>
</tr>
<tr>
<td>GS-13150</td>
<td>'Display, Add, Modify, Search are all disabled for record '$1' This message is a warning, not an error. It simply tells you that all functionality is disabled for the record in question. Verify that having no functionality selected for the record is intentional.</td>
</tr>
<tr>
<td>GS-13151</td>
<td>'The '$1' aggregation function is not supported for this field' Ensure the data type of the field is compatible with the aggregation function:</td>
</tr>
<tr>
<td></td>
<td>• AVG and SUM can only be performed on Numeric fields.</td>
</tr>
<tr>
<td></td>
<td>• COUNT and DISTINCT COUNT cannot be performed on a Byte field.</td>
</tr>
<tr>
<td></td>
<td>• MAX and MIN cannot be performed on a Byte or Boolean field.</td>
</tr>
<tr>
<td>GS-13152</td>
<td>'Missing $1 file' The Java file argument of the 'schemagen' tool is missing. This error can occur on a build of a ReportData file(.4rdj) with Java. Ensure that the generated Java file exists.</td>
</tr>
<tr>
<td>GS-13153</td>
<td>'$1' An error occurred when the external tool used to create the schema file was executing. This error can occur on a build of a ReportData file(.4rdj) with Java.</td>
</tr>
<tr>
<td>GS-13154</td>
<td>The '$1' property is unused in the code generation In the Business Application diagram, a warning is raised when a property is unused in code generation. Reset the unused property to the default value to remove this warning.</td>
</tr>
<tr>
<td>GS-13155</td>
<td>Database connection not generated in Program. The databaseName property is not set on first Form If the databaseName property is not defined on the first form defined of the outgoing relation of a program, then that database connection is not generated in the program.</td>
</tr>
<tr>
<td>GS-13165</td>
<td>Outgoing Form from '$1' not implemented In the Business Application diagram, a program has an outgoing relationship to an unimplemented form. Implement the form.</td>
</tr>
<tr>
<td>GS-13166</td>
<td>Outgoing Web Service from '$1' not implemented In the Business Application diagram, a Web Service Server has an outgoing relationship to an unimplemented Web Service. Implement the Web Service.</td>
</tr>
</tbody>
</table>
## Business Records error messages

A list of Business Records error messages. For messages that are not self-explanatory, additional information is provided.

### Table 103: Business Records Error Messages

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-24001</td>
<td><strong>Error loading file</strong> file.</td>
</tr>
<tr>
<td></td>
<td>An error occurred loading the file.</td>
</tr>
<tr>
<td></td>
<td>Check the file name and permissions .</td>
</tr>
<tr>
<td>GS-24002</td>
<td>Malformed XML.</td>
</tr>
<tr>
<td></td>
<td>XML file content is invalid.</td>
</tr>
<tr>
<td></td>
<td>Select the correct file or correct the XML.</td>
</tr>
<tr>
<td>GS-24003</td>
<td><strong>File file not found in Business Application diagram (&lt;BA file name&gt;).</strong></td>
</tr>
<tr>
<td></td>
<td>A referenced element is not found in the Business Application diagram.</td>
</tr>
<tr>
<td></td>
<td>Select the correct file.</td>
</tr>
<tr>
<td>GS-24004</td>
<td><strong>Invalid value for %1 property.</strong></td>
</tr>
<tr>
<td></td>
<td>Check the property syntax in the documentation and correct any errors.</td>
</tr>
<tr>
<td>GS-24005</td>
<td><strong>Conflicted item.</strong></td>
</tr>
<tr>
<td></td>
<td>The Business Application diagram and Business Record unique id do not match.</td>
</tr>
<tr>
<td></td>
<td>Open the 4ba file and resolve the conflict.</td>
</tr>
<tr>
<td>GS-24201</td>
<td><strong>Missing master Record.</strong></td>
</tr>
<tr>
<td></td>
<td>The document must contain at least one record that is master.</td>
</tr>
<tr>
<td></td>
<td>Create a record.</td>
</tr>
<tr>
<td>GS-24202</td>
<td><strong>Unique query key must be set.</strong></td>
</tr>
<tr>
<td></td>
<td>At least one field must be declared the unique key in a record.</td>
</tr>
<tr>
<td></td>
<td>Set a unique key.</td>
</tr>
<tr>
<td>GS-24203</td>
<td><strong>Record used in relation must be active.</strong></td>
</tr>
<tr>
<td></td>
<td>Change the record to active, or remove the relation.</td>
</tr>
<tr>
<td>GS-24204</td>
<td><strong>Empty relation.</strong></td>
</tr>
<tr>
<td></td>
<td>This relation has no field definition.</td>
</tr>
<tr>
<td></td>
<td>Add foreign/primary (or source/destination) fields.</td>
</tr>
<tr>
<td>GS-24205</td>
<td><strong>Database table column referenced more than once in the form.</strong></td>
</tr>
<tr>
<td></td>
<td>In one form, a database table column can be referred only once. Table aliases should be used to attach more than one field to a database.</td>
</tr>
<tr>
<td></td>
<td>• Select one of the wrong Formfields and change the fieldType attribute from table_column to table_alias.</td>
</tr>
<tr>
<td></td>
<td>• Select one of the wrong Formfields and change fieldType attribute to non_database.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-24206</td>
<td>Missing master table for record record.</td>
</tr>
<tr>
<td></td>
<td>Set the master table property.</td>
</tr>
<tr>
<td>GS-24207</td>
<td>Table %1 unused in query.</td>
</tr>
<tr>
<td></td>
<td>A database table is used in the record, but there is no join for it in the query.</td>
</tr>
<tr>
<td></td>
<td>Add a join for the table in the record query.</td>
</tr>
<tr>
<td>GS-24208</td>
<td>No schema attached.</td>
</tr>
<tr>
<td></td>
<td>A database is required for the business records.</td>
</tr>
<tr>
<td></td>
<td>Set the database name property to an existing database.</td>
</tr>
<tr>
<td>GS-24209</td>
<td>Invalid relation, primaryField and foreignField must have same number of fields.</td>
</tr>
<tr>
<td></td>
<td>Modify the relation fields so that a primary field corresponds to each foreign field.</td>
</tr>
<tr>
<td>GS-24210</td>
<td>Invalid query, left and right join must have same number of columns.</td>
</tr>
<tr>
<td></td>
<td>Check the query fields so that a left field corresponds to each right field.</td>
</tr>
<tr>
<td>GS-24211</td>
<td>Relation types don't match exactly.</td>
</tr>
<tr>
<td></td>
<td>The type of a foreign key doesn't match the corresponding primary key's type.</td>
</tr>
<tr>
<td></td>
<td>Check the relation, change the field types, or fix the primary / foreign key.</td>
</tr>
<tr>
<td>GS-24212</td>
<td>Relation field fieldName not found.</td>
</tr>
<tr>
<td></td>
<td>The relation refers a field that is missing in the record.</td>
</tr>
<tr>
<td></td>
<td>Modify the relation or add the field to the record.</td>
</tr>
<tr>
<td>GS-24213</td>
<td>Non existing schema schema attached to document.</td>
</tr>
<tr>
<td></td>
<td>Check the schema being referenced.</td>
</tr>
<tr>
<td>GS-24214</td>
<td>Nonexistent table table referenced in document.</td>
</tr>
<tr>
<td></td>
<td>The document uses a database table that is not present in the schema.</td>
</tr>
<tr>
<td></td>
<td>Update the schema, or change the field using the table.</td>
</tr>
<tr>
<td>GS-24215</td>
<td>Nonexistent column table.column referenced in document.</td>
</tr>
<tr>
<td></td>
<td>The document uses a database column that is not present in the schema.</td>
</tr>
<tr>
<td></td>
<td>Update the schema or change the field using the table.column.</td>
</tr>
<tr>
<td>GS-24216</td>
<td>Database table defined in &quot;no database&quot; document.</td>
</tr>
<tr>
<td></td>
<td>Set the database property to an existing schema or remove the field.</td>
</tr>
<tr>
<td>GS-24217</td>
<td>Table alias referenced more than once in the form.</td>
</tr>
<tr>
<td></td>
<td>The same table alias is associated with two columns in the form.</td>
</tr>
<tr>
<td></td>
<td>• Change one table alias name.</td>
</tr>
<tr>
<td></td>
<td>• Change either the table or column name.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-24218</td>
<td><strong>Alias alias referenced for different tables in document.</strong></td>
</tr>
<tr>
<td></td>
<td>The same alias is used for different tables.</td>
</tr>
<tr>
<td></td>
<td>Rename the alias so that it refers to the same database table.</td>
</tr>
<tr>
<td>GS-24219</td>
<td><strong>Alias Lookup field is ignored on %1 as it is the master table of the record.</strong></td>
</tr>
<tr>
<td></td>
<td>A lookup field is dedicated to foreign field update in the master table; do not set it on a master table column.</td>
</tr>
<tr>
<td></td>
<td>Remove the lookup property value or update the field database settings.</td>
</tr>
<tr>
<td>GS-24220</td>
<td><strong>Name value %1 is already used.</strong></td>
</tr>
<tr>
<td></td>
<td>Duplicate name used in the document.</td>
</tr>
<tr>
<td></td>
<td>Rename the element so that the name is unique.</td>
</tr>
<tr>
<td>GS-24221</td>
<td><strong>Invalid INTERVAL qualifier.</strong></td>
</tr>
<tr>
<td></td>
<td>The qual1 or qual2 set for the INTERVAL sqlType is not valid. It should belong to the INTERVAL classes (i.e., YEAR-MONTH or DAY-TIME)</td>
</tr>
<tr>
<td></td>
<td>Change either qual1 or qual2 to fit the respective class range or change the sqlType property from INTERVAL to another sqlType.</td>
</tr>
<tr>
<td>GS-24222</td>
<td><strong>Startfield of DATETIME or INTERVAL qualifiers must come earlier in the time-list than its endfield.</strong></td>
</tr>
<tr>
<td></td>
<td>The qual1 value should be greater than the qual2 value, when the sqlType is either DATETIME or INTERVAL.</td>
</tr>
<tr>
<td></td>
<td>Ensure qual1 is greater than qual2 or change the sqlType property from INTERVAL to another sqlType.</td>
</tr>
<tr>
<td>GS-24223</td>
<td><strong>Query properties set without attached database schema.</strong></td>
</tr>
<tr>
<td></td>
<td>Some query properties (join, order, additional tables or where) are defined without a database schema attached.</td>
</tr>
<tr>
<td></td>
<td>Clear the query properties or attach a database schema to document.</td>
</tr>
<tr>
<td>GS-24224</td>
<td><strong>Table table referenced by property is not present in the record.</strong></td>
</tr>
<tr>
<td></td>
<td>A table is referenced in a query property (join or order), but is not referenced in any record's field or additional tables.</td>
</tr>
<tr>
<td></td>
<td>• Remove the join or order that references this table.</td>
</tr>
<tr>
<td></td>
<td>• Add a field referencing a column from this table to the record.</td>
</tr>
<tr>
<td></td>
<td>• Add the table to the 'additional properties' property.</td>
</tr>
<tr>
<td></td>
<td>• In the join or order, change the table to another one that is present in the record.</td>
</tr>
<tr>
<td>GS-24225</td>
<td><strong>Invalid relation, it must have at least one field.</strong></td>
</tr>
<tr>
<td></td>
<td>• Remove the relation.</td>
</tr>
<tr>
<td></td>
<td>• Add one or more source, destination field pair.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-24226</td>
<td>Value not compatible with dataType.</td>
</tr>
<tr>
<td></td>
<td>The value is not compatible with dataType set. This error occurs when <code>defaultValue</code> and <code>include</code> property value is incompatible with dataType property.</td>
</tr>
<tr>
<td></td>
<td>Change <code>defaultValue/include</code>, enter a value compatible to dataType format.</td>
</tr>
<tr>
<td>GS-24227</td>
<td>Invalid join between %1 and %2.</td>
</tr>
<tr>
<td></td>
<td>A join between the two tables is not correct (another one exists with a different operator).</td>
</tr>
<tr>
<td></td>
<td>- Remove the join</td>
</tr>
<tr>
<td></td>
<td>- Change the operator</td>
</tr>
<tr>
<td></td>
<td>- Change the join table(s)</td>
</tr>
<tr>
<td>GS-24228</td>
<td>There is no join between table %1 and master table.</td>
</tr>
<tr>
<td></td>
<td>The specified table is joined to another one, but not to the master table, creating a Cartesian product.</td>
</tr>
<tr>
<td></td>
<td>- Add the missing join for the %1 table</td>
</tr>
<tr>
<td></td>
<td>- Remove the table %1</td>
</tr>
<tr>
<td>GS-24229</td>
<td>Invalid initializer.</td>
</tr>
<tr>
<td></td>
<td>The initializer format is incorrect.</td>
</tr>
<tr>
<td></td>
<td>Change the initializer to respect the format.</td>
</tr>
<tr>
<td>GS-24230</td>
<td>Invalid source %1 for initializer %2.</td>
</tr>
<tr>
<td></td>
<td>The initializer source (left of “:”) is unknown.</td>
</tr>
<tr>
<td></td>
<td>Change the initializer to respect the format.</td>
</tr>
<tr>
<td>GS-24231</td>
<td>Initializer property %1 is missing.</td>
</tr>
<tr>
<td></td>
<td>The property used in the initializer does not exist.</td>
</tr>
<tr>
<td></td>
<td>Change the property name in the initializer.</td>
</tr>
<tr>
<td>GS-24232</td>
<td>Cannot resolve initializer value.</td>
</tr>
<tr>
<td></td>
<td>The initializer cannot be resolved, the database element is not found, or the property is missing.</td>
</tr>
<tr>
<td></td>
<td>Change the initializer to point to a valid element.</td>
</tr>
<tr>
<td>GS-24235</td>
<td>File type not defined in Application Generator settings.</td>
</tr>
<tr>
<td></td>
<td>A file used in Application Generator to generate code (<code>generate</code>) is of an undefined file type in <code>settings.agconf</code>.</td>
</tr>
<tr>
<td></td>
<td>Add the item type definition to the <code>settings.agconf</code>.</td>
</tr>
<tr>
<td>GS-24236</td>
<td>Cannot resolve initializer without a valid database schema.</td>
</tr>
<tr>
<td></td>
<td>The property initializer uses the database schema but the file doesn’t have a schema.</td>
</tr>
<tr>
<td></td>
<td>- Set the database.</td>
</tr>
<tr>
<td></td>
<td>- Set the property value so that the initializer is not resolved.</td>
</tr>
<tr>
<td></td>
<td>- Remove the initializer.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-24237</td>
<td>Unique key field %1 is not present in the Record. The unique key field value contains one field which is not in the record. • Remove the field from the unique key. • Add the field to the record.</td>
</tr>
<tr>
<td>GS-24239</td>
<td>Inactive table %1 in the database schema. The database table active flag is set to false, the table is inactive and cannot be used for code generation. Make the table active or do not use it.</td>
</tr>
<tr>
<td>GS-24240</td>
<td>Duplicate uid ‘%1’. Invalid file format. A unique identifier has been used more than once. Contact Genero support.</td>
</tr>
</tbody>
</table>

**Business Application Diagram error messages**

A list of Business Application Diagram error messages. For messages that are not self-explanatory, additional information is provided.

**Table 104: Business Application Diagram Error Messages**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-23001</td>
<td>Updated %1 property from %2 to %3. A property value was incorrect and was updated during the load.</td>
</tr>
<tr>
<td>GS-23004</td>
<td>Cannot build '%1', file is not present in current project Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-23400</td>
<td>Invalid Relation '%1': Invalid source item/Invalid Relation '%1': Invalid destination item/Invalid tag '%1'. Cannot register node %1./&lt;xml error&gt; The .4ba file is invalid and contains incorrect node definitions. Contact support.</td>
</tr>
<tr>
<td>GS-23402</td>
<td>Item is conflicted. There are conflicts between items (they share same ID or file path). Resolve the conflict. (Right-click and select Resolve Conflict menu option.)</td>
</tr>
<tr>
<td>GS-23403</td>
<td>Item not implemented. Implement the item. (Right-click on item and select Implement.)</td>
</tr>
<tr>
<td>GS-23404</td>
<td>Invalid file: %1 Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-23405</td>
<td>File %1 is missing. Some implemented item file is missing on the disk. Restore the file or recreate the item and implement it.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-23406</td>
<td>Item/Relation type %1 not supported. Relation type %1 not supported</td>
</tr>
<tr>
<td></td>
<td>Open the template file settings.agconf and fix the errors.</td>
</tr>
<tr>
<td>GS-23407</td>
<td>Unsupported property %1 found for %2.</td>
</tr>
<tr>
<td></td>
<td>Open the template file settings.agconf and add the property definition.</td>
</tr>
<tr>
<td>GS-23408</td>
<td>Duplicate Item name %1 found.</td>
</tr>
<tr>
<td></td>
<td>Multiple items share the same name property and it should be unique. Modify name property value so that it is unique within the file items.</td>
</tr>
<tr>
<td>GS-23409</td>
<td>Duplicate Relation name %1 found.</td>
</tr>
<tr>
<td></td>
<td>Multiple relations share the same name property, name should be unique. Modify name property value so that it is unique within the file relations.</td>
</tr>
<tr>
<td>GS-23410</td>
<td>&lt;Constraint&gt; (&lt;description&gt;) constraint failed.</td>
</tr>
<tr>
<td></td>
<td>Invalid source &lt;Item&gt;. When a constraint is defined with minSource and maxSource values as 0, it implies that the source item should not have any outgoing relations of type given in reference. This error occurs as the item has non-supported outgoing relations. Remove all the unsupported outgoing relations from the item.</td>
</tr>
<tr>
<td>GS-23411</td>
<td>&lt;Constraint&gt; (&lt;description&gt;) constraint failed.</td>
</tr>
<tr>
<td></td>
<td>Invalid destination &lt;Item&gt;. When a constraint is defined with minDestination and maxDestination values as 0, it implies that the destination item should not have any incoming relations of type given in reference. This error occur as the item has non-supported incoming relations. Remove all the unsupported incoming relations from the item.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-23412</td>
<td><code>&lt;Constraint&gt; (&lt;description&gt;)</code> constraint failed. Number of outgoing Relations, <code>&lt;Count&gt;</code>, are less than <code>&lt;Expected Count&gt;</code>. The numbers of outgoing relations from the item are less than the expected minimum outgoing relations count. Add at least <code>&lt;Expected Relation Count&gt;</code> number of outgoing relations to fix this problem. Number of outgoing Relations, <code>&lt;Count&gt;</code>, are greater than <code>&lt;Expected Count&gt;</code>. The numbers of outgoing relations from the item are greater than the expected maximum outgoing relation count. User has to remove the extra added relations, so that the relation count will not across <code>&lt;Expected Relation Count&gt;</code> which is the maximum number of outgoing relations allowed. Number of incoming Relations, <code>&lt;Count&gt;</code>, are less than <code>&lt;Expected Count&gt;</code>. The numbers of incoming relations to the item are less than the expected minimum incoming relations count. User has to add at least <code>&lt;Expected Relation Count&gt;</code> number of incoming relations to fix this problem. Number of incoming Relations, <code>&lt;Count&gt;</code>, are greater than <code>&lt;Expected Count&gt;</code>. The numbers of incoming relations to the item are greater than the expected maximum incoming relation count. User has to remove the extra added relations, so that the relation count will not across <code>&lt;Expected Relation Count&gt;</code> which is the maximum number of incoming relations allowed.</td>
</tr>
<tr>
<td>GS-23413</td>
<td>Item ID can not be blank. An ID is blank, the file is invalid. Contact your support center or remove the item containing the ID an create a new one.</td>
</tr>
<tr>
<td>GS-23414</td>
<td>Duplicate ID found for <code>%1</code> and <code>%2</code>. Multiple items share the same id, creating a conflict. (Right-click and select Resolve Conflict menu option.)</td>
</tr>
<tr>
<td>GS-23415</td>
<td><code>%1</code> file not supported by Business Application Diagram <code>%2</code>. A build rule execution is trying to access data not described in the template settings.agconf (through the Business Application diagram). Update the template settings.agconf file, adding the unsupported item or change the file build rule to not use the file.</td>
</tr>
<tr>
<td>GS-23416</td>
<td>Orphan property <code>%1</code>, clean document settings to remove it. Some defined property does not belong to the current settings (orphan) and are ignored by the compilation process. Check the metadata and either update the current template settings.agconf or clean orphan properties.</td>
</tr>
<tr>
<td>GS-23417</td>
<td>Orphan property group <code>%1</code>, clean document settings to remove it. Some defined property group does not belong to the current settings (orphan). Check the metadata and either update the current template settings.agconf or clean orphan properties.</td>
</tr>
<tr>
<td>GS-23418</td>
<td>Node <code>%1</code> contains orphan properties, clean document settings to remove them. Some node contains metadata definitions which do not belong to the current settings (orphan). They are ignored by the compilation process. Check the metadata and either update the current template settings.agconf or clean orphan properties.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-23419</td>
<td>Obsolete item type %1. The Item definition does not exist in the current template settings. It is ignore by the compilation process. Add the Item definition to the template settings.agconf, convert the item to a corresponding supported Item type (Right-click and select Convert menu option.), or remove the item.</td>
</tr>
<tr>
<td>GS-23420</td>
<td>Obsolete relation type %1. The relation definition does not exist in the current template settings and is ignored by the compilation process. To resolve this error: 1. Add the Relation definition to the template settings.agconf. 2. Convert the item to a corresponding supported Item type (context menu convert action). 3. Remove the item.</td>
</tr>
</tbody>
</table>

**Meta-schema Manager**

The *Meta-schema Manager* is a visual tool used to design, create and maintain database meta-schema files.

- What is a database meta-schema? (4db) on page 343
- Creating a meta-schema on page 344
- Adding to a meta-schema on page 349
- Viewing a meta-schema on page 356
- Comparing two meta-schemas on page 358
- Update a meta-schema from database on page 359
- Generate a database script from meta-schema on page 359
- Generate meta-schema documentation on page 361
- Copy a table or column definition as text on page 361
- Meta-schema Manager Reference on page 361

**What is a database meta-schema? (4db)**

A database *meta-schema file* is the central repository of a database's meta-data, containing information about the tables, columns, and relations, and default values of a relational database. Information from the database meta-schema file is used by Genero Studio Form Designer, Business Application Modeling, and Code Editor.

**4db and 4dbx files**

There are two types of database meta-schema files used by Genero Studio.

- **4db**
  - Used in standard Genero Studio projects.

- **4dbx**
  - Used in code-generated projects in Business Application Modeling. It includes special features for application generation such as a table for managing serial columns.

**When is it used?**

A database meta-schema file is required when you use Genero Studio to:

- Examine the structure of a tables, columns, and other attributes.
• Create a Genero Studio form definition based on a database table.
• Generate application code.

The database meta-schema is not used by Genero Studio when an application is executed. Genero Studio uses the Genero runtime system and the specific database client software to access any databases referenced in a Genero application.

Meta-schema creation and maintenance

Database meta data can be extracted from a relational database into a meta-schema file, or it can be based on an existing database schema file.

It is important that the database meta-schema file match the current structure of the database itself. If changes are made in the database structure, any database meta-schema files that describe the database must also be updated.

Meta-schema Manager includes options to create and update a database based on meta-schema file changes.

**GSTSCHEMANAMES**

Set the **GSTSCHEMANAMES** on page 167 environment variable to make meta-schemas available to all projects.

**Related concepts**

- Creating a meta-schema on page 344
  Information on creating meta-schemas.

**Related tasks**

- Update a meta-schema from database on page 359
  Update a schema file to the current structure of its associated database.
- Generate a database script from meta-schema on page 359
  Generate a 4gl source file to be used to create or update a database that is described in the meta-schema file.

---

**Creating a meta-schema**

Information on creating meta-schemas.

• Create a meta-schema on page 277
• Extract meta-schema information from database on page 277
• BDL schema file (sch) on page 349
• Add a meta-schema to a project on page 281

**Create a meta-schema**

Create a database meta-schema file.

1. Select **File > New**.
   The **New** dialog opens.
2. Select a category from the left-side panel.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genero BAM Desktop</td>
<td>Select this option if you are creating a database meta-schema for use in a Business Application Modeling project for a desktop application. The file created will be a 4dbx.</td>
</tr>
<tr>
<td>Genero BAM Mobile</td>
<td>Select this option if you are creating a database meta-schema for use in a Business Application Modeling project for a mobile application. The file created will be a 4dbx.</td>
</tr>
</tbody>
</table>
3. Select an option from the Database section from the right-side panel.

<table>
<thead>
<tr>
<th>Database option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Schema</td>
<td>Creates a new meta-schema file.</td>
</tr>
<tr>
<td>DB Schema from Database</td>
<td>Opens the New Meta-schema dialog to which you enter your database and connection information. See Extract meta-schema information from database on page 277.</td>
</tr>
</tbody>
</table>

The meta-schema file displays in the document view. A new meta-schema file is automatically added to the list in the DB Schemas Tab if you have saved the file in the current project.

**Extract meta-schema information from database**

The **New Meta-schema** dialog assists in extracting schema information from a database.

To extract a database meta-schema file from a database, you must have access and permissions for the database. If you have trouble connecting to a database, make sure the database and the corresponding database client software are installed and configured properly.

When you extract the meta-schema information from the database, you overwrite the existing schema. Any user changes that had been made to the schema are lost when using the extract schema option. If you wish to keep user changes, you must update the schema. See Update a meta-schema from database on page 359.

1. Select Database >> Extract Schema. The first step is specifying the name and location of the meta-schema file.
**Meta-schema file path**

Enter the name and path for the new database meta-schema file. Specify a `.4db` file extension for standard projects, or a `.4dbx` file extension if you are working with a Business Application Modeling managed project.

You can either click on the browse button to browse for a location on disk, or you can enter the full path and file name directly in the **Meta-schema file path** field. If you browse for the location, you select the file type in the **Save as type** combobox when saving the file. If you enter the file name directly, you must ensure that you specify the correct file extension (`.4db` or `.4dbx`).

**Insert the file in the project**

Check this box to add the meta-schema file in the project. Select the node where the file should be added.

2. Click the **Next** button to continue to **Connection information**. This connection information is only used to extract the information for the database meta-schema file from the referenced database.

![Connection information](image)

**Figure 183: Connection information**

a) In the **Database Connection Information** section, select either **Use explicit settings** or **Use external settings**.
Use explicit settings, previous connection

You can use a previous connection that was created for the same database. The drop down list provides a list of the existing connections.

Use explicit settings, database type

You can enter the Database Type by selecting the desired type from the drop down list, and the corresponding information for that type. The Database driver for the database type is automatically entered. If other drivers exist, they are available in the drop down list.

Use external settings

Information in the FGLPROFILE configuration file is used to extract the corresponding connection information for the specified database. Genero Studio will use the schema name that you entered to check for any related entry in the FGLPROFILE configuration file, and will use those values to define the connection. See information on the FGLPROFILE file in the Genero Business Development Language User Guide.

b) In the Database User Information section, provide the necessary database user details. The required information varies based on the database type selected. See Database server/user information on page 365.

c) Click Test Connection to verify that the information is correct and that you are able to access the database.

3. Click the Next button to continue to Extraction Options. Select the options for the meta-schema file.
Figure 184: Extraction options

Case sensitivity
Specify how case in database object names should be handled. **Case sensitive**: case won't be changed on database objects. **Lower case**: database object names will be converted to lower case, **Upper case**: database object names will be converted to upper case.

Import system tables
Check this box to include system tables in the schema.

Ignore errors
Specify that conversion errors should be ignored. If this option is unchecked, the extraction will stop as soon as an error occurs (for example, if a table column has an unsupported type.)

Conversion method
Select the type of conversion you wish for the specific data types; the default choice is Type A.

4. **Click Finish** to begin the extraction process.
If you didn't already do so, save the database meta-schema file in a node in the project; the database meta-schema will be added to the DB Schemas tab and made available to other modules.

Any application that uses the meta-schema file must have a dependency to the node where the meta-schema file was added. See **Add a meta-schema to a project** on page 281.
**BDL schema file (sch)**

Genero BDL *sch* files contain definitions of the database tables and columns.

This *sch* file is automatically created when you compile a Genero Studio meta-schema file, or you can import an existing one using the **Database > Import SCH file** menu option.

Importing an existing Genero BDL column definition file (*sch*), converts the format to a Genero Studio meta-schema file (*4db*). You are prompted for the meta-schema name and path.

**When the sch file is used**

The BDL *sch* file is used when a Genero BDL module (*4gl* file) is compiled. If you compile a BDL module/program from the command line, you must have a copy of the *sch* file in the same directory or set **FGLDBPATH** environment variable specifying the directory in which the *sch* files can be found.

**Add a meta-schema to a project**

You may have to add the meta-schema file to a project.

If you used **File > New** to create your project, the default structure of your project includes nodes for a **project**, **application**, **library**, and **databases**; the dependencies between the default nodes has been predefined. When you save your Meta-schema file in the Databases node of the project, the dependency for the application node in the project already exists.

However, if you created your own project structure, you must follow these steps.

1. Open the project.
2. Right-click on the application or library node in the Project view to which you want to add the meta-schema file and select **Add Files**. Locate and add the meta-schema file.
3. Add a dependency to the Meta-schema file for any application or library nodes. Right-click the node and select **Advanced Properties, Dependencies**. Check the box for the node containing the Meta-schema file.

**Adding to a meta-schema**

Add more items and information to meta-schemas, such as tables and columns, constraints, indexes, and foreign keys.

**Warning:** The **Build** command does not automatically rebuild the entire project for each modification of the meta-schema file (*4db*, *4dbx*). It is the responsibility of the developer to recompile the appropriate parts of the project or to use the **Rebuild all** command.

- **Add new tables and columns** on page 349
- **Add constraints or indexes** on page 350
- **Add foreign keys** on page 351
- **Add a many-to-many relationship** on page 354
- **Centralize field information (label, widget, default value)** on page 355

**Add new tables and columns**

You can add tables and columns to a meta-schema.

Right-click in the background of the meta-schema diagram. Select **Add Table**. A new table with a single column is added to the diagram.

Right-click on the table and select **Add Column** to add an additional column. Repeat for as many columns as needed.

Set properties for the table and each column by selecting the item in the diagram or Structure view and editing its properties in the Properties view.
Add constraints or indexes

The Business Application Modeler requires that tables are defined with primary keys. Constraints and indexes that are part of the database structure are displayed as part of the table, but additional constraints (such as primary keys) can be added.

Right-click the table in the meta-schema diagram and select Add Constraint or Index.

Figure 185: Add Constraint or Index dialog

Specify the index type.

- **Primary Key**: A Primary Key is a column or set of columns that uniquely identifies a row of data. The Not Null property must be set on the columns used in a Primary Key. The Not Null property indicates the column does not allow NULL values. It is used by the generation and update scripts when generating the SQL statements used to manage the database.

- **Secondary Key**: Also known as a Unique Constraint. Like Primary Key, the Secondary Key ensure uniqueness on the columns it is defined, but also allows NULL values.

- **Index**: An index improves the speed of looking up data in tables. Indexes can be defined on one or more table column.

- **Unique Index**: Unique Index behaves the same as an Unique Constraint. It ensures the data in the column is unique. The difference between a Unique Constraint and a Unique Index depends on the database engine.

The index can be viewed in the Structure view.
Select the index to display its properties in the Properties view.

If an index contains more than one column, the order of the columns is indicated.

**Add foreign keys**

A foreign key constraint specifies that values in one table must also appear in another table. Foreign keys that are part of the database structure are displayed, but foreign keys can also be added.

Right-click on the table in the meta-schema diagram to which you want to add a foreign key. Select **Add Foreign Key**.
Figure 188: Add Foreign Key dialog

| Meta-schema Manager | 352 |

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Name of foreign key, a suggested name is provided. This name will display in the Structure and Properties views.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Select the table that contains the foreign key.</td>
</tr>
<tr>
<td><strong>Referenced table</strong></td>
<td>Select the table that contains the primary key being referenced.</td>
</tr>
<tr>
<td><strong>Column</strong></td>
<td>Select the table column that references the primary key column in the referenced table.</td>
</tr>
<tr>
<td><strong>Referenced Column</strong></td>
<td>Select the name of the primary key column in the referenced table.</td>
</tr>
<tr>
<td><strong>Cascade delete</strong></td>
<td>Check this box if this is a foreign key with cascade delete. A foreign key with a cascade delete specifies that if a row in the parent table is deleted, then the corresponding rows in the child tables are automatically deleted. When this box is unchecked, the deletion of a row in a parent table will be aborted if a corresponding row exists in child tables. See Cascade delete on page 284.</td>
</tr>
</tbody>
</table>
Figure 189: Foreign Key relationship in diagram

The foreign key constraint is added to the table in the Structure view. Select the foreign key constraint to display or edit its properties.
Add a foreign key by drawing relationship

You can also define the foreign key by drawing the relationship between columns using the mouse. Right-click on the background of the meta-schema diagram and select Add Foreign Key. Select the foreign key column in one table, and drag the mouse towards the primary key column in the table to be referenced. Once you release the mouse button, the relationship is displayed and the foreign key is added. Confirm the foreign key properties in the Properties view.

Add a many-to-many relationship

To design a many-to-many relationship between two tables, create a junction table to link them together.

Create the junction table using the primary key from each table.

Example

Table_A has a primary key column named Column_A.

Table_B has a primary key column named Column_B.
To create the many-to-many relationship, add a junction table. This table is comprised of the primary key columns from each of the other tables.

Centralize field information (label, widget, default value)

Specifying field properties at the meta-schema level allows you to centralize properties such as the label, widget, and default value.

1. Open the meta-schema file (4db or 4dbx).
2. Select a column in the table and set its **Default value**, **widget**, and/or **label** properties.
   
   For example, if you set the state field's **label** property to *State* and its **widget** property to *comboBox*, when you build a form that includes this field, it will be built with a label of State and as a comboBox instead of the default Edit widget type.
3. Repeat for all fields to which you want to centralize information.
4. Save the meta-schema file.

**Related concepts**

- **Meta-schema properties** on page 362
- Properties can be set for each element in a meta-schema.

**Viewing a meta-schema**

Once a database meta-schema file has been created, it can be viewed and enriched with information that is not present in the database. Opening a meta-schema file displays it as a diagram for viewing and editing.

Each table is described by a table in the diagram. The structure of the database tables is displayed in the Database Structure view.
Icons on the diagram and in the Database Structure view indicate when a column's status has been changed: added (⁺), modified (%), or removed (−).

A red circle icon with a white exclamation point, found in the upper left corner of the visual representation of the table, identifies a table that has been modified. To view the specifics of the modification, hover over the icon with your mouse.

Figure 195: Viewing the modifications for a table

**Zoom in and out**

Use Ctrl+mouse wheel to zoom in and out on the diagram.
Reverting a change
Revert any changes made to the schema with Database > Revert.... See Revert schema changes dialog on page 373

Multiple views
Right-click the View tab at the bottom of the document to duplicate, rename, or delete a view. You may display multiple views of the same meta-schema.

Filter shown items
Filter the items shown on the diagram with the right-click context menu Filter Items....

Re-order columns
You can re-order columns using drag-and-drop.

Move a column to another table
You can use drag-and-drop to move a column from one table to another.

Show or hide foreign key names
To show foreign key names, right-click on the diagram and select Display Foreign Keys Label.
To hide foreign key names, right-click on the diagram and clear Display Foreign Keys Label.

View the details behind the icons
You can mouse over any object on the diagram that has a visual icon (such as a primary key icon or a status icon) to view the details of that icon. For example, if a table has a status icon indicating a change, mouse over the change icon to view the details of the changes.

Comparing two meta-schemas
Compare two database meta-schema files.
Select Database > Diff Schema... to create a merged schema view where:
- An object found in the first schema and not in the second will have a status of removed
  ❌
  . An object with a status of removed is not included in the schema diagram, but can be seen in the Database Structure view.
- An object found in the second schema and not in the first will have a status of added
  ✔
  .
- An object with the same name in both schemas but whose properties have changed in the second will have a status of modified
  ❗
  .

Figure 196: Selecting two meta-schema files to compare with Schema Diff
Update a meta-schema from database

Update a schema file to the current structure of its associated database.

This procedure merges any changes made in the structure of a database into your meta-schema file. This requires that you have access to the database. Any information that you have added to the meta-schema will be preserved.

When the changes are merged, the Meta-schema Manager verifies that the database objects in the original meta-schema still exist. If the object is no longer present in the database, it is removed from the meta-schema.

1. Right-click on the meta-schema file in the Projects or DB Schemas tab.
2. Select Update Schema ... or Update from Database.
3. Complete the Update meta-schema dialog as described in Extract meta-schema information from database on page 277.

The database meta-schema file is updated to match the current structure of the database.

Warning: The Build command does not automatically rebuild the entire project for each modification of the meta-schema file (.4db, .4dbx). It is the responsibility of the developer to recompile the appropriate parts of the project or to use the Rebuild all command.

Generate a database script from meta-schema

Generate a 4gl source file to be used to create or update a database that is described in the meta-schema file.

1. Right-click on the meta-schema file in the project.
2. Select an option:
   - Generate Database Creation Script to generate a source 4gl file that can be used to create a new database and tables according to the meta-schema file.
   - Generate Database Update Script to generate a source 4gl file that can be used to update an existing database based on the meta-schema file.

   Note: The database update script will first drop the existing tables including their data, and then recreate the structure of the database based on the modified schema. Previous versions of the tables will be backed up and the data will be migrated to the new tables when applicable. It is recommended to perform a backup of the database prior to running the update script.
3. Complete the Generate Database Script dialog.
Figure 197: Generate Database Script dialog

- **Database schema**: Name of selected meta-schema file.
- **Target database**: Specify the database to use in the script.
- **Generate primary keys, secondary keys, indexes**: Specify whether to include primary keys, secondary keys, and indexes in the script.
- **Use national char types**: By default, the database creation / update scripts will generate column using standard char types. If this option is set, the scripts will produce columns using national char types. For example, with an Oracle database, the column types will be CHAR or VARCHAR2 when the option is not selected, and NCHAR, NVARCHAR2 when the option is selected.
- **Populate database with sample data**: Add statements in the script to add sample data to the database.

4. Select **Generate**.

**Related concepts**
- Database Generation Script dialog on page 372
Generate a 4GL source file to create or update the database that is described in the meta-schema file.

**Generate meta-schema documentation**

You can generate an HTML file that provides documentation on the meta-schema. It lists the tables, columns, indexes and foreign keys.

Any modifications made to the meta-schema will be considered as if applied to the database. New objects added to the meta-schema and modification made on existing objects will be seen in the documentation. Removed objects will not be shown.

1. Open a meta-schema file.
2. Select **Database > Generate Schema Documentation**.
   The documentation is generated in a temporary file and displayed in a browser.
3. To keep a copy of the documentation, save the page from the browser.

**Copy a table or column definition as text**

From a meta-schema diagram, you can copy a table or a column and paste the results in a Code Editor text document.

**About this procedure**

Using the copy-and-paste method, you can quickly add DEFINE statements to your source code.

When you copy a table, the DEFINE statement creates a record using the table name as part of the record variable name:

```fortran
DEFINE t_orders RECORD LIKE orders.*
```

When you copy a column, the DEFINE statement creates a variable using the column name as part of the variable name:

```fortran
DEFINE c_orderid LIKE orders.orderid
```

**Before you begin:**

You should have two documents open in the central work area: a text file displayed in Code Editor and a database meta-schema file displaying a diagram of tables and columns.

1. Select the tab displaying the database meta-schema file.
2. Right-click on a table or column, and select **Copy** from the context menu.
3. Select the tab holding your text document in Code Editor.
4. Place the cursor where you wish to insert the DEFINE statement, right-click and select **Paste**.
   The DEFINE statement is written to the text file.

**Meta-schema Manager Reference**

Reference information for Meta-schema Manager.

- [Meta-schema properties](#) on page 362
- [Data types](#) on page 363
- [Database server/user information](#) on page 365
- [Dialogs](#) on page 367
- [Views](#) on page 373
- [Meta-Schema Manager preferences](#) on page 375
- [Meta-schema diagram context menu](#) on page 376
Meta-schema properties

Properties can be set for each element in a meta-schema.

Table 105: Table properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the table.</td>
</tr>
<tr>
<td>Description</td>
<td>A description of the table. This value is used as a tooltip when moving the</td>
</tr>
<tr>
<td></td>
<td>mouse cursor over the table name in the database diagram. The values set in</td>
</tr>
<tr>
<td></td>
<td>the schema document will be overwritten by the values found in the database</td>
</tr>
<tr>
<td></td>
<td>when updating the database schema from database.</td>
</tr>
<tr>
<td></td>
<td><strong>Important:</strong> The description property is extracted only from Oracle®</td>
</tr>
<tr>
<td></td>
<td>databases for now.</td>
</tr>
<tr>
<td>View</td>
<td>This checkbox indicates that the object was created from a view, not a</td>
</tr>
<tr>
<td></td>
<td>table. This property is a read-only property set when extracting the database</td>
</tr>
<tr>
<td></td>
<td>schema. Views cannot be edited. They can only be deleted.</td>
</tr>
<tr>
<td>Active</td>
<td>The active property indicates that the table participates in the application</td>
</tr>
<tr>
<td></td>
<td>code generation.</td>
</tr>
<tr>
<td>Category</td>
<td>The category property allows you to group tables using categories that you</td>
</tr>
<tr>
<td></td>
<td>define yourself. There is no master list of categories; you simply type in</td>
</tr>
<tr>
<td></td>
<td>the category name that you wish to assign a table.</td>
</tr>
<tr>
<td></td>
<td>Categories can be used to filter items in the database meta-schema diagram.</td>
</tr>
<tr>
<td></td>
<td>See Filter View dialog on page 121.</td>
</tr>
<tr>
<td></td>
<td>This property is optional.</td>
</tr>
</tbody>
</table>

Table 106: Column properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the column.</td>
</tr>
<tr>
<td>Description</td>
<td>A description of the column. This value is used as a tooltip when moving the</td>
</tr>
<tr>
<td></td>
<td>mouse cursor over the column name in the database diagram. The values set in</td>
</tr>
<tr>
<td></td>
<td>the schema document will be overwritten by the values found in the database</td>
</tr>
<tr>
<td></td>
<td>when updating the database schema from database.</td>
</tr>
<tr>
<td></td>
<td><strong>Important:</strong> The description property is extracted only from Oracle®</td>
</tr>
<tr>
<td></td>
<td>databases for now.</td>
</tr>
<tr>
<td>Order</td>
<td>Position of the column in the table.</td>
</tr>
<tr>
<td>Data Type</td>
<td>Specifies the data type of the column. See Data types on page 363.</td>
</tr>
<tr>
<td>Length</td>
<td>Defines the maximum length of the character string. The upper limit is</td>
</tr>
<tr>
<td></td>
<td>65534.</td>
</tr>
<tr>
<td>Precision</td>
<td>For DECIMAL data types; defines the number of significant digits (limit is</td>
</tr>
<tr>
<td></td>
<td>32, default is 16).</td>
</tr>
<tr>
<td>Scale</td>
<td>For DECIMAL data types; defines the number of digits to the right of the</td>
</tr>
<tr>
<td></td>
<td>decimal point.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Qualifier 1, Qualifier 2</td>
<td>Specify the qualifiers for INTERVAL and DATETIME data typed columns, for example YEAR and MONTH.</td>
</tr>
<tr>
<td>Not null</td>
<td>Specifies that the column does not accept NULL values.</td>
</tr>
<tr>
<td>Default value</td>
<td>Assigns a default value to a column.</td>
</tr>
<tr>
<td>Label</td>
<td>Specifies the default label for a form item using the column.</td>
</tr>
<tr>
<td>Widget</td>
<td>Widgets are designed for data handling, action triggering, or decoration. Specify the default widget for a form item using the column.</td>
</tr>
</tbody>
</table>

Table 107: Index properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the index.</td>
</tr>
<tr>
<td>Columns</td>
<td>The table columns that define the index.</td>
</tr>
</tbody>
</table>
| Type     | The type of index:  
- Primary Key  
- Secondary Key  
- Index  
- Unique Index |

Table 108: Foreign key properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the foreign key.</td>
</tr>
<tr>
<td>Table</td>
<td>Table that contains the foreign key.</td>
</tr>
<tr>
<td>Columns</td>
<td>The table column that references the primary key column in the referenced table.</td>
</tr>
<tr>
<td>Referenced table</td>
<td>The table that contains the primary key being referenced.</td>
</tr>
<tr>
<td>Referenced columns</td>
<td>The name of the primary key column in the referenced table.</td>
</tr>
<tr>
<td>Cascade delete</td>
<td>A foreign key with a cascade delete specifies that if a row in the parent table is deleted, then the corresponding rows in the child tables are automatically deleted.</td>
</tr>
</tbody>
</table>

Data types

Data types in a meta-schema have the same meaning for every supported database type.

If needed, the data type will be converted to the appropriate target database type when generating a database creation / update script. See the SQL adaptation guides in the Genero Business Development Language User Guide for more information on the use of data types in your database.

Table 109: Common data types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>The BIGINT data type is used for storing very large whole numbers.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BIGSERIAL</td>
<td>The BIG SERIAL data type produces automatic integer sequences. BIGSERIAL is based on 64 bit integer sequences.</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>The BOOLEAN data type stores a logical value, TRUE or FALSE.</td>
</tr>
<tr>
<td>BYTE</td>
<td>The BYTE data type stores any type of binary data, such as images or sounds.</td>
</tr>
<tr>
<td>CHAR</td>
<td>The CHAR data type is a fixed-length character string data type.</td>
</tr>
<tr>
<td>DATE</td>
<td>The DATE data type stores calendar dates with a Year/Month/Day representation.</td>
</tr>
<tr>
<td>DATETIME</td>
<td>The DATETIME data type stores date and time data with time units from the year to fractions of a second.</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>The DECIMAL data type is provided to handle large numeric values with exact decimal storage.</td>
</tr>
<tr>
<td>FLOAT</td>
<td>The FLOAT data type stores values as double-precision floating-point binary numbers with up to 16 significant digits.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>The INTEGER data type is used for storing large whole numbers.</td>
</tr>
<tr>
<td>SERIAL</td>
<td>The SERIAL data type produces automatic integer sequences. SERIAL is based on 32 bit integer sequences.</td>
</tr>
<tr>
<td>SMALLFLOAT</td>
<td>The SMALLFLOAT data type stores values as single-precision floating-point binary numbers with up to 8 significant digits.</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>The SMALLINT data type is used for storing small whole numbers.</td>
</tr>
<tr>
<td>TEXT</td>
<td>The TEXT data type stores large text data.</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>The VARCHAR data type is a variable-length character string data type, with a maximum size. It is converted to the appropriate target database type when generating database creation / update script.</td>
</tr>
</tbody>
</table>

**Table 110: Informix® specific data types**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERVAL</td>
<td>The INTERVAL data type stores spans of time as Year/Month or Day/Hour/Minute/Second/Fraction units.</td>
</tr>
<tr>
<td>MONEY</td>
<td>The MONEY data type is provided to store currency amounts with exact decimal storage.</td>
</tr>
<tr>
<td>NCHAR / NVARCHAR</td>
<td>IBM® Informix® supports the standard NCHAR and NVARCHAR data types. These types are equivalent to CHAR and VARCHAR (the same character set is used), except that the collation order is locale specific with NCHAR/ NVARCHAR types.</td>
</tr>
</tbody>
</table>
Informix® specific types should be avoided when designing databases. A warning is displayed when a column uses an Informix® specific type: INT8, INTERVAL, MONEY, NCHAR, NVARCHAR, NVARCHAR2, SERIAL8. To correct this warning convert columns to a common data type:

- INT8 to BIGINT
- MONEY to DECIMAL
- SERIAL8 to BIGSERIAL
- NCHAR / NVARCHAR / NVARCHAR2 to CHAR / VARCHAR and check the Use national char types option when generating database scripts
- avoid using INTERVAL

**Note:** The internal data type is used when generating database creation / update scripts or when working with the Business Application Modeler. However, if you edit a data type in a schema from the extracted database, the generic data type used in the creation of the schema might change the internal data type. For example, a column extracted from an Informix® database of type INT8 will be displayed in the schema as BIGINT. If the you change the type definition in the schema back to BIGINT, the database creation / update scripts will this time create a column of type BIGINT in the Informix® database.

### Database server/user information

Database server and user information for each supported database type.

- IBM DB2 on page 365
- Informix on page 365
- MySQL on page 366
- Oracle on page 366
- PostgreSQL on page 366
- SQLServer on page 366
- SQLite on page 366

**Related tasks**

[Extract meta-schema information from database on page 277](#)

The New Meta-schema dialog assists in extracting schema information from a database.

#### IBM® DB2®

**Table 111: IBM® DB2®**

<table>
<thead>
<tr>
<th>ODBC datasource</th>
<th>Name of the DataSource that has been previously created in ODBC.</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name</td>
<td>User name used for connection, or blank if not required.</td>
</tr>
<tr>
<td>User password</td>
<td>User password for the connection, or blank if not required.</td>
</tr>
</tbody>
</table>

#### Informix®

**Table 112: Informix®**

<table>
<thead>
<tr>
<th>Informix® server</th>
<th>Name of the Informix® database server instance (the Informix® environment that contains the database to which you wish to connect). This is the same as the value of the INFORMIXSERVER variable.</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name</td>
<td>User name used for connection, or blank if not required.</td>
</tr>
<tr>
<td>User password</td>
<td>User password for the connection, or blank if not required.</td>
</tr>
</tbody>
</table>

**Note:** For Informix® SE, the system variable DBPATH must be set.
**MySQL**

**Table 113: MySQL**

<table>
<thead>
<tr>
<th>Host name or IP address</th>
<th>Where the database is located.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database name</td>
<td>MySQL database name.</td>
</tr>
<tr>
<td>User name</td>
<td>User name used for connection, or blank if not required.</td>
</tr>
<tr>
<td>User password</td>
<td>User password for the connection, or blank if not required.</td>
</tr>
</tbody>
</table>

**Note:** If you are using a remote MySQL database and a dynamic runner, these environment variables must be set in Configurations:

- `MYSQLDIR=mysql installation directory`
- `MYSQL_UNIX_PORT=$MYSQLDIR/mysql.sock`
- `MYSQLPORT=port used for mysql socket`

**Oracle**

**Table 114: Oracle**

<table>
<thead>
<tr>
<th>TNS name</th>
<th>Oracle® TNS service name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema name</td>
<td>Name of Oracle® schema.</td>
</tr>
<tr>
<td>User name</td>
<td>User name used for connection, or blank if not required.</td>
</tr>
<tr>
<td>User password</td>
<td>User password for the connection, or blank if not required.</td>
</tr>
</tbody>
</table>

**PostgreSQL**

**Table 115: PostgreSQL**

<table>
<thead>
<tr>
<th>Host name or IP address</th>
<th>Where the database is located.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Port number used to access the database host</td>
</tr>
<tr>
<td>Database name</td>
<td>PostgreSQL database name.</td>
</tr>
<tr>
<td>User name</td>
<td>User name used for connection, or blank if not required.</td>
</tr>
<tr>
<td>User password</td>
<td>User password for the connection, or blank if not required.</td>
</tr>
</tbody>
</table>

**SQLServer**

**Table 116: SQLServer**

<table>
<thead>
<tr>
<th>ODBC datasource</th>
<th>Name of the DataSource that has been previously created in ODBC.</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name</td>
<td>User name used for connection, or blank if not required.</td>
</tr>
<tr>
<td>User password</td>
<td>User password for the connection, or blank if not required.</td>
</tr>
</tbody>
</table>

**SQLite**

**Table 117: SQLite**

<table>
<thead>
<tr>
<th>Database file</th>
<th>Name of the file that contains the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name</td>
<td>User name used for connection, or blank if not required.</td>
</tr>
</tbody>
</table>
User password

User password for the connection, or blank if not required.

**Dialogs**

Information about Meta-schema Manager dialogs.

- **Advanced Properties dialog** on page 367
- **Connection information dialog** on page 370
- **Database Generation Script dialog** on page 372
- **Revert schema changes dialog** on page 373

**Advanced Properties dialog**
The Advanced properties dialog provides options for extracting and generating database schema information.

The **Advanced Properties** dialog appears when you open a meta-schema (.4db, .4dbx) file and either:

- Right click on an empty space in the diagram and choose **Advanced properties** from the contextual menu
- Choose **Database > Advanced properties**.

In addition:

- The form shown on the **Extraction** tab displays when you complete the process to extract a meta-schema from a database.
- The form shown on the **Generation** tab displays when you complete the process to generate a database script from a meta-schema file.
Extraction

![Advanced Properties dialog, Extraction tab](image)

Figure 198: Advanced Properties dialog, Extraction tab

**Case sensitivity**

Specify how case in database object names should be handled. **Case sensitive**: case won’t be changed on database objects, **Lower case**: database object names will be converted to lower case, **Upper case**: database object names will be converted to upper case.

**Import system tables**

Check this box to include system tables in the schema.

**Ignore errors**

Specify that conversion errors should be ignored. If this option is unchecked, the extraction will stop as soon as an error occurs (for example, if a table column has an unsupported type.)

**Conversion method**

Select the type of conversion you wish for the specific data types; the default choice is Type A.
Generation

Figure 199: Advanced Properties dialog, Generation tab

<table>
<thead>
<tr>
<th>Database schema</th>
<th>Name of selected meta-schema file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target database</td>
<td>Specify the database to use in the script.</td>
</tr>
<tr>
<td>Generate primary keys, secondary keys, indexes</td>
<td>Specify whether to include primary keys, secondary keys, and indexes in the script.</td>
</tr>
<tr>
<td>Use national char types</td>
<td>By default, the database creation / update scripts will generate column using standard char types. If this option is set, the scripts will produce columns using national char types. For example, with an Oracle database, the column types will be CHAR or VARCHAR2 when the option is not selected, and NCHAR, NVARCHAR2 when the option is selected.</td>
</tr>
<tr>
<td>Populate database with sample data</td>
<td>Add statements in the script to add sample data to the database.</td>
</tr>
</tbody>
</table>

Related tasks

Extract meta-schema information from database on page 277
The **New Meta-schema** dialog assists in extracting schema information from a database.

Generate a database script from meta-schema on page 359
Generate a 4gl source file to be used to create or update a database that is described in the meta-schema file.

**Connection information dialog**
The Connection information dialog gathers the details needed to connect to a database.

The Connection information dialog appears when you are creating or updating a meta-schema file, or when you are changing the connection for the DB Explorer plug-in.

![Connection information dialog](image)

**Figure 200: Connection information dialog**
The dialog can be seen as having four parts.

**Schema details**
The schema details section consists of two items. When not applicable, this section does not appear in the dialog.

**Current schema**
Identifies the current schema, if any. When you select to edit the database connection for a meta-schema file, the name of the currently connected database displays. If no
connection has been made, the field states that no schema is selected.

**Note:** In some contexts, this read-only field does not display.

**Keep schema connection information synchronized checkbox**

When selected, the schema connection information will be permanently updated for the current meta-schema (.4db or .4dbx). This check box appears as selected and read-only when you edit the database connection for an existing meta-schema file. If you edit the connection from DB Explorer, you can uncheck this option to dissociate the DB Explorer connection information from the meta-schema connection information.

**Note:** In some contexts, this read-only field does not display.

**Database Connection Information**

In the **Database Connection Information** section, select either **Use explicit settings** or **Use external settings**.

**Use explicit settings, previous connection**

You can use a previous connection that was created for the same database. The drop down list provides a list of the existing connections.

**Use explicit settings, database type**

You can enter the **Database Type** by selecting the desired type from the drop down list, and the corresponding information for that type. The **Database driver** for the database type is automatically entered. If other drivers exist, they are available in the drop down list.

**Use external settings**

Information in the FGLPROFILE configuration file is used to extract the corresponding connection information for the specified database. Genero Studio will use the schema name that you entered to check for any related entry in the FGLPROFILE configuration file, and will use those values to define the connection. See information on the FGLPROFILE file in the **Genero Business Development Language User Guide**.

**Database User Information**

In the **Database User Information** section, provide the necessary database user details. The required information varies based on the database type selected. See **Database server/user information** on page 365.

**Test Connection**

Click **Test Connection** to verify that the information is correct and that you are able to access the database.
**Database Generation Script dialog**

Generate a 4gl source file to create or update the database that is described in the meta-schema file.

![Database Generation Script dialog](image)

**Figure 201: Database Generation Script dialog**

- **Database schema**
  - Name of selected meta-schema file.

- **Target database**
  - Specify the database to use in the script.

- **Generate primary keys, secondary keys, indexes**
  - Specify whether to include primary keys, secondary keys, and indexes in the script.

- **Use national char types**
  - By default, the database creation / update scripts will generate column using standard char types. If this option is set, the scripts will produce columns using national char types. For example, with an Oracle database, the column types will be CHAR or VARCHAR2 when the option is not selected, and NCHAR, NVARCHAR2 when the option is selected.

- **Populate database with sample data**
  - Add statements in the script to add sample data to the database.

**Related tasks**

- Generate a database script from meta-schema on page 359
Generate a 4gl source file to be used to create or update a database that is described in the meta-schema file.

**Revert schema changes dialog**
Changes made to the meta-schema can be reverted with the **Revert schema changes** dialog.

![Figure 202: Revert changes dialog](image)

**Properties to revert**
Lists all elements in the meta-schema that have been modified. Check the elements you wish to restore to its status when the schema was last extracted.

**Resulting actions**
Lists dependencies affected by the modified elements.

**Views**
Information about Meta-schema Manager views.

- [DB Schemas tab](#) on page 374
- [Filter view](#) on page 375
**DB Schemas tab**

The **DB Schemas** view displays all database meta-schema files associated with the project.

A database meta-schema is added to the **DB Schemas** view when you:

- Save a new database meta-schema file (**4db** or **4dbx**) in the current project; the Meta-schema is available in the **DB Schemas** view when you work on projects in that workspace.
- Set the **GSTSCHEMANAMES** on page 167 environment variable; the available meta-schemas are available in the **DB Schemas** view for all projects.

You can find files more easily by using the **Filter items** text box at the bottom of the **DB Schemas** view.

Use the integrated Toolbar to:

**Open a meta-schema**
Open the meta-schema in Meta-schema Manager.

**Update from database**
See **Update a meta-schema from database** on page 359.

**Sort column in database order**
Change the sort order of the columns, from alphabetical order (the default) to the order in which the columns are defined in the database.
Filter view
The Filter View dialog allows you to hide and show items on a diagram.

![Select items to filter](image)

Figure 204: Filter View

Meta-Schema Manager preferences
Set default preferences for Meta-Schema Manager and DB Explorer.
Select Tools > Preferences > Meta-schema Manager to access these preferences.

DB Explorer
Table 118: DB Explorer options

<table>
<thead>
<tr>
<th>Preference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of results for show/edit data field</td>
<td>Limit the number of rows returned by DB Explorer. When set to zero, unlimited rows can be returned.</td>
</tr>
<tr>
<td>Verbose mode check box</td>
<td>When checked, information about the executed query displays. For a successful query execution, a message like the following displays: Executed query: <code>select userid, password from signon</code> If an error occurs, information about the SQL error displays. For example, if you edit the query and provide a wrong column name, an error message like the following displays: <code>error (-6372): near &quot;wrong_userid&quot;: syntax error Connection id: C:/Users/name/xxx/officestore.db Executed query: insert wrong_userid, password from signon</code></td>
</tr>
</tbody>
</table>

Related concepts
Setting Preferences on page 128
Customize Genero Studio to meet your needs.

**Meta-schema diagram context menu**
Right-click on a meta-schema diagram or item in the Database Structure view for a context menu of options.

**Table 119: Meta-schema diagram context menu**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>Edit properties of the item selected.</td>
</tr>
<tr>
<td>Add Table</td>
<td>See Add new tables and columns on page 349.</td>
</tr>
<tr>
<td>Add Column</td>
<td>See Add new tables and columns on page 349.</td>
</tr>
<tr>
<td>Add Constraint or Index</td>
<td>See Add constraints or indexes on page 350.</td>
</tr>
<tr>
<td>Edit Constraint or Index</td>
<td>See Add constraints or indexes on page 350.</td>
</tr>
<tr>
<td>Add Foreign Key</td>
<td>See Add foreign keys on page 351.</td>
</tr>
<tr>
<td>Edit Foreign Key</td>
<td>See Add foreign keys on page 351.</td>
</tr>
<tr>
<td>Insert Column Before/After</td>
<td>See Add new tables and columns on page 349.</td>
</tr>
<tr>
<td>Revert</td>
<td>See Revert schema changes dialog on page 373.</td>
</tr>
<tr>
<td>Layout</td>
<td>Rearrange the items in the diagram.</td>
</tr>
<tr>
<td>Advanced Properties</td>
<td>Specify the extraction and/or generation options. See Advanced Properties dialog on page 367.</td>
</tr>
<tr>
<td>Filter Items...</td>
<td>The Filter View dialog allows you to hide and show items on a diagram.</td>
</tr>
<tr>
<td>Locate in Diagram</td>
<td>This action brings focus in the diagram to the selected item. If the selected object is not visible in the current view, the Meta-schema Manager tries to find another view where the object is visible. If no view is found, you are prompted to make the object visible in the current view or to create a new view.</td>
</tr>
</tbody>
</table>

**Meta-schema Manager error messages**
A list of Meta-schema Manager error messages. For messages that are not self-explanatory, additional information is provided.

**Table 120: Meta-schema Manager Error Messages**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-11000</td>
<td>Unexpected error.</td>
</tr>
<tr>
<td></td>
<td>An unexpected error occurred.</td>
</tr>
<tr>
<td>GS-11001</td>
<td>Error while reading file schema: description.</td>
</tr>
<tr>
<td></td>
<td>An error occurred while reading the file file.</td>
</tr>
<tr>
<td></td>
<td>The details about the error can be found in the description part of the message.</td>
</tr>
<tr>
<td>GS-11002</td>
<td>Cannot find codec.</td>
</tr>
<tr>
<td></td>
<td>The appropriate codec cannot be found to read a file.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-11003</td>
<td>Cannot find codec for encoding encoding. The appropriate codec for encoding encoding cannot be found.</td>
</tr>
<tr>
<td>GS-11004</td>
<td>Error while creating new schema document. The Meta-schema Manager failed to create a new document.</td>
</tr>
<tr>
<td>GS-11005</td>
<td>Dynamic property property not found for node node. A dynamic property used in the meta-schema is not found in the current template. Check the appropriate template is selected before opening the meta-schema. Failed to read dynamic property. An error occurred while reading dynamic properties. The meta-schema file might be corrupted.</td>
</tr>
<tr>
<td>GS-11050</td>
<td>Schema file defined in GSTSCHEMANAMES cannot be found in FGLDBPATH. The file cannot be found.</td>
</tr>
<tr>
<td>GS-11051</td>
<td>Failed to load file file. An error occurred while loading the file file.</td>
</tr>
<tr>
<td>GS-11052</td>
<td>Failed to open file file for writing. Check the file permissions and those of its owner directory.</td>
</tr>
<tr>
<td>GS-11053</td>
<td>Failed to open file file for reading. Check the file permissions and those of its owner directory.</td>
</tr>
<tr>
<td>GS-11054</td>
<td>File already exists. Check the file name and directory.</td>
</tr>
<tr>
<td>GS-11100</td>
<td>Failed to locate node. The specified database object cannot be found in the schema.</td>
</tr>
<tr>
<td>GS-11101</td>
<td>Unexpected schema. An unexpected schema object has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11102</td>
<td>Missing schema name. The schema name cannot be found. This property is required and the loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11103</td>
<td>Missing schema property property. This property is required, but it cannot be found. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11104</td>
<td>Invalid schema property property. The schema property property has an invalid value. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>GS-11105</td>
<td>Schema contains no table or column. No valid tables have been found in the schema. A valid table is a table in which <em>is not flagged</em> has been removed and which contains at least one column in which <em>is not flagged</em> has been removed.</td>
</tr>
<tr>
<td>GS-11106</td>
<td>Empty schema file. The schema document contains no information.</td>
</tr>
<tr>
<td>GS-11107</td>
<td>File is saved using a newer version of Genero Studio. Please upgrade Genero Studio. The schema document has been saved with a newer version of Genero Studio. You will need to upgrade Genero Studio to be able to open it.</td>
</tr>
<tr>
<td>GS-11108</td>
<td>Schema name is not lowercase, its use in 4GL source will generate errors. The schema name contains uppercase characters. Using it in 4GL programs might generate compilation errors. Use lowercase for the Schema name.</td>
</tr>
<tr>
<td>GS-11109</td>
<td>The usage of both CHAR/VARCHAR and NCHAR/NVARCHAR columns is not supported by all databases. The schema contains columns with mutually exclusive types. For instance it contains both CHAR / VARCHAR and NCHAR / NVARCHAR columns.</td>
</tr>
<tr>
<td>GS-11110</td>
<td>Schema name is invalid, its use in 4GL source will generate errors. The schema name contains characters not supported by the Genero language.</td>
</tr>
<tr>
<td>GS-11111</td>
<td>Unexpected table. An unexpected table object has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11112</td>
<td>Missing table name. The table name cannot be found. This property is required and the loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11113</td>
<td>Missing table property <em>property</em>. The table property <em>property</em> is required but it cannot be found. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11114</td>
<td>Invalid table property <em>property</em>. The table property <em>property</em> has an invalid value. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11115</td>
<td>Schema already contains a table <em>table</em>. The table has already been found in the 4db or 4dbx document. The document cannot contain two tables with the same name.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>GS-1116</td>
<td>Table <em>table</em> is not found. Creating temporary table. The table cannot be found, and it is required by another database object. A surrogate table will be created.</td>
</tr>
<tr>
<td>GS-1117</td>
<td>Table contains no columns. The table does not contain any columns.</td>
</tr>
<tr>
<td>GS-1121</td>
<td>Unexpected column. An unexpected column object has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-1122</td>
<td>Missing column name. The column name cannot be found. This property is required and the loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-1123</td>
<td>Missing column property <em>property</em>. The column property <em>property</em> is required but it cannot be found. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-1124</td>
<td>Table <em>table</em> already contains a column <em>column</em>. The column has already been found in the table. The document cannot contain two columns with the same name within the same table.</td>
</tr>
<tr>
<td>GS-1125</td>
<td>Constraint or index <em>index</em> already contains a column <em>column</em>. The column has already been found in the index. The document cannot contain two columns with the same name within the same index.</td>
</tr>
<tr>
<td>GS-1126</td>
<td>Foreign key <em>key</em> already contains a column <em>column</em>. The document cannot contain two columns with the same name within the same foreign key.</td>
</tr>
<tr>
<td>GS-1127</td>
<td>Column <em>column</em> is not found in table <em>table</em>. Creating temporary column. The column cannot be found in the table and it is required by another database object. A surrogate column will be created.</td>
</tr>
<tr>
<td>GS-1128</td>
<td>Column <em>column</em> in table <em>table</em> has unsupported type: <em>type</em>. The type is not supported for column within table. An unsupported type is of form encoded=(type, length), and can be generated while extracting database schemas.</td>
</tr>
<tr>
<td>GS-1129</td>
<td>Column <em>column</em> in table <em>table</em> has invalid type: <em>type</em>. The type is invalid for column within table. An invalid type is not handled by the 4GL language.</td>
</tr>
<tr>
<td>GS-1131</td>
<td>Unexpected foreign key. An unexpected foreign key object has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-11132</td>
<td>Missing foreign key name.</td>
</tr>
<tr>
<td></td>
<td>The foreign key name cannot be found. This property is required, and the</td>
</tr>
<tr>
<td></td>
<td>loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11133</td>
<td>Missing foreign key property <em>property</em>.</td>
</tr>
<tr>
<td></td>
<td>The foreign key property <em>property</em> is required, but it cannot be found.</td>
</tr>
<tr>
<td></td>
<td>The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11134</td>
<td>Schema already contains a foreign key <em>key</em>.</td>
</tr>
<tr>
<td></td>
<td>The foreign key has already been found in the 4db or 4dbx document.</td>
</tr>
<tr>
<td></td>
<td>The document cannot contain two foreign keys with the same name.</td>
</tr>
<tr>
<td>GS-11135</td>
<td>Schema already contains a foreign key <em>key</em>. Renaming user-defined foreign</td>
</tr>
<tr>
<td></td>
<td>key to <em>name</em>.</td>
</tr>
<tr>
<td></td>
<td>The foreign key has already been found in the 4db or 4dbx document.</td>
</tr>
<tr>
<td></td>
<td>The document cannot contain two foreign keys with the same name.</td>
</tr>
<tr>
<td></td>
<td>The foreign key will be renamed to <em>name</em>.</td>
</tr>
<tr>
<td>GS-11136</td>
<td>Foreign key <em>key</em> is not found. Creating temporary foreign key.</td>
</tr>
<tr>
<td></td>
<td>The foreign key <em>key</em> cannot be found, and it is required by another</td>
</tr>
<tr>
<td></td>
<td>database object. A surrogate foreign key will be created.</td>
</tr>
<tr>
<td>GS-11137</td>
<td>Table <em>table</em> is not found for foreign key <em>key</em>.</td>
</tr>
<tr>
<td></td>
<td>The table the foreign key refers to cannot be found. The loading of the</td>
</tr>
<tr>
<td></td>
<td>schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11138</td>
<td>Column <em>column</em> is not found in table <em>table</em> for foreign key <em>key</em>.</td>
</tr>
<tr>
<td></td>
<td>The column the foreign key refers to cannot be found in the table.</td>
</tr>
<tr>
<td></td>
<td>The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11139</td>
<td>Foreign key <em>key</em> does not contain any table column.</td>
</tr>
<tr>
<td></td>
<td>The foreign key does not contain any columns.</td>
</tr>
<tr>
<td>GS-11140</td>
<td>Foreign key <em>key</em> columns <em>column</em> and <em>reference</em> have different data</td>
</tr>
<tr>
<td></td>
<td>types.</td>
</tr>
<tr>
<td></td>
<td>The foreign key columns have mismatching data types.</td>
</tr>
<tr>
<td>GS-11141</td>
<td>Unexpected constraint or index.</td>
</tr>
<tr>
<td></td>
<td>An unexpected index object has been found in the 4db or 4dbx document.</td>
</tr>
<tr>
<td></td>
<td>The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11142</td>
<td>Missing constraint or index name.</td>
</tr>
<tr>
<td></td>
<td>The index name cannot be found. This property is required and the loading</td>
</tr>
<tr>
<td></td>
<td>of the schema file cannot continue.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-11143</td>
<td><strong>Missing constraint or index property</strong> <code>property</code>.</td>
</tr>
<tr>
<td></td>
<td>The index property is required but it cannot be found. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11144</td>
<td><strong>Table</strong> <code>table</code> already contains a constraint or index named <code>index</code>.</td>
</tr>
<tr>
<td></td>
<td>The index has already been found in the table <code>table</code>. The document cannot contain two indexes with the same name within the same table.</td>
</tr>
<tr>
<td>GS-11145</td>
<td><strong>Table</strong> <code>table</code> already contains a constraint or index named <code>index</code>. Renaming user-defined constraint or index to <code>name</code>.</td>
</tr>
<tr>
<td></td>
<td>The index has already been found in the table. The document cannot contain two indexes with the same name within the same table. The index will be renamed to <code>name</code>.</td>
</tr>
<tr>
<td>GS-11146</td>
<td><strong>Constraint or index</strong> <code>index</code> is not found. Creating temporary constraint or index.</td>
</tr>
<tr>
<td></td>
<td>The index cannot be found in the table and it is required by another database object. A surrogate index will be created.</td>
</tr>
<tr>
<td>GS-11147</td>
<td><strong>Column</strong> <code>column</code> is not found in table <code>table</code> for constraint or index <code>index</code>.</td>
</tr>
<tr>
<td></td>
<td>The column that the index refers to cannot be found in the table. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11148</td>
<td><strong>Table</strong> <code>table</code> contains more than one primary key.</td>
</tr>
<tr>
<td></td>
<td>The table contains more than one primary index. Only one primary index can be defined for a table.</td>
</tr>
<tr>
<td>GS-11149</td>
<td><strong>Constraint or index</strong> <code>index</code> does not contain any table column.</td>
</tr>
<tr>
<td></td>
<td>The index does not contain any columns.</td>
</tr>
<tr>
<td>GS-11150</td>
<td><strong>Column</strong> <code>column</code> allows null values but is referred by primary key <code>primary key</code>.</td>
</tr>
<tr>
<td></td>
<td>A column referred in a primary key doesn't have its <strong>NOT NULL</strong> flag set.</td>
</tr>
<tr>
<td>GS-11151</td>
<td><strong>Unexpected column validation attributes.</strong></td>
</tr>
<tr>
<td></td>
<td>An unexpected index object has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11152</td>
<td><strong>Missing column name for validation attributes.</strong></td>
</tr>
<tr>
<td></td>
<td>The column name cannot be found. This property is required and the loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11153</td>
<td><strong>Column '%%' already contains validation attributes.</strong></td>
</tr>
<tr>
<td></td>
<td>Column validation attributes have already been defined for the column. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>GS-11154</td>
<td>Column '%1' is not found in table '%2' for column validation attributes. Column validation attributes refer to a non-existing column. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11161</td>
<td>Unexpected column video attributes. An unexpected index object has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11162</td>
<td>Missing column name for video attributes. The column name cannot be found. This property is required and the loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11163</td>
<td>Column '%1' already contains video attributes. Video attributes have already been defined for the column. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11164</td>
<td>Column '%1' is not found in table '%2' for column video attributes. Column video attributes refer to a non-existing column. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11171</td>
<td>Unexpected user attributes. An unexpected user attribute has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11172</td>
<td>Missing user attribute name. The user attribute name is required but cannot be found. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11173</td>
<td>Missing user attribute property '%1'. The user attribute property property is required but cannot be found. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11174</td>
<td>Object already contains a user attribute '%1'. A user attribute has already been defined for the database object. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11181</td>
<td>Unexpected layout data section. An unexpected layout section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11182</td>
<td>Missing layout data property property. The layout property property is required but it cannot be found. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-11191</td>
<td>Unexpected database data section. An unexpected data section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11192</td>
<td>Missing database data property property. The data property property is required but it cannot be found. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11193</td>
<td>Document already contains database value for identifier. Database data has already been found in the 4db or 4dbx document. The document cannot contain twice data for the same identifier.</td>
</tr>
<tr>
<td>GS-11194</td>
<td>Document contains invalid identifier identifier for database value (value: value). Database value will be ignored and removed. The database data identifier doesn't match any database object. The associated value is ignored and will be discarded upon save.</td>
</tr>
<tr>
<td>GS-11201</td>
<td>Unexpected dynamic property. An unexpected dynamic property has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11202</td>
<td>Missing dynamic property attribute attribute. The dynamic property attribute attribute is required but it cannot be found. The loading of the schema file cannot continue.</td>
</tr>
<tr>
<td>GS-11203</td>
<td>Object doesn't support dynamic property property. A dynamic property has been read for a database object but this property is not defined for this object. Check the appropriate template is selected before opening the meta-schema.</td>
</tr>
<tr>
<td>GS-11204</td>
<td>Object already contains a dynamic property property. A dynamic property has been defined twice for a database object.</td>
</tr>
<tr>
<td>GS-11210</td>
<td>Invalid model node info version. Model node info for type type not found. Failed to add property property to model node info: a dynamic property with this name already exists Failed to add property property to model node info: a static property with this name already exists Failed to add property property to model node info: a property with this name was not found Information about model node is invalid.</td>
</tr>
<tr>
<td>GS-11220</td>
<td>Name cannot be empty. The name of a database object is empty.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-11221</td>
<td>Same name cannot be applied to more than one table.</td>
</tr>
<tr>
<td></td>
<td>Two tables cannot have the same name in the meta-schema.</td>
</tr>
<tr>
<td>GS-11222</td>
<td>Same name cannot be applied to more than one column of same table.</td>
</tr>
<tr>
<td></td>
<td>Two columns of the same table cannot have the same name in the meta-schema.</td>
</tr>
<tr>
<td>GS-11223</td>
<td>Same name cannot be applied to more than one foreign key.</td>
</tr>
<tr>
<td></td>
<td>Two foreign keys cannot have the same name in the meta-schema.</td>
</tr>
<tr>
<td>GS-11224</td>
<td>Same name cannot be applied to more than one constraint or index of same table.</td>
</tr>
<tr>
<td></td>
<td>Two constraints or indexes of the same table cannot have the same name in the meta-schema.</td>
</tr>
<tr>
<td>GS-11225</td>
<td>Table name is invalid.</td>
</tr>
<tr>
<td></td>
<td>A table cannot be named with the provided name.</td>
</tr>
<tr>
<td>GS-11226</td>
<td>Column name is invalid.</td>
</tr>
<tr>
<td></td>
<td>A column cannot be named with the provided name.</td>
</tr>
<tr>
<td>GS-11227</td>
<td>Database object name contains invalid character(s).</td>
</tr>
<tr>
<td></td>
<td>The name of the database object contains invalid characters.</td>
</tr>
<tr>
<td>GS-11228</td>
<td>The name name has been converted to other name as database object name contains invalid character(s).</td>
</tr>
<tr>
<td></td>
<td>The original name of the database object contains invalid characters. The database object has been renamed to a new name.</td>
</tr>
<tr>
<td>GS-11270</td>
<td>Unexpected extraction options.</td>
</tr>
<tr>
<td></td>
<td>An unexpected extraction options section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11271</td>
<td>Unexpected extraction settings.</td>
</tr>
<tr>
<td></td>
<td>An unexpected extraction settings section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11272</td>
<td>Unexpected generation options.</td>
</tr>
<tr>
<td></td>
<td>An unexpected generation options section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11273</td>
<td>Unexpected generation settings.</td>
</tr>
<tr>
<td></td>
<td>An unexpected generation settings section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11303</td>
<td>The database object name has been newly added in schema.</td>
</tr>
<tr>
<td></td>
<td>The database object has been added to the schema and is not yet found in the database. Creating a database update script will generate SQL instructions to add this object to the database.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-11304</td>
<td>The database object <em>name</em> has been marked as removed from schema. The database object has been removed from the schema and can still be found in the database. Creating a database update script will generate SQL instructions to remove this object from the database.</td>
</tr>
<tr>
<td>GS-11310</td>
<td>Name has been modified from <em>name</em> to <em>other name</em>. The database object has been renamed in the schema and can still be found in the database. Creating a database update script will generate SQL instructions to update this object in the database.</td>
</tr>
<tr>
<td>GS-11311</td>
<td>Database object <em>name</em> has an invalid uuid. A new one has been generated (<em>uuid</em>). The internal identifier of the database object is invalid and has been replaced with a new one.</td>
</tr>
<tr>
<td>GS-11315</td>
<td>Column type has been changed from <em>type</em> to <em>other type</em>. The type of the column has been changed in the schema. Creating a database update script will generate SQL instructions to update the column in the database.</td>
</tr>
<tr>
<td>GS-11316</td>
<td>Not null property has been changed from <em>value</em> to <em>other value</em>. The value of the 'Not null' property of the column has been changed in the schema. Creating a database update script will generate SQL instructions to update the column in the database.</td>
</tr>
<tr>
<td>GS-11318</td>
<td>Column <em>column</em> in table <em>table</em> uses Informix specific type: <em>type</em>. The column uses an Informix® specific data type.</td>
</tr>
<tr>
<td>GS-11319</td>
<td>Invalid qualifiers for column '%1' in table '%2'. DATETIME or INTERVAL qualifiers break the qualifiers rules for the data type. See the <em>Genero Business Development Language User Guide</em> for further details.</td>
</tr>
<tr>
<td>GS-11320</td>
<td>Constraint or index type has been changed from <em>type</em> to <em>other type</em>. The type of the constraint has been changed in the schema. Creating a database update script will generate SQL instructions to update the constraint in the database.</td>
</tr>
<tr>
<td>GS-11322</td>
<td>Constraint or index columns of <em>name</em> has been changed from <em>columns to other columns</em>. The columns of the constraint have been changed in the schema. Creating a database update script will generate SQL instructions to update the constraint in the database.</td>
</tr>
<tr>
<td>GS-11323</td>
<td>Constraint or index <em>name</em> column order of <em>name</em> conflicts with column order of <em>foreign key</em>. Indicates the order of columns in the foreign key is not the same as the one defined for the constraint or index.</td>
</tr>
<tr>
<td>GS-11325</td>
<td>Foreign key table has been changed from <em>table to other table</em>. The table associated to the foreign key has been changed in the schema. Creating a database update script will generate SQL instructions to update the foreign key in the database.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-11326</td>
<td>Foreign key referenced table has been changed from <em>table</em> to <em>other table</em>.</td>
</tr>
<tr>
<td></td>
<td>The reference table associated to the foreign key has been changed in the schema. Creating a database update script will generate SQL instructions to update the foreign key in the database.</td>
</tr>
<tr>
<td>GS-11327</td>
<td>Foreign key columns of <em>foreign key</em> has been changed from <em>columns</em> to <em>other columns</em>.</td>
</tr>
<tr>
<td></td>
<td>The columns associated to the foreign key have been changed in the schema. Creating a database update script will generate SQL instructions to update the foreign key in the database.</td>
</tr>
<tr>
<td>GS-11328</td>
<td>Foreign key referenced columns of <em>foreign key</em> has been changed from <em>columns</em> to <em>other columns</em>.</td>
</tr>
<tr>
<td></td>
<td>The reference columns associated to the foreign key have been changed in the schema. Creating a database update script will generate SQL instructions to update the foreign key in the database.</td>
</tr>
<tr>
<td>GS-11329</td>
<td>Cascade delete property has been changed from <em>value</em> to <em>other value</em>.</td>
</tr>
<tr>
<td></td>
<td>The value of the 'Cascade delete' property of the foreign key has been changed in the schema. Creating a database update script will generate SQL instructions to update the foreign key in the database.</td>
</tr>
<tr>
<td>GS-11330</td>
<td>Foreign key <em>name</em> column order of <em>column</em> conflicts with column order of <em>other column</em>.</td>
</tr>
<tr>
<td></td>
<td>Several columns are defined in the foreign key with the same order. The order of the column is automatically updated to solve the issue.</td>
</tr>
<tr>
<td>GS-11340</td>
<td>Constraints or indexes constraint and <em>other constraint</em> are defined on same set of columns.</td>
</tr>
<tr>
<td></td>
<td>Both constraints are defined on the same set of columns.</td>
</tr>
<tr>
<td>GS-11341</td>
<td>Foreign key <em>foreign key</em> references columns which are not part of primary key or unique constraint.</td>
</tr>
<tr>
<td></td>
<td>No primary key or secondary key is defined on the columns referenced by the foreign key.</td>
</tr>
<tr>
<td>GS-11342</td>
<td>Constraint cannot be placed on a BYTE or TEXT column.</td>
</tr>
<tr>
<td></td>
<td>A unique constraint has been defined on column of type BYTE or TEXT.</td>
</tr>
<tr>
<td>GS-11343</td>
<td>Non-unique index index conflicts with constraint constraint.</td>
</tr>
<tr>
<td></td>
<td>A non-unique index has been defined on the same set of columns has a unique constraint.</td>
</tr>
<tr>
<td>GS-11360</td>
<td>Foreign key references an inactive table.</td>
</tr>
<tr>
<td></td>
<td>The table referenced by the foreign key is flagged as inactive whereas the table associated to the foreign key is active.</td>
</tr>
<tr>
<td>GS-11361</td>
<td>Foreign key table is inactive.</td>
</tr>
<tr>
<td></td>
<td>The table associated to the foreign key is flagged as inactive whereas the table referenced by the foreign key is active.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-11370</td>
<td>Name has been modified to '%1' but matches value in database. The property of the database object has been changed in both the meta-schema and the database with the same value. The change becomes useless and can be reverted.</td>
</tr>
<tr>
<td>GS-11371</td>
<td>Column type has been changed to '%1' but matches value in database. The property of the database object has been changed in both the meta-schema and the database with the same value. The change becomes useless and can be reverted.</td>
</tr>
<tr>
<td>GS-11372</td>
<td>Not null property has been changed to '%1' but matches value in database. The property of the database object has been changed in both the meta-schema and the database with the same value. The change becomes useless and can be reverted.</td>
</tr>
<tr>
<td>GS-11373</td>
<td>Constraint or index type has been changed to '%1' but matches value in database. The property of the database object has been changed in both the meta-schema and the database with the same value. The change becomes useless and can be reverted.</td>
</tr>
<tr>
<td>GS-11374</td>
<td>Constraint or index columns of '%1' has been changed to '%2' but matches value in database. The property of the database object has been changed in both the meta-schema and the database with the same value. The change becomes useless and can be reverted.</td>
</tr>
<tr>
<td>GS-11375</td>
<td>Foreign key columns of '%1' has been changed to '%2' but matches value in database. The property of the database object has been changed in both the meta-schema and the database with the same value. The change becomes useless and can be reverted.</td>
</tr>
<tr>
<td>GS-11376</td>
<td>Foreign key referenced columns of '%1' has been changed to '%2' but matches value in database. The property of the database object has been changed in both the meta-schema and the database with the same value. The change becomes useless and can be reverted.</td>
</tr>
<tr>
<td>GS-11377</td>
<td>Cascade delete property has been changed to '%1' but matches value in database. The property of the database object has been changed in both the meta-schema and the database with the same value. The change becomes useless and can be reverted.</td>
</tr>
<tr>
<td>GS-11378</td>
<td>Foreign key table has been changed to '%1' but matches value in database. The property of the database object has been changed in both the meta-schema and the database with the same value. The change becomes useless and can be reverted.</td>
</tr>
<tr>
<td>GS-11379</td>
<td>Foreign key referenced table has been changed to '%1' but matches value in database. The property of the database object has been changed in both the meta-schema and the database with the same value. The change becomes useless and can be reverted.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-11400</td>
<td>Unknown tag <em>tag</em>.</td>
</tr>
<tr>
<td></td>
<td>The meta-schema document contains unknown data. The schema file is corrupted and cannot be read.</td>
</tr>
<tr>
<td>GS-11550</td>
<td>Wrong line format.</td>
</tr>
<tr>
<td></td>
<td>The data contained in the file being imported to a meta-schema has an unsupported format. The data cannot be imported.</td>
</tr>
<tr>
<td>GS-11551</td>
<td>Can't convert type for column <em>column</em> of table <em>table</em>.</td>
</tr>
<tr>
<td></td>
<td>The type of the table column defined in the imported table is unsupported. The data cannot be imported.</td>
</tr>
<tr>
<td>GS-11552</td>
<td>Can't convert length for column <em>column</em> of table <em>table</em>.</td>
</tr>
<tr>
<td></td>
<td>The length of the table column defined in the imported table is invalid. The data cannot be imported.</td>
</tr>
<tr>
<td>GS-11553</td>
<td>Can't convert order for column <em>column</em> of table <em>table</em>.</td>
</tr>
<tr>
<td></td>
<td>The column has an unsupported order value. The import of the sch file cannot continue.</td>
</tr>
<tr>
<td>GS-11600</td>
<td>Compilation generated an empty sch file.</td>
</tr>
<tr>
<td></td>
<td>The compilation of the schema document generated an empty sch file. This can happen when no valid table columns have been found in the document.</td>
</tr>
<tr>
<td>GS-11650</td>
<td>The extraction process task can't be generated.</td>
</tr>
<tr>
<td></td>
<td>An error occurred while trying to create the database extraction process.</td>
</tr>
<tr>
<td>GS-11651</td>
<td>Database name can't be empty.</td>
</tr>
<tr>
<td></td>
<td>An empty database name has been found while trying to set up the database connection.</td>
</tr>
<tr>
<td>GS-11652</td>
<td>Schema file name can't be empty.</td>
</tr>
<tr>
<td></td>
<td>An empty file name has been found while trying to generate the schema document.</td>
</tr>
<tr>
<td>GS-11653</td>
<td>Preparing schema extraction...</td>
</tr>
<tr>
<td></td>
<td>This message indicates the schema extraction process is being set up.</td>
</tr>
<tr>
<td>GS-11654</td>
<td>Schema extraction started...</td>
</tr>
<tr>
<td></td>
<td>This message indicates the schema extraction process has started.</td>
</tr>
<tr>
<td>GS-11655</td>
<td>Merging extracted schema with existing one...</td>
</tr>
<tr>
<td></td>
<td>This message indicates the schema extraction process tries to merge the existing schema document with the extracted one.</td>
</tr>
<tr>
<td>GS-11656</td>
<td>Generating schema id...</td>
</tr>
<tr>
<td></td>
<td>This message indicates a new schema identifier is being generated.</td>
</tr>
<tr>
<td>GS-11657</td>
<td>Schema extraction successful.</td>
</tr>
<tr>
<td></td>
<td>This message indicates the schema extraction process has successfully completed.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-11658</td>
<td>Schema extraction failed.</td>
</tr>
<tr>
<td></td>
<td>The schema extraction process has failed at some point. Refer to other messages to get additional information.</td>
</tr>
<tr>
<td>GS-11659</td>
<td>Schema extraction aborted.</td>
</tr>
<tr>
<td></td>
<td>The schema extraction process has been aborted by the user.</td>
</tr>
<tr>
<td>GS-11660</td>
<td>Failed to move temporary schema file to destination.</td>
</tr>
<tr>
<td></td>
<td>An error occurred while trying to move the temporary schema file to its destination.</td>
</tr>
<tr>
<td>GS-11661</td>
<td>Failed to create schema backup file.</td>
</tr>
<tr>
<td></td>
<td>An error occurred while trying to create a backup of the schema file.</td>
</tr>
<tr>
<td>GS-11662</td>
<td>Failed to remove schema backup file.</td>
</tr>
<tr>
<td></td>
<td>An error occurred while trying to delete a backup of the schema file.</td>
</tr>
<tr>
<td>GS-11700</td>
<td>The connection test process task can't be generated.</td>
</tr>
<tr>
<td></td>
<td>An error occurred while trying to create the database connection test process.</td>
</tr>
<tr>
<td>GS-11750</td>
<td>Generating documentation for meta-schema...</td>
</tr>
<tr>
<td></td>
<td>This message indicates documentation for the schema is being generated.</td>
</tr>
<tr>
<td>GS-11751</td>
<td>Documentation generated at location</td>
</tr>
<tr>
<td></td>
<td>This message indicates documentation for the schema has been generated and is saved at the given location.</td>
</tr>
<tr>
<td>GS-11800</td>
<td>error (error_code): error_message</td>
</tr>
<tr>
<td></td>
<td>The execution of the DB Explorer query failed. Refer to the error message for a detailed explanation.</td>
</tr>
<tr>
<td>GS-11810</td>
<td>Warning: no row affected.</td>
</tr>
<tr>
<td></td>
<td>The executed DB Explorer query did not affect any row.</td>
</tr>
<tr>
<td>GS-11830</td>
<td>Executed query: query</td>
</tr>
<tr>
<td></td>
<td>This message displays the executed DB Explorer query.</td>
</tr>
<tr>
<td>GS-11831</td>
<td>Success: number_of_rows row(s) affected.</td>
</tr>
<tr>
<td></td>
<td>This message displays the number of rows affected by the DB Explorer query.</td>
</tr>
<tr>
<td>GS-11832</td>
<td>Select execution successful : number_of_rows row(s) returned.</td>
</tr>
<tr>
<td></td>
<td>This message displays the number of rows returned by a SELECT statement by the DB Explorer.</td>
</tr>
<tr>
<td>GS-11833</td>
<td>Sql command successful</td>
</tr>
<tr>
<td></td>
<td>The DB Explorer query was successful.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-11834</td>
<td>Limit of <code>number_of_rows</code> row(s) reached.</td>
</tr>
<tr>
<td></td>
<td>The DB Explorer query has returned more rows than the limit set in the DB Explorer preferences.</td>
</tr>
<tr>
<td>GS-11835</td>
<td>Connection id: <code>connection_information</code></td>
</tr>
<tr>
<td></td>
<td>This message displays the connection parameter used by the DB Explorer to connect to the database.</td>
</tr>
<tr>
<td>GS-11850</td>
<td>Failed to extract database meta data.</td>
</tr>
<tr>
<td></td>
<td>The extraction of the meta-schema has failed. Refer to the previous messages for more information.</td>
</tr>
<tr>
<td>GS-11851</td>
<td>Failed to connect to database.</td>
</tr>
<tr>
<td></td>
<td>The connection to the database has failed. Ensure the appropriate connection information has been provided.</td>
</tr>
<tr>
<td>GS-11852</td>
<td>Cannot instantiate database driver <code>driver</code>.</td>
</tr>
<tr>
<td></td>
<td>The database driver cannot be loaded. Check the driver name is correct and accessible.</td>
</tr>
<tr>
<td>GS-11853</td>
<td>Unsupported database <code>database</code>.</td>
</tr>
<tr>
<td></td>
<td>This database engine is not supported by the extraction tool.</td>
</tr>
<tr>
<td>GS-11860</td>
<td>The type <code>type</code> of column &quot;<code>table.column</code>&quot; is unsupported - table is ignored.</td>
</tr>
<tr>
<td></td>
<td>The data type of the table column is not supported by the extraction tool. The extraction of the table is canceled.</td>
</tr>
<tr>
<td>GS-11861</td>
<td><code>type</code> column <code>table.column</code> is ignored.</td>
</tr>
<tr>
<td></td>
<td>The data type of the table column has been flagged to be ignored by the extraction tool. The table definition will not contain this column.</td>
</tr>
<tr>
<td>GS-11862</td>
<td>Foreign key <code>foreign key</code> is ignored due to missing table.</td>
</tr>
<tr>
<td></td>
<td>The foreign key references a table that was not found during extraction. The meta-schema will not contain this foreign key.</td>
</tr>
<tr>
<td>GS-11863</td>
<td>Failed to extract indexes for table &quot;<code>table</code>&quot;.</td>
</tr>
<tr>
<td></td>
<td>A SQL error occurred while indexes were extracted for the table.</td>
</tr>
<tr>
<td>GS-11864</td>
<td>Failed to extract foreign keys.</td>
</tr>
<tr>
<td></td>
<td>A SQL error occurred while the foreign keys were extracted from the schema.</td>
</tr>
<tr>
<td>GS-11865</td>
<td>No column extracted for table &quot;<code>table</code>&quot; - table is ignored.</td>
</tr>
<tr>
<td></td>
<td>No table column has been extracted by the extraction tool. The extraction of the table is cancelled.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| GS-11866 | Database object "name" is ignored.  
The database object has been ignored by the extraction tool. This usually happens with objects generated by the database. |
| GS-11867 | Table column "table.column" datatype [type] is unsupported and has been converted.  
The data type of the table column is not supported by the extraction tool, but the tool converted it to a similar data type. |
| GS-11870 | Unexpected conversion parameter.  
The conversion method provided to the extraction tool is invalid. |
| GS-11871 | Missing conversion method.  
The conversion method is required by the extraction tool. |
| GS-11872 | Missing auto-increment indicator.  
The parameter required for the extraction of the table column is not found. |
| GS-11873 | Missing charlen indicator.  
The parameter required for the extraction of the table column is not found. |
| GS-11874 | Missing column type indicator.  
The parameter required for the extraction of the table column is not found. |
| GS-11875 | Missing distinct types indicator.  
The parameter required for the extraction of the table column is not found. |
| GS-11876 | Missing identity indicator.  
The parameter required for the extraction of the table column is not found. |
| GS-11877 | Missing serial indicator.  
The parameter required for the extraction of the table column is not found. |
| GS-11878 | Missing datetime precision indicator.  
The parameter required for the extraction of the table column is not found. |
| GS-11880 | No database connection.  
The connection to the database is not found. |
| GS-11881 | Table name is required.  
The name of the extracted table is required by the extraction tool. |
| GS-11882 | Column name is required.  
The name of the extracted table column is required by the extraction tool. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
</table>
| GS-11883 | Database schema is required.  
The database schema / table owner is required by the extraction tool. |
| GS-11890 | Missing database driver.  
The database driver has not been provided to the extraction tool. |
| GS-11891 | Missing output file parameter.  
The meta-schema file has not been provided to the extraction tool. |
| GS-11892 | Failed to read user password.  
The password is unreadable. |
| GS-11899 | unexpected error  
An unexpected error has been encountered. Refer to the error message for more information. |

DB Explorer

The DB Explorer plug-in is a tool that allows you to view, create and modify data stored in a relational database. To design a good report, it is important to know your data. DB Explorer provides the concept of "show data", where you can see the data stored in the rows of a database table. You can view this data from the table perspective, or from the business record perspective. By knowing your data, you are better equipped to write valid expressions and reports that your readers will understand.

When testing your report design, you may need to see how the report handles specific data values. For example, you may have a sales report where you want to highlight all sales that are above (or below) a certain value. You may need to modify your table data in order to have a row that meets the criterion. In a more extreme example, you may simply have the table schema without any data records. You may need to provide the sample data yourself.

The DB Explorer plug-in exists for these reasons, and more.

Layout of the view

![DB Explorer](image)

Figure 205: Table data in DB Explorer

At the top of the view, an editable combobox allows you to select a previously written SQL query, edit the current SQL query, or enter in your own SQL query. Next to this field are three icons:
• The **Edit SQL query** icon opens the **Query Editor**.
• The **Execute query** icon executes the displayed query.
• The **Choose connection information** icon allows you to change your database connection.

Under the combobox, there are four icons for data modification.

• **Save** icon - Saves changes to the current row.
• **Insert** icon - Creates a new, blank row.
• **Copy** icon - Create a copy of the selected row.
• **Delete** icon - Delete the selected row.

The data itself displays in the scrollable table container at the bottom of the view.

**When to use**

Use DB Explorer to quickly view or make changes to your data. It is important to understand that, by using DB Explorer, you are making actual changes to the data in the tables; these changes are permanent. As such, DB Explorer is intended as a developer tool, not as a production tool.

**Limitations**

The following SQL commands are not supported:

• SQL commands that output information as text (for example, a command that shows a list of tables) will execute, however the text will not display.
• SQLite-specific commands (known as dot commands, such as .show) are not supported.

When you change or modify data using this tool, the changes are done under-the-covers by SQL statements. These SQL statements are governed by the rules of the database itself. The change needs to be valid by the rules of the database, in order for the query to run successfully.

You cannot use DB Explorer to view or edit binary data (BLOB data type). In addition, for some databases you cannot use DB Explorer to view or edit text-based large objects, such as the CLOB data type in IBM-Informix database servers.

DB Explorer will always work with the actual database structure. You may have made changes to the tables and column definitions in the database meta-schema file (.4db), but if those changes are not made to the actual database, they are ignored by DB Explorer. Conversely, if changes are made to the underlying database structure (through the use of DDL or another database tool), the meta-schema file (.4db) must be manually updated using the **Database > Update Schema** menu option.

**Open DB Explorer**

DB Explorer is one of many views available from within Genero Studio.

While you can explicitly open DB Explorer, it will open automatically as needed when you follow the steps to show or edit data.

**Using the Views menu**

To explicitly open DB Explorer, you use the **Views** menu.

Select **Window > Views > DB Explorer**.

The DB Explorer view opens.
As a next step, you will likely specify the database connection.

From the meta-schema or Data Model

When you ask to show or edit data, the result set opens for you in DB Explorer. See Show data on page 395 and Change the data on page 396.

Change connection details

In order to show or edit data, you must be connected to a relational database.

1. Open the Connection information dialog.
   - From within DB Explorer view, click the Choose Connection Information icon.
   - With the meta-schema file (.4db, .4dbx) opened in the central work area, select Database > Edit Database Connection.
   - With the meta-schema file (.4db, .4dbx) opened in the central work area, click anywhere within the white space in the meta-schema diagram and select Edit Database Connection from the contextual menu.
   - With the meta-schema file (.4db, .4dbx) opened in the central work area, in the Database Structure view, right-click on the database node and select Edit Database Connection.

2. On the Connection information page, complete the three areas.
   a) Check (or de-select) the Keep schema connection information synchronized check box.
      When selected, the schema connection information will be permanently updated for the current meta-schema (.4db or .4dbx). This check box appears as selected and read-only when you edit the database connection for an existing meta-schema file. If you edit the connection from DB Explorer, you can uncheck this option to dissociate the DB Explorer connection information from the meta-schema connection information.
   b) In the Database Connection Information section, select either Use explicit settings or Use external settings.

   Use explicit settings, previous connection
   You can use a previous connection that was created for the same database. The drop down list provides a list of the existing connections.

   Use explicit settings, database type
   You can enter the Database Type by selecting the desired type from the drop down list, and the corresponding information for that type. The Database driver for the database type is automatically entered. If other drivers exist, they are available in the drop down list.

   Use external settings
   Information in the FGLPROFILE configuration file is used to extract the corresponding connection information for the specified database. Genero
c) In the **Database User Information** section, provide the necessary database user details. The required information varies based on the database type selected. See [Database server/user information](#) on page 365.

3. Click **Test Connection** to verify your configuration.
4. Click **OK** to close the dialog.

---

**Show data**

You can show data for a table, for a set of columns, or for a business record.

You have several choices to make, to include whether you wish to view some or all columns of data, or whether to start from the meta-schema file or from DB Explorer itself. You can also view, but not edit, the data for a business record.

You cannot use DB Explorer to view or edit binary data (BLOB data type). In addition, for some databases you cannot use DB Explorer to view or edit text-based large objects, such as the CLOB data type in IBM-Informix database servers.

---

**Show table data (start with meta-schema diagram)**

Follow these steps to view data for a specific table, using the meta-schema diagram as the starting point.

1. Open the meta-schema (.4db or .4dbx) diagram.
2. Right-click anywhere within the table object and select **Show/Edit Table Data**.

DB Explorer opens, with the table data displayed.

---

**Show table data (start with DB Explorer)**

Follow these steps to view data for a specific table, using DB Explorer as the starting point.

1. Open DB Explorer.
2. Click the **Edit SQL query** icon.
   The **Query Editor** opens.
3. Select the **Edit table data** radio button.
4. From the combobox, select the table.

   **Tip:** If you were to select **Execute** at this time, the table data would display in edit mode.

   The SQL query used to select the rows from the selected table displays.
5. Select the **Execute query** radio button.
6. Click **Execute**.

   The table data displays.

---

**Show data for select columns**

Follow these steps to view data for a subset of table columns.

1. Open the meta-schema (.4db or .4dbx) diagram.
2. Within in the table object (the box containing the table details), press the CTRL key and select one or more columns.

   **Tip:** As another options, you can do your column selection from within the **Database Structure** view.
The column names of the selected columns turn red.

3. Right-click within the table object and select **Show Column(s) Data**.
   DB Explorer opens, with the data displayed for the selected columns.

**Show data for a business record**

A business record can contain data from multiple tables, depending on how it was defined. Follow these steps to view the data for a business record.

1. Open the business record (`.4rdj`) diagram.
2. Right-click in the Business Record object and select **Show Data**.

   **Tip:** As another options, you can also right-click on the business record node in the **Structure View** view.

**Change the data**

You may need to change your data, in order to test your report designs.

While you can write SQL statements, DB Explorer is designed to allow you to make data changes directly within the user interface, to the data shown.

Use DB Explorer to quickly view or make changes to your data. It is important to understand that, by using DB Explorer, you are making actual changes to the data in the tables; these changes are permanent. As such, DB Explorer is intended as a developer tool, not as a production tool.

When you change or modify data using this tool, the changes are done under-the-covers by SQL statements. These SQL statements are governed by the rules of the database itself. The change needs to be valid by the rules of the database, in order for the query to run successfully.

**Show data from a table in edit mode**

To edit the data without having to hand-write SQL statements, you must start with the meta-schema file.

1. Open the meta-schema (`.4db`) file.
2. Right-click on a table object and select **Show/Edit Table Data**.

   The table data shows in DB Explorer, in edit mode.

**Show data from a different table in edit mode**

When in edit mode, you can switch to a different table for editing.

1. Click the **Edit SQL query** icon. The **Edit SQL query** icon consists of a pad and pencil image.
   The **Query Editor** dialog opens.
2. Select the **Edit table data** radio button.
3. From the combobox, select the table.
4. Click **Execute**.

   The table data shows in DB Explorer, in edit mode.

**Update data**

Before you begin, you must be viewing the data in edit mode.

This procedure tells you the recommended method for updating one or more values.

1. Double-click a table cell.
2. Change the value in the cell.
3. Repeat for any other fields within the same record.

   **Note:** If you switch to a different row, DB Explorer will save the changes to the current row automatically.
4. When you have finished updating values for a record, click the **Save** icon. The **Save** icon is a picture of a computer disk.

**Insert a row**

Before you begin, you must be viewing the data in edit mode.

This procedure tells you the recommended method for inserting a new row.

1. Click the **Insert** icon. The **Insert** icon consists of a plus sign and an arrow.
   An empty row appears at the end of the row listing, with the cursor in the first column / field.
2. Enter a value in the field, and press **TAB** to move to the next field. To leave the field blank, simply press **TAB**.
   Repeat until all fields are populated.
3. Click the **Save** icon. The **Save** icon is a picture of a computer disk.
   
   **Note:** If you tab past the last field, DB Explorer will insert the new row for you automatically.
   The row is saved. A success message is written to the status bar.
   
   **Note:** If there is an error, review the Output view to identify the issue. See Execute a query on page 399.

**Duplicate a row**

Before you begin, you must be viewing the data in edit mode.

This procedure tells you the recommended method for creating a copy of a row. It is assumed that you will then modify one or more of the fields, before saving the copy as a new row.

1. Select a row.
2. Click the **Duplicate** icon. The **Duplicate** icon consists of "x2" inside a green circle.
3. A new row opens at the bottom of the list, with the values of the originally selected row duplicated.
4. Make modifications to the fields in the row.
   
   **Tip:** Be sure to change the value of the primary key field, to avoid returning a SQL constraint error.
5. Click the **Save** icon. The **Save** icon is a picture of a computer disk.
   The row is saved. A success message is written to the status bar.
   
   **Note:** If there is an error, review the Output view to identify the issue. See Execute a query on page 399.

**Delete a row**

Before you begin, you must be viewing the data in edit mode.

This procedure tells you the recommended method for deleting a row.

1. Select a row.
2. Click the **Delete** icon. The **Delete** icon consists of a red "X".
3. A dialog appears asking you to confirm your delete request. Click **Yes** to delete the row.
   The row is deleted. A success message is written to the status bar.

**Limit rows**

You can set a limit to the number of rows to display.

Without a limit, you could end up retrieving massive amounts of data. By default, the limit is set to 1000. You can remove all limits by entering zero (0).

When the row limit is met, a message displays in the status area of DB Explorer stating that the limit has been met.
Figure 207: Message displays when limit is met

Regardless of the limit, the data returned always starts with the first row returned by the database server. There is no mechanism to change the order of the rows fetched, or which rows are returned, without altering the SQL by hand.

To set the limit on the number of rows returned:

1. Select **Tools > Preferences**.
2. Select **Meta-Schema manager Preferences**.
3. Under **DB Explorer options**, specify the maximum number of results (or records) to display and retrieve.

**Related concepts**

- Meta-Schema Manager preferences on page 375
- Set default preferences for Meta-Schema Manager and DB Explorer.

---

**Write a SQL query by hand**

DB Explorer is designed to allow users to create or modify data using the graphical interface. You can, however, use DB Explorer to edit SQL queries you write by hand.

The following SQL commands are not supported:

- SQL commands that output information as text (for example, a command that shows a list of tables) will execute, however the text will not display.
- SQLite-specific commands (known as dot commands, such as `.show`) are not supported.

When you change or modify data using this tool, the changes are done under-the-covers by SQL statements. These SQL statements are governed by the rules of the database itself. The change needs to be valid by the rules of the database, in order for the query to run successfully.

You cannot use DB Explorer to view or edit binary data (BLOB data type). In addition, for some databases you cannot use DB Explorer to view or edit text-based large objects, such as the CLOB data type in IBM-Informix database servers.

**Note:** The Query Editor does not provide any language syntax assistance.

1. Open DB Explorer.
2. Establish a connection to a database. See Change connection details on page 394.
3. Click the **Edit SQL query** icon.
   The **Edit query** dialog opens.
4. Enter your query.
   You have three options:
   - Write and execute a new query by hand.
     a) Enter your query in the text box.
b) Click **Execute**.

Edit a previously executed query.

a) In the **Previous queries** combobox, select the query to edit.
   The query appears in the text box.

b) Edit the query.

c) Click **Execute**.

Edit a query generated by the Edit table data wizard.

**Tip:** To use this method, you must have opened DB Explorer from the meta-schema diagram. See Show data on page 395.

a) Select the **Edit table data** radio button.

b) From the combobox, select the table.

**Tip:** If you were to select Execute at this time, the table data would display in Edit mode.

The SQL query to select the rows from the selected table displays.

c) Select the **Execute query** radio button.

d) Edit the query.

e) Click **Execute**.

**Execute a query**

Executing your query is the last step to viewing or modifying the data.

When you execute your query, it will either be successful or it will have a query execution error.

DB Explorer will always work with the actual database structure. You may have made changes to the tables and column definitions in the database meta-schema file (.4db), but if those changes are not made to the actual database, they are ignored by DB Explorer. Conversely, if changes are made to the underlying database structure (through the use of DDL or another database tool), the meta-schema file (.4db) must be manually updated using the **Database > Update Schema** menu option.

**Successful execution of the query**

If the query is a SELECT statement, and it is successful, the results appear in DB Explorer. The complete SQL query is written to the Output view.

If the query is an INSERT, UPDATE or DELETE statement, and it is successful, you receive no visual confirmation. The complete SQL query, however, is written to the **Output** view.

**Query execution error**

When you have a SQL error, a message displays in a pop-up window. This message provides you with the error message being returned by the database, along with a message ID.

The error message, including the complete SQL query, are also written to the Output view.

**Project Manager**

*Project Manager* is a tool to manage the organization and build of executables from the program's source files.

From Project Manager you can easily create or import projects; add and edit source files in Code Editor; add and edit forms in Form Designer; build, link, and execute programs; execute programs with the Profiler; and debug with the Graphical Debugger.
Genero project file (4pw)

A Genero Project (4pw) is an XML file that manages the source files and the properties for building and executing programs. Genero Studio displays a project file as a tree view in the Projects view.

A project includes:

- the name, type, and location of the source code and executable files
- the database meta-schema, if used
- the name, type and description of each node
- the parent-child relationships between nodes
- properties such as file paths, environment variables, and dependencies
- one or more sub-projects, if used

The project structure has no connection to the structure of the files on the disk. Files can be deleted and moved from projects without affecting the actual file on the disk.

A project remembers the last configuration used. If Genero Studio is set to use a different configuration when you open a project, you will be prompted to switch back to the previously-used configuration. Using a consistent configuration is helpful for Business Application Modeler projects that use dynamic properties, to avoid the creation of orphan properties.

Related concepts
Organizing projects on page 410
Information about organizing projects.

Related tasks
Create a new project on page 404
Create a new project to manage the project source files, libraries, and database schemas.

Structured projects

A structured project starts with a default logical structure in the Projects view and a default physical structure on disk.

The exact structure depends on the type of project. Is it a desktop or mobile project? Are you generating your application using the Business Application Modeler (BAM)?

Once the project is created, you are responsible for saving new files into the nodes and directories. This topic provides you with guidelines for using the logical and physical structures provided.

Note: These are recommendations. You can modify the structure or create your own rules regarding where you save your files. Such modifications may require you to examine and set external dependencies appropriately.
Structured projects: the logical structure in Project Manager

When you create a structured project, the **Projects** view displays a default structure of group, application, library, and packaging nodes. Figure 208: BAM Desktop Project: Project Manager perspective on page 401 shows the default **Projects** view for a new Genero BAM Desktop project, while Figure 209: BAM Mobile Project: Project Manager perspective on page 401 shows the default **Projects** view for a new Genero BAM Mobile project.

Note: For further information about the Application_GAR package, see Create a Genero package node on page 1171.
Table 121: Default nodes in a structured project on page 402 provides the guidelines as to which files should be saved to each of the Project nodes.

<table>
<thead>
<tr>
<th>Node</th>
<th>What should it contain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>The Application node is intended to hold application source files. At a minimum, it must hold the source file that contains the <code>MAIN</code> function. These files are <code>.4gl</code> files. For BAM structured projects, this node can also have the implemented program file (<code>4prg</code>) from a Business Application diagram.</td>
</tr>
<tr>
<td>Applicationflow</td>
<td>The Applicationflow node is intended for Business Application diagrams (<code>4ba</code>). This node is created for BAM projects only.</td>
</tr>
<tr>
<td>Config</td>
<td>The Config node is intended for configuration files, such as the <code>FGLPROFILE</code> file. The exception is external application configuration files for use with the Genero Application Server and the Genero Browser Client. That configuration file should be saved in the Resources node.</td>
</tr>
<tr>
<td>Database</td>
<td>The Database node contains no files when the structured project is first created. It is intended for database and database meta-schema files, such as those ending with a <code>.db</code> or <code>.4dbx</code> extension.</td>
</tr>
<tr>
<td>Entities</td>
<td>The Entities node is intended for entities, such as form files. This includes form files (<code>4fd</code>) that you created or import, as well as the form files that you implement from a Business Application diagram. For example, a CRUD form (<code>4fdm</code>) can be saved into this node.</td>
</tr>
<tr>
<td>Resources</td>
<td>The Resources node is intended for XML files that define user interface items, such as presentation style files (<code>4st</code>), toolbars (<code>4tb</code>), topmenus (<code>4tm</code>), action defaults files (<code>4ad</code>), and start menus (<code>4sm</code>). It is also intended for the XML-based external application configuration files (<code>xcf</code>).</td>
</tr>
<tr>
<td>Test</td>
<td>The Test application node contains no files when the structured project is first created. It is intended for a test scenario file (<code>4gl</code>). Dependencies for the node include the Application node of the application being tested. For more information, see Create and run a unit test on page 481.</td>
</tr>
</tbody>
</table>

Structured projects: the physical structure on disk

When you create a new structured project, you must provide the location of the project’s root directory as part of the creation process. From this root directory, a set of sub-directories are created, with each directory intended for a specific set of files.

For example, Figure 210: BAM Mobile Project: File Browser perspective on page 403 displays the directory structure on disk for a BAM mobile project created in the `MySandbox\new_BAM_mobile_project\` directory. A BAM desktop project will have a similar directory structure.

Tip: To see these directories, right-click on a file in the Project Manager and select **Locate in File Browser** or **Locate in System File Browser**.
Figure 210: BAM Mobile Project: File Browser perspective

The specifics of the directory structure can vary based on the project type. Table 122: Default directories for new structured project on page 403 outlines the content of these directories.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Files to be saved to this directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>appflow (BAM projects only)</td>
<td>.4ba diagram.</td>
</tr>
<tr>
<td>config</td>
<td>fglprofile, application configuration files (.xcf)</td>
</tr>
<tr>
<td>database</td>
<td>sqlite file, meta-schema diagram</td>
</tr>
<tr>
<td>databaseMaintenance</td>
<td>Database creation or update scripts</td>
</tr>
<tr>
<td>images</td>
<td>Image files, .ttf fonts and the mapping of these fonts (image2font.txt)</td>
</tr>
<tr>
<td>resources</td>
<td>Resource files, to include presentation style (.4st), toolbar (.4tb), topmenu (.4tm), startmenu (.4sm), and action defaults (.4ad) files.</td>
</tr>
<tr>
<td>src</td>
<td>Genero source files, to include source code (.4gl), form (.4fd, .4fdm, and so on), and text-based form (.per) files.</td>
</tr>
<tr>
<td>tests</td>
<td>GGC test scenarios (.4gl)</td>
</tr>
<tr>
<td>webcomponents</td>
<td>Web component supporting files (.wcsettings, associated image files)</td>
</tr>
</tbody>
</table>

After you build the project, new bin directories are created to hold the compiled files.

Table 123: Directories for built structured project

<table>
<thead>
<tr>
<th>Directory</th>
<th>Files to be saved to this directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin</td>
<td>Compiled executables</td>
</tr>
<tr>
<td>dbbin</td>
<td>Default target directory for database creation or update scripts.</td>
</tr>
<tr>
<td></td>
<td>This directory is created for Genero BAM Desktop projects only.</td>
</tr>
<tr>
<td>distbin</td>
<td>Genero archive files (.gar), package files, files to support OTA installation of mobile projects</td>
</tr>
<tr>
<td>testbin</td>
<td>Default target directory for GGC test scenarios</td>
</tr>
</tbody>
</table>
Managing TTF fonts

Structured projects are designed to manage the deployment of customer .ttf fonts and the mapping of these fonts. An empty image2font.txt file is written to the images directory, and the Images node has both *.ttf and image2font.txt listed in the list of files to include (set by the Include Files property).

For additional information, see the Fonts section in the Genero Business Development Language User Guide.

Related concepts

Creating new projects on page 404
Information about creating new projects.

Organizing projects on page 410
Information about organizing projects.

Quick Start: Create a project

Quickly create and test a new project. This example creates a project using Genero Studio sample source files.

1. Select File > New > Genero > Simple Project.
2. Save and name the project.
3. Expand the tree to see all default nodes in the project.
4. Right-click on the Application node and select Add Files. Navigate to the My Genero Files/samples/HelloWorld directory and select and open HelloSource.4gl and HelloSource.4fd.
5. Add a new Library node. Right-click on the Group node and select New Library. Name the library Forms. Drag the HelloForm.4fd file and drop it on top of the Forms node.
6. To identify that the application is dependent upon the files in the new Forms library, right-click on the Application node and select Advanced Properties. Notice that the Forms check box is not selected. Select the Forms check box to indicate that the application is dependent on the files in this library. Select OK.
7. Right-click the Group or Application node, and select Build to compile and link the files into an executable program. Check the output of the build in the Output view.
8. If the build is successful, right-click the Application node and select Execute to run the program through Genero Studio. The compiled and executable files for the program are stored in the Target Directory specified.

Creating new projects

Information about creating new projects.

- Create a new project on page 404
- Import existing files as a new project on page 409
- mkproject - Convert a Makefile to a project on page 409
- Connect to existing build systems on page 409

Create a new project

Create a new project to manage the project source files, libraries, and database schemas.

Start with the Genero Studio main menu to create a Project (4pw).

1. Select File > New.
2. Click on the type of project you wish to create.

Choose between a structured or simple project. Do you want to start with a predefined structure, or do you want to define your own structure? See Structured projects on page 400 for more detail about the predefined structure of a structured project.
Choose between a BAM project or a general project. BAM projects intend to use the Business Application Modeler to generate the application. See What is Business Application Modeling (BAM)? on page 223 for more information.

Choose between a Desktop or Mobile project. This decision impacts which packaging nodes are given when you create the project.

### Table 124: Choose your project type

<table>
<thead>
<tr>
<th>Category</th>
<th>Project Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genero BAM Desktop</td>
<td>BAM Desktop Project</td>
<td>Creates a structured project designed to handle generated Genero Desktop applications.</td>
</tr>
<tr>
<td>Genero BAM Desktop</td>
<td>Simple BAM Desktop Project</td>
<td>Creates a project to handle generated Genero Desktop applications. This option does not create an underlying directory structure.</td>
</tr>
<tr>
<td>Genero BAM Mobile</td>
<td>BAM Mobile Project</td>
<td>Creates a structured project designed to handle generated Genero Mobile applications.</td>
</tr>
<tr>
<td>Genero BAM Mobile</td>
<td>Simple BAM Mobile Project</td>
<td>Creates a project to handle generated Genero Mobile applications. This option does not create an underlying directory structure.</td>
</tr>
<tr>
<td>Genero</td>
<td>Desktop Project</td>
<td>Creates a structured project for desktop applications that are not code-generated.</td>
</tr>
<tr>
<td>Genero</td>
<td>Mobile Project</td>
<td>Creates a structured project for mobile apps that are not code-generated.</td>
</tr>
<tr>
<td>Genero</td>
<td>Simple Project</td>
<td>Creates a project to handle non-generated Genero applications. This option does not create an underlying directory structure.</td>
</tr>
</tbody>
</table>

3. For structured projects, specify the **Project name** and **Location** in the **Additional Information** section.

   The **Project name** becomes the name of the .4pw file. The **Location** sets the project directory ($ProjectDir). For structured projects, an initial directory structure is created for you based on the project type selected.

4. Click **OK**.

The project displays in the **Projects** view. If the project was a structured project, the initial directories and files are written to disk. If the project is not a structured project, it will not be written to disk until you save the project, at which time you will be prompted to specify the location.

### Structured projects

A structured project starts with a default logical structure in the **Projects** view and a default physical structure on disk.

The exact structure depends on the type of project. Is it a desktop or mobile project? Are you generating your application using the Business Application Modeler (BAM)?

Once the project is created, you are responsible for saving new files into the nodes and directories. This topic provides you with guidelines for using the logical and physical structures provided.

**Note:** These are recommendations. You can modify the structure or create your own rules regarding where you save your files. Such modifications may require you to examine and set external dependencies appropriately.

### Structured projects: the logical structure in Project Manager

When you create a structured project, the **Projects** view displays a default structure of group, application, library, and packaging nodes. Figure 211: BAM Desktop Project: Project Manager perspective on page 406 shows the default
Projects view for a new Genero BAM Desktop project, while Figure 212: BAM Mobile Project: Project Manager perspective on page 406 shows the default Projects view for a new Genero BAM Mobile project.

Figure 211: BAM Desktop Project: Project Manager perspective

Note: For further information about the Application_GAR package, see Create a Genero package node on page 1171.

Figure 212: BAM Mobile Project: Project Manager perspective

Table 125: Default nodes in a structured project on page 407 provides the guidelines as to which files should be saved to each of the Project nodes.
Table 125: Default nodes in a structured project

<table>
<thead>
<tr>
<th>Node</th>
<th>What should it contain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>The Application node is intended to hold application source files. At a minimum, it must hold the source file that contains the MAIN function. These files are .4gl files. For BAM structured projects, this node can also have the implemented program file (.4prg) from a Business Application diagram.</td>
</tr>
<tr>
<td>Applicationflow</td>
<td>The Applicationflow node is intended for Business Application diagrams (.4ba). This node is created for BAM projects only.</td>
</tr>
<tr>
<td>Config</td>
<td>The Config node is intended for configuration files, such as the FGLPROFILE file. The exception is external application configuration files for use with the Genero Application Server and the Genero Browser Client. That configuration file should be saved in the Resources node.</td>
</tr>
<tr>
<td>Database</td>
<td>The Database node contains no files when the structured project is first created. It is intended for database and database meta-schema files, such as those ending with a .db or .4dbx extension.</td>
</tr>
<tr>
<td>Entities</td>
<td>The Entities node is intended for entities, such as form files. This includes form files (.4fd) that you created or import, as well as the form files that you implement from a Business Application diagram. For example, a CRUD form (.4fdm) can be saved into this node.</td>
</tr>
<tr>
<td>Resources</td>
<td>The Resources node is intended for XML files that define user interface items, such as presentation style files (.4st), toolbars (.4tb), topmenus (.4tm), action defaults files (.4ad), and start menus (.4sm). It is also intended for the XML-based external application configuration files (.xcf).</td>
</tr>
<tr>
<td>Test</td>
<td>The Test application node contains no files when the structured project is first created. It is intended for a test scenario file (.4gl). Dependencies for the node include the Application node of the application being tested. For more information, see Create and run a unit test on page 481.</td>
</tr>
</tbody>
</table>

Structured projects: the physical structure on disk

When you create a new structured project, you must provide the location of the project's root directory as part of the creation process. From this root directory, a set of sub-directories are created, with each directory intended for a specific set of files.

For example, Figure 213: BAM Mobile Project: File Browser perspective on page 408 displays the directory structure on disk for a BAM mobile project created in the MySandbox\new_BAM_mobile_project\ directory. A BAM desktop project will have a similar directory structure.

Tip: To see these directories, right-click on a file in the Project Manager and select Locate in File Browser or Locate in System File Browser.
Figure 213: BAM Mobile Project: File Browser perspective

The specifics of the directory structure can vary based on the project type. Table 126: Default directories for new structured project on page 408 outlines the content of these directories.

Table 126: Default directories for new structured project

<table>
<thead>
<tr>
<th>Directory</th>
<th>Files to be saved to this directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>appflow (BAM projects only)</td>
<td>.4ba diagram.</td>
</tr>
<tr>
<td>config</td>
<td>fglprofile, application configuration files (.xcf)</td>
</tr>
<tr>
<td>database</td>
<td>sqlite file, meta-schema diagram</td>
</tr>
<tr>
<td>databaseMaintenance</td>
<td>Database creation or update scripts</td>
</tr>
<tr>
<td>images</td>
<td>Image files, .ttf fonts and the mapping of these fonts (image2font.txt)</td>
</tr>
<tr>
<td>resources</td>
<td>Resource files, to include presentation style (.4st), toolbar (.4tb), topmenu (.4tm), startmenu (.4sm), and action defaults (.4ad) files.</td>
</tr>
<tr>
<td>src</td>
<td>Genero source files, to include source code (.4gl), form (.4fd, .4fdm, and so on), and text-based form (.per) files.</td>
</tr>
<tr>
<td>tests</td>
<td>GGC test scenarios (.4gl)</td>
</tr>
<tr>
<td>webcomponents</td>
<td>Web component supporting files (.wcsettings, associated image files)</td>
</tr>
</tbody>
</table>

After you build the project, new bin directories are created to hold the compiled files.

Table 127: Directories for built structured project

<table>
<thead>
<tr>
<th>Directory</th>
<th>Files to be saved to this directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin</td>
<td>Compiled executables</td>
</tr>
<tr>
<td>dbbin</td>
<td>Default target directory for database creation or update scripts.</td>
</tr>
<tr>
<td></td>
<td>This directory is created for Genero BAM Desktop projects only.</td>
</tr>
<tr>
<td>distbin</td>
<td>Genero archive files (.gar), package files, files to support OTA installation of mobile projects</td>
</tr>
<tr>
<td>testbin</td>
<td>Default target directory for GGC test scenarios</td>
</tr>
</tbody>
</table>
Managing TTF fonts

Structured projects are designed to manage the deployment of customer .ttf fonts and the mapping of these fonts. An empty image2font.txt file is written to the images directory, and the Images node has both *.ttf and image2font.txt listed in the list of files to include (set by the Include Files property).

For additional information, see the Fonts section in the Genero Business Development Language User Guide.

Related concepts
Creating new projects on page 404
Information about creating new projects.
Organizing projects on page 410
Information about organizing projects.

Import existing files as a new project

Files in a file system can be imported as a new project using the Project > Import menu option.

Before you begin, Set up a remote environment if your source files are on a remote host.

1. If the source files to be imported contain reference fields from a database table, create or import a meta-schema file (4db) for the database, if not already completed.

2. Create a new, simple project to manage the files using Project Manager.

3. Use the Import Project from the Projects menu to import the directory as a new project. To make sure that only source files appear in the project, either clean all binaries and generated files from the directory before importing or set the import filters in the Import Project dialog to include/exclude files of various types.

   Genero Studio will organize the files into the project. Each 4gl file with a MAIN will reside in an Application node. All other files will be located in Library nodes. You can organize the files how you wish.

4. If sources contain files that are preprocessed to generate other files, set up the appropriate build rules. (For example, if a file with an extension of 5gl generates a 4gl and then builds it: Add a build rule 5gl # 42m.)

5. Once the source files are imported into the project, if the Compute dependencies checkbox is selected, the import process will automatically fill the required information for any libraries that must be linked in the same project. Right-click the Application node and select Advanced Properties, Dependencies to check the appropriate dependencies for each application or library.

6. Add the meta-schema file to the Databases node of your project.

7. Set the other properties for the nodes in the project, as needed. For example:
   a) To change the default location of the output files when the program is built, set the Target Directory property for the Group or Application nodes. The default target directory is $ProjectDir/bin.
   b) If there are external libraries (outside the Project) that should be linked when the application is built, add the library to the value of the external dependencies property for the application node.

8. Build the Project to compile and link the files, checking for any dependency errors.

mkproject - Convert a Makefile to a project

mkproject can be used to automatically create 4pw project files from the older build system make.

See GSTDIR/tools/mkproject/readme.txt on how to use mkproject.

Connect to existing build systems

To connect to an existing build system, edit the build rules or link rules to execute the building system commands.

For example, if the existing build system is make, you can call make in the link rule to create a library.

Related concepts
What are build rules on page 413
Organizing projects

Information about organizing projects.

- Groups, Applications, Libraries, and Packages on page 410
- Using external libraries on page 411
- Setting external dependencies on page 411

For information about source code management for your project, see Source Code Management (SCM) on page 624.

Groups, Applications, Libraries, and Packages

Projects are organized into Group, Application, Library, and Package nodes. The Projects view visually displays a project file (4pw) for easy management of project source files.

Group

Group nodes organize the Application and Library nodes that make the project. Define default properties by setting them at the group level (TargetDir, Language, Compiler options, etc.) Properties defined for the Group node are inherited by all child nodes. To organize the files, you can add virtual folders under a Group node.

Note: Virtual folders exist only in the logical view in Project Manager; they do not exist in the physical file structure.

Application

An Application node is used to generate an executable program (42x). Application nodes can contain both files and virtual folders.

Only one of the files in an Application node may have a MAIN statement; a single Application node creates a single executable. The name of the Application node is used as the name of the 42x file, so it must be unique and can only contain characters allowed by the file system.

The default application is shown in boldface. Use the Projects view integrated Toolbar to set a different application as the default. The options on the Build menu execute for the default application.

Library

A library node is used to group binary files into a single library and generate a library file (42x). Library nodes can contain both files and virtual folders. The name of the library node is used as the name of the 42x file, so it must be unique and can only contain characters allowed by the file system.

Libraries should be used when creating a set of features having a common goal, such as the logic of an application or a library of mathematical functions. A library can also be used to group other project files together (images, styles, or other resources).

If a library node contains no 4gl file, no 42x is built.

A library from a different project can be added to a project using the context menu option Add External Project.

Important: A library must be linked to any application in which it will be used. Right-click the application node, select Advanced Properties, dependencies, and select the checkbox for any required library.

Packages

A package node is used to package an application for deployment to a mobile device or for distribution to users. You can add Directory nodes under a Package node to organize the files. See Packaging, deploying, and distributing apps on page 1168.
Related concepts

Projects view on page 119
The Projects view displays a project and its components.

Structured projects on page 400
A structured project starts with a default logical structure in the Projects view and a default physical structure on disk.

Using external libraries

Libraries defined in one project can be used in other projects.

To create a link to external libraries, right-click on a Group node in the projects view and select Add External Project. A new node is created with the name of the external 4pw file.

When an external project is added, all of its projects, application and libraries are automatically added as children. The child nodes are in grey because they are in read-only mode; they cannot be moved, renamed, and so on. Their properties are also read-only.

It is important to set the dependencies for any application or standard library node that requires the use of this library, using Advanced Properties. The new node, with all its libraries as children, will be shown in the Dependencies property page; check each library that should be linked with the application or standard library.

Setting external dependencies

External dependencies are files that are not a part of the project, but should be included in the linking process.

External libraries are mainly used when a library is used in many projects and is not intended to be modified, or if third party libraries are used for which the sources are not available.

Select the Application or Library node and in the Project view, set the External dependencies property to the list of files that should be included in the linking process.

The directories containing these files must be added to the FGLLDPATH environment variable. See Genero variables.

Application configuration files and Project Manager

Genero Studio creates a default application configuration file to display applications to the Web client. You can update the default configuration with a custom application configuration file.

By default, when Genero Studio executes an application for display in the Web client, it generates a default application configuration file for that application, and places it in a directory specified by the Genero Application Server configuration. For information on the Genero Application Server configuration within Genero Studio, see Configure for the Web client on page 171.

You may need to modify this default configuration. To achieve this, add a custom application configuration (.xcf) file to the project. This custom file can have any valid file name, but must end with an .xcf extension. Add the file to an application node or dependent library.

![Figure 214: Project View with Custom XCF file displayed](image)

In the custom file, only add those elements that you need to modify.
How Genero Studio processes your configuration file

When you launch an application from a project, Genero Studio searches for a custom application configuration file in the following order:

1. In the application node.
2. In the libraries that are dependencies of the application. If multiple custom .xcf files are located, the first found file is used.

The details in the application configuration file are applied to the configuration file generated by Genero Studio. Two .xcf files are generated when the application is run with the GBC client:

- **application_user_studio.xcf**: This file is temporary and only used by Genero Studio for running the Web application for a single session. This configuration cannot be used to run the application directly from the browser.
- **application_user.xcf**: This file is intended to run the application directly from the browser during development.

**Warning**: Genero Studio adds environment variables, resources, and more to the application configuration file (.xcf) to ensure the application runs as defined in the development environment.

When creating the deployment application configuration (.xcf) file to include in the Studio Project for packaging, you should create a new file using File > New... > Web / AS > Application Configuration (.xcf), and only the necessary environment variables and resources should be present.

For example, resources such as res.path.webcomponent.user should not be present in the deployment file.

**Related tasks**

- Create an application configuration file on page 1169

An application or Web service configuration file provides details required by the Genero Application Server to launch an application or Web service. If you created your project using the Desktop Project (.4pw) or BAM Desktop Project options, you already have this file. However, if this is not the case, you can create this file in Genero Studio.

Building and linking programs

Build rules compile each file in a project. Link rules create the applications and libraries. Execution rules execute the application. Build rules, link rules, and execution rules together make up a Language in Genero Studio.

**Warning**: A rebuild of a project may not automatically be done when files within the project are updated. It is the responsibility of the developer to recompile the appropriate parts of the project.

- What are build rules on page 413
- Add/Edit a build rule on page 414
- Example: How build rules work on page 415
- Languages on page 413
- Link rules on page 416
- Execution rules on page 416
- Command line options for build, link, execution rules on page 416
- Environment variables on page 418
- Pre/Post compile on page 420
- Pre/Post link on page 421
- The gsmake command on page 421
Languages

A language is a named set of build rules (to compile files), link rules (to link application or library nodes), execution rules (to execute/debug/profile an application) and a set of environment variables.

Genero Studio comes preconfigured for some languages. You can add your own languages.

You select which language to use by setting the Language property on an application, group or library node.

Related concepts

Build Rules Configuration dialog (Languages) on page 427

The Build Rules Configuration dialog is used to edit a language (add, modify, delete build/link/execution rules and variables) and to add a new language or remove an existing language.

What are build rules

Build rules are used to compile each file in a project.

Global and Project build rules can be modified. New build rules can be created.

Build rules are grouped into the following hierarchy:

- Project - If relevant build rule found under Project, it is used.
- Specific - If no relevant build rule is found under Project, then relevant build rule under Specific is used.
- Global - if no relevant build rule is found under Project or Specific, then relevant build rule under Global is used.
- Default - if no relevant build rule is found under Project, Specific, or Global, then relevant Default build rule is used.

Only one build rule can be active for a specific file type.

Access the Build Rules dialog using Tools > Global setup > Edit Build Rules, Tools > Specific setup > Edit Build Rules or right-click on the Project view and select Edit Build Rules.

Related concepts

Build Rules Configuration dialog (Languages) on page 427
The **Build Rules Configuration** dialog is used to edit a language (add, modify, delete build/link/execution rules and variables) and to add a new language or remove an existing language.

**Add/Edit a build rule**

From the Build rules dialog you can add or edit a build rule.

![Add/Edit Build Rule dialog](image)

**Figure 215: Add/Edit Build Rule dialog**

<table>
<thead>
<tr>
<th>Label</th>
<th>STR_compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>File type</td>
<td>application/genero-str</td>
</tr>
<tr>
<td>Command lines</td>
<td></td>
</tr>
</tbody>
</table>
| $(fglmkstr) "$(InputPath)"
| $(move) "$(InputDir)/$(InputBaseName).42s" "$\{TargetDir\}/$(InputBaseName).42s"

**Output files**

| $(TargetDir)/$(InputBaseName).42s |

**Intermediate files**

**Additional dependencies**

**Label**

Enter a label to be assigned to the rule.

**File type**

Defines the MIME-type of the source files that can be compiled with this Build rule; click the icon on the right to display the File Type Selection dialog. Select the file type from a list generated from the **File Associations** in **Genero Studio Preferences**. Use the Search field to limit the display of the list; enter the file extension, or the words Genero or studio, for example, to display only the corresponding file types. The icons can be used to display the list by category, and to expand or contract a category.
Output files
Enter the list of files that are generated by the Build rule. All the predefined file node variables can be used when specifying the path of the file.

Note: When entering file paths, do not use quotations surrounding the file path. That is only necessary in the Command line field.

Intermediate files
Indicates which files are generated during compilation, as intermediate files, before generating the output file. This displays the Intermediate files in the Project view.

Note: When entering file paths, do not use quotations surrounding the file path. That is only necessary in the Command line field.

Additional dependencies
The additional dependencies is a list of files used to verify that the build is up to date. If one of these files has a modified date more recent than any of the output files, the build is considered not up to date and the build rule will be executed.

Note: When entering file paths, do not use quotations surrounding the file path. That is only necessary in the Command line field.

Example
For a preprocessor named mypp for source files (*.my), the Build Rule for the mime type "text/my" would be:

```bash
mypp $(InputPath) -o $(TargetDir)/$(InputBaseName).4gl
$(fglcomp) $(TargetDir)/$(InputBaseName).4gl
$(delete) $(TargetDir)/$(InputBaseName).4gl
```

Redirect output
When a command is executed in a build, link, execution rule or in a user action, the output (standard or error) is displayed in the Output view. You can redirect the output to a file. For example, if you write `echo "Hello World"` in a build rule, Hello World is displayed in the Output view. To redirect the output to a file, use standard Linux™ syntax such as: `echo "Hello World" > "c:\file.txt"`. Supported syntax is `>`, `1>`, `>>`, `2>`, `1>&2`, `2>&1`.

Related concepts
Build Rules Configuration dialog (Languages) on page 427
The Build Rules Configuration dialog is used to edit a language (add, modify, delete build/link/execution rules and variables) and to add a new language or remove an existing language.

Example: How build rules work

Example: myfile.4gl
The process used by Project Manager to compile a file called myfile.4gl is:

1. Project Manager first searches the language to be used as defined in the parent application or library node.
2. Project Manager determines the MIME type of this file; it is "application/genero-4gl".
3. Then, Project Manager looks for a corresponding build rule in the Project build rules. After searching the Project build rules, if no rule is found that handles the "application/genero-4gl" MIME type, it searches the Template build rules, and then the Global build rules and then the Default build rules.
4. Finally, Project Manager executes the commands defined in the build rules after having replaced the variables.
5. If no build rule is found, the file is skipped and the next one is processed.

**Related concepts**

*Build Rules Configuration dialog (Languages)* on page 427

The *Build Rules Configuration* dialog is used to edit a language (add, modify, delete build/link/execution rules and variables) and to add a new language or remove an existing language.

**Link rules**

Link rules create the applications and libraries.

A link rule executes when you build an application or library node.

**Related concepts**

*Build Rules Configuration dialog (Languages)* on page 427

The *Build Rules Configuration* dialog is used to edit a language (add, modify, delete build/link/execution rules and variables) and to add a new language or remove an existing language.

**Execution rules**

An execution rule is executed when you run, debug, or profile an application node.

**Related concepts**

*Build Rules Configuration dialog (Languages)* on page 427

The *Build Rules Configuration* dialog is used to edit a language (add, modify, delete build/link/execution rules and variables) and to add a new language or remove an existing language.

**Command line options for build, link, execution rules**

Special command line options can be used for build, link, and execution rules.

**Table 128: Command line operands**

<table>
<thead>
<tr>
<th>Command line / syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(4dbcomp)</td>
<td>Builds the schema file (<em>sch</em>) from the database file (<em>4db</em>).</td>
</tr>
<tr>
<td>$(4dbcomp) sourceFile[4db]</td>
<td>Builds the schema file (<em>sch</em>) from the database file (<em>4db</em>).</td>
</tr>
<tr>
<td>$(4fdcomp)</td>
<td>Builds the compiled form file (<em>42f</em>) from the form file (<em>4fd</em>).</td>
</tr>
<tr>
<td>$(4fdcomp) [options] sourceFile[4fd]</td>
<td>Builds the compiled form file (<em>42f</em>) from the form file (<em>4fd</em>).</td>
</tr>
<tr>
<td>$(blockpoint)</td>
<td>The <em>$(blockpoint)</em> command manages user added code by extracting or injecting code between BLOCK and POINT tags in a generated <em>4gl</em> file.</td>
</tr>
<tr>
<td>$(blockpoint) [options] filename</td>
<td>The <em>$(blockpoint)</em> command manages user added code by extracting or injecting code between BLOCK and POINT tags in a generated <em>4gl</em> file.</td>
</tr>
<tr>
<td>$(copy)</td>
<td>Copies the given file or directory to the given destination in a platform independent way.</td>
</tr>
<tr>
<td>$(copy) sourceFilePath destinationFilePath</td>
<td>Copies the given file or directory to the given destination in a platform independent way.</td>
</tr>
<tr>
<td>Command line / syntax</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>$(delete) filePath1 filePath2 ...</td>
<td>Removes the given files or directories in a platform independent way.</td>
</tr>
<tr>
<td>$(fglcomp) [options] sourceFile[4gl]</td>
<td>The fglcomp tool compiles BDL program sources files into a p-code version.</td>
</tr>
<tr>
<td>$(fglmkmsg) [options] sourceFile[.msg] outFile[.iem]</td>
<td>The fglmkmsg tool compiles message files into a binary version used by the BDL programs.</td>
</tr>
<tr>
<td>$(fglmkstr) sourceFile[.str]</td>
<td>The fglmkstr tool compiles localized string files.</td>
</tr>
<tr>
<td>$(fglrun) [options] program</td>
<td>Calls fglrun tool, the runtime system program that executes p-code programs.</td>
</tr>
<tr>
<td>$(fglwsdl) [options] &lt;filename</td>
<td>Calls the fglwsdl tool for creating a web services program.</td>
</tr>
<tr>
<td>$(generate) [options] filename</td>
<td>The $(generate) command creates an intermediary XML file from modeled entities.</td>
</tr>
<tr>
<td>$(gstdebug) filename</td>
<td>Calls internal Genero Studio debugger. Execution rules only.</td>
</tr>
<tr>
<td>$(gstrun) program</td>
<td>Calls command to run an application through Genero Studio. Execution rules only.</td>
</tr>
<tr>
<td>$(move) sourceFilePath destinationFilePath</td>
<td>Moves the given file or directory to the given destination in a platform independent way.</td>
</tr>
<tr>
<td>$(percomp) [options] sourceFile[per]</td>
<td>The fglform tool compiles form specification files into XML formatted files used by the programs.</td>
</tr>
<tr>
<td>$(tcl) - deprecated</td>
<td>The tclsh executable generates the final file by using both a Tcl template file and the intermediary XML file created by the $(generate) command.</td>
</tr>
</tbody>
</table>
Related concepts

Predefined node variables on page 436
Each Project Manager node has a defined set of variables containing values determined at runtime.

Environment variables

Information about environment variables.

- Add or edit environment variables on page 168
- What determines the value of an environment variable on page 419

Add or edit environment variables

The Environment Variable dialog is used to add and edit environment variables.

When the Environment Variable dialog appears, enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>The type of environment variable. Options are Value, Value List, Directory, Directory List, File, or File List.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the environment variable.</td>
</tr>
<tr>
<td>Value</td>
<td>The value of the environment variable. When entering the value, if the type is Value List, Directory, Directory List, File, or File List, select the ellipses (...) to browse for the correct value. If the value contains a variable name, that name must be prefaced with $ and enclosed in parenthesis; for example $ (FGLLDPATH) . The list separator is always a semicolon (;) on all systems (Windows® and UNIX™). The directory separator in a path is always a slash (/) on all systems. <strong>Tip:</strong> Use the semicolon to separate directories in a list, and the slash (/) as the separator in a path, for portability of projects across operating systems.</td>
</tr>
</tbody>
</table>

Figure 216: Setting FGLLDPATH
Reusing existing environment variables

A variable defined for a parent or ancestor node can also be reused in definitions for a child node:

For example:

- The parent node defines: `MY_VALUE=hello`
- The child node can reuse the parent node variable: `MY_COMPLETE_VALUE=$(MY_VALUE) world`
- The final value of `MY_COMPLETE_VALUE` is "hello world".

For example:

- The parent node defines: `MY_VALUE=foo`
- The child node can reuse the parent node value and redefine the variable: `MY_VALUE=$(MY_VALUE) bar`
- The final value of `MY_VALUE` will be "foo bar".

As a result, the System environment variables or Genero Studio Configuration variables can be reused in User Variable definitions within Project Manager.

What determines the value of an environment variable

Environment variables define the environment used by the compiler and executables launched from Project Manager. Where and when the environment variable is important.

Order of precedence

An environment variable can be set in multiple places, inside and outside of Genero Studio. Environment variables have an order of priority (high to low):

1. The environment defined on the node itself (Genero variables and User variables). This has the highest priority.
2. The environment of the node's parents (ancestors).
3. The environment of the node's dependencies.
4. Environment variables defined in the current language.
5. Environment variables defined in Environment sets on page 164.
6. System environment variables. This has the lowest priority.

The order of priority allows you to override both system environment variables (6, above) or environment variables defined in a Genero Studio configuration (5, above) by setting them in Project Manager (1 through 4, above).

Order within an Environment Set

Within an Environment Set, Arrow keys can be used to change the order of the variables defined. This affects the way the variables are interpreted when the program is executed. For example:
Table 129: Variable interpretation examples

<table>
<thead>
<tr>
<th>Project level</th>
<th>Variables definition and order</th>
<th>Result when application is executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>V1=&quot;hello&quot;</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>V2=&quot;$(V1) world&quot;</td>
<td>V1=&quot;goodbye&quot;</td>
</tr>
<tr>
<td></td>
<td>V1=&quot;goodbye&quot;</td>
<td>V2=&quot;hello world&quot;</td>
</tr>
<tr>
<td>Application</td>
<td>V1=&quot;goodbye&quot;</td>
<td>V1=&quot;goodbye&quot;</td>
</tr>
<tr>
<td></td>
<td>V2=&quot;$(V1) world&quot;</td>
<td>V2=&quot;goodbye world&quot;</td>
</tr>
</tbody>
</table>

Note: If a foreign language Language support (text encoding) on page 189 is selected in Genero Studio preferences, the LANG variable must be set appropriately to correspond to the selected encoding. Set the variable for a specific environment set in Genero Studio configurations (Tools>>Configurations).

Example: Priority and Environment Variables

In this example, the program node in the project has a library node as a dependency. The environments have been defined as shown:

Table 130: Environment Set for levels used in the example

<table>
<thead>
<tr>
<th>Level</th>
<th>Environment Set</th>
</tr>
</thead>
</table>
| System environment      | PATH=/bin;/usr/bin
                         | VAR1=hello
                         | VAR2=bonjour
                         | VAR3=guten tag
| My Library Environment  | PATH=$(PATH);$(ProjectDir)/scripts
                         | VAR1=goodbye
| My Program Environment  | PATH=$(PATH);$(ProjectDir)/scripts
                         | VAR1=$(VAR1) world
                         | VAR2=$(VAR2) $(VAR1)

If the project directory $(ProjectDir) is set to "/home/joe/project", the environment for the program will be:

PATH=/bin;/usr/bin;/home/joe/project/scripts
VAR1=goodbye world
VAR2=bonjour goodbye world
VAR3=guten tag

Pre/Post compile

Pre-compile commands are executed just before the compilation of a file, while post-compile commands are executed just after the compilation of a file.

To set pre- or post-compile commands, right-click on a file in the Projects view and select Advanced Properties. The Pre/Post Compile command dialog opens.
Pre/Post link

Pre-link commands are executed just before the link of the application or library node, while post-link commands are executed just after the link of the application or library node.

To set pre- or post-link commands, right-click on the application or library node in the Projects view and select Advanced Properties. The Pre/Post Compile command dialog opens, select the Pre/Post Link command option in the Pages listing.

The gmake command

The gmake command is a command line option to build projects. The tool is located in the $GSTDIR/bin directory.

Syntax

gmake [options] <file_list>

where $file_list$ is a list of target Project (.4pw) files, with or without the extension.

Table 131: gmake - Help and version information

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td>Display help information.</td>
</tr>
<tr>
<td>-V</td>
<td>Display this program name and version.</td>
</tr>
</tbody>
</table>

Table 132: gmake - Targets

<table>
<thead>
<tr>
<th>Targets</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-active</td>
<td></td>
<td>Target the default application, as set in the project.</td>
</tr>
<tr>
<td>-all</td>
<td></td>
<td>Target the complete project (default behavior).</td>
</tr>
<tr>
<td>-t</td>
<td>$TARGET$</td>
<td>Add $TARGET$ to the list of targets to build. This argument can be used multiple times to build several targets.</td>
</tr>
</tbody>
</table>

Table 133: gmake - Operations (exclusive)

<table>
<thead>
<tr>
<th>Operations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-b</td>
<td>Build the target (default behavior). The files that are not up-to-date are compiled, the others are not changed.</td>
</tr>
<tr>
<td>-r</td>
<td>Rebuild the target. The output files are deleted, then all files are compiled.</td>
</tr>
<tr>
<td>-c</td>
<td>Clean the target. The output files are deleted.</td>
</tr>
<tr>
<td>-convert</td>
<td>Convert the project to the latest format.</td>
</tr>
<tr>
<td>-force-build</td>
<td>Force the build/rebuild of the target. The files are compiled, whether or not they are up-to-date.</td>
</tr>
</tbody>
</table>
### Table 134: gsmake - Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-j</code></td>
<td><code>NB</code></td>
<td>Set the number of parallel jobs to <code>NB</code> (default: 1). Set <code>NB</code> to 0 to use the local computer's number of CPUs. When using this option, gsmake will try to start multiple compilations in parallel when possible. This should speed up the global compilation time.</td>
</tr>
<tr>
<td><code>-encoding</code></td>
<td><code>ENCODING</code></td>
<td>Set the encoding to <code>ENCODING</code>. (default: System encoding)</td>
</tr>
<tr>
<td><code>-max-errors</code></td>
<td><code>NB</code></td>
<td>Set the maximum number of erroneous files to <code>NB</code>. Set to 0 (zero) for an unlimited number of error files. Default is 5.</td>
</tr>
<tr>
<td><code>-disable-dependencies</code></td>
<td></td>
<td>Disable the computation of the dependencies database. (default: false)</td>
</tr>
<tr>
<td><code>-ag-templateDir</code></td>
<td><code>TEMPLATE</code></td>
<td>Set the Application Generator template directory to <code>TEMPLATE</code>. Enter an absolute path, or a path relative to the <code>GSTDIR/bin/src/ag/tpl</code> directory. The default value (&quot;default&quot;) corresponds to the multiple-dialog template directory.</td>
</tr>
<tr>
<td><code>-generate-4pwdb</code></td>
<td></td>
<td>Generate the 4pwdb file only. Default is False.</td>
</tr>
<tr>
<td><code>-wcDir</code></td>
<td><code>WEBCOMPONENTS_DIR</code></td>
<td>WebComponents Directory path</td>
</tr>
<tr>
<td><code>-verbose</code></td>
<td></td>
<td>Specify verbose mode for build / link / execution rules.</td>
</tr>
<tr>
<td><code>-port</code></td>
<td><code>PORT</code></td>
<td>Set the port number.</td>
</tr>
<tr>
<td><code>-javaServerPort</code></td>
<td><code>JAVAPORT</code></td>
<td>Set the Java Server port number.</td>
</tr>
<tr>
<td><code>-javaServerLog</code></td>
<td><code>LOGLEVEL</code></td>
<td>Set the Java Server log level.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINE</td>
<td>Turn on all logs.</td>
</tr>
<tr>
<td>INFO</td>
<td>Turn on log level to info message.</td>
</tr>
<tr>
<td>WARNING</td>
<td>Turn on log level to warnings.</td>
</tr>
<tr>
<td>SEVERE</td>
<td>Turn on log level to severe errors (default).</td>
</tr>
<tr>
<td>OFF</td>
<td>Turn off logs.</td>
</tr>
</tbody>
</table>

## Packaging

Package nodes are used to package an app for deployment to a mobile device or for distribution to users.

See [Packaging, deploying, and distributing apps](#) on page 1168.
Locate a file (starting at Project Manager)

From the Projects view, you can locate the file in the File Browser, in the System File Browser, or in a BA diagram.

Before you begin, the Projects view is open.

You have a file visible in the Projects view that you wish to locate in an alternate view, diagram, or in the System File Browser.

1. Right-click on the file.
   The contextual menu displays.

2. Select the appropriate menu option:
   - To open in File Browser, select Locate in File Browser.
   - To open in the System File Browser, select Locate in System File Browser.
   - To open in the BA diagram, select Locate in BA Diagram. This option is only available for files created using the Business Application Modeler (BAM).

The desired view, diagram, or dialog opens in the selected option, showing the location of the file.

Project Manager Reference

Reference information for Project Manager.

- Project Manager context menu on page 423
- Dialogs on page 424
- Views on page 466
- Predefined node variables on page 436

Project Manager context menu

Select a node in the project and right-click to display a menu of context relevant actions. Select multiple nodes using Ctrl-click.

Table 135: Project Manager Context Menu

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Group</td>
<td>Create a new node in the project .</td>
</tr>
<tr>
<td>Import Project</td>
<td>Import one or more existing files to project . See Import existing files as a new project on page 409</td>
</tr>
<tr>
<td>Compute Dependencies</td>
<td>Remove and recompute all the dependencies between all applications and libraries.</td>
</tr>
<tr>
<td>Build</td>
<td>Build and link the selected application.</td>
</tr>
<tr>
<td>Rebuild</td>
<td>Rebuild and link the selected application.</td>
</tr>
<tr>
<td>Clean</td>
<td>Clean the selected application.</td>
</tr>
<tr>
<td>Open Dependency Diagram</td>
<td>Opens Dependency Diagram</td>
</tr>
<tr>
<td>Cut/Copy/Paste</td>
<td>Cut, copy, or paste from clipboard.</td>
</tr>
<tr>
<td>Rename</td>
<td>Rename node.</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete selected item.</td>
</tr>
<tr>
<td>Delete from Disk</td>
<td>Delete file from disk.</td>
</tr>
<tr>
<td>Menu Option</td>
<td>Usage</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SCM</td>
<td>If a file is under <strong>Version Control</strong>, additional options from the context menu are available to commit, update and revert. See <strong>Source Code Management (SCM)</strong> on page 624.</td>
</tr>
<tr>
<td>New Group/Application/Library/Virtual Folder/File</td>
<td>Create a new node in the project.</td>
</tr>
<tr>
<td>Add External Project</td>
<td>Add a library from a different project to the current project. See <strong>Using external libraries</strong> on page 411.</td>
</tr>
<tr>
<td>Edit Build Rules</td>
<td>See <strong>What are build rules</strong> on page 413</td>
</tr>
<tr>
<td>Display Environment</td>
<td>Displays the values of system environment and current Genero Studio configuration environment variables and of the local node variables used by Project Manager.</td>
</tr>
<tr>
<td>Advanced Properties</td>
<td>Set dependencies between applications and libraries, pre/post-link commands and environment variables. See <strong>Advanced Properties dialog</strong> on page 425. The Dependency property <strong>always</strong> must be set if the project consists of both application and library nodes.</td>
</tr>
<tr>
<td>Set as Default Application</td>
<td>Set the selected application as the default. Only one application can be the default at one time. By default, the first application created is the default application. The default application node is boldfaced in the project tree. The options on the <strong>Build</strong> menu execute for the default application.</td>
</tr>
<tr>
<td>New File</td>
<td>Create a new file to add to an Application or Library node. See <strong>The File New menu</strong> on page 98</td>
</tr>
<tr>
<td>Add Files</td>
<td>Locate in file system and add existing files to an application or library node. This adds a link to the given file in the project; it does not physically move the files. All the files must be located on the same drive.</td>
</tr>
<tr>
<td>Open Dependency Diagram</td>
<td>See <strong>Dependency Diagrams</strong> on page 484</td>
</tr>
<tr>
<td>Add Web Service</td>
<td>See <strong>Add Web Service</strong> on page 1076</td>
</tr>
<tr>
<td>Execute</td>
<td>Execute selected application. If multiple applications are selected, they will be run sequentially.</td>
</tr>
<tr>
<td>Execute with Profiler</td>
<td>Execute selected application with Profiler. See <strong>Profiler</strong> on page 596</td>
</tr>
<tr>
<td>Debug</td>
<td>Launch the Debugger for selected application. Multiple selections cannot be debugged.</td>
</tr>
<tr>
<td>Open</td>
<td>Open the selected file in Code Editor or Form Designer, depending on the file type.</td>
</tr>
<tr>
<td>Locate in File Browser</td>
<td>Locate file in File Browser.</td>
</tr>
<tr>
<td>Compile File</td>
<td>Compile the selected file.</td>
</tr>
</tbody>
</table>

**Dialogs**

Information about Project Manager dialogs.

- **Advanced Properties dialog** on page 425
- **Import Project dialog** on page 426
- **Build Rules Configuration dialog (Languages)** on page 427
Advanced Properties dialog

The Advanced Properties dialog provides access to setting Dependencies, Pre/Post link commands, and Environment variables for Group, Library, Application, or File nodes in the project.

To access the Advanced Properties dialog right-click on a selected Group, Library, Application, or File node and select Advanced Properties.

![Advanced Properties dialog](image)

**Figure 218: Advanced Properties dialog**

**Table 136: Advanced Properties Pages**

<table>
<thead>
<tr>
<th>Page</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependencies (for Application, Library nodes)</td>
<td>A list of the available libraries and applications from the current project is displayed. Select the check box for any library or application to include in the build.</td>
</tr>
<tr>
<td>Pre/Post link command (for Application, Library nodes)</td>
<td>Shell scripts or other programs to be executed before or after linking. Enter the complete path of the script or program, if necessary. For example, if the script <code>myscript</code> is in <code>/home/user/scripts</code>, and this directory is included in the PATH environment variable of the system, enter <code>myscript</code>. Otherwise, enter <code>/home/user/scripts/myscript</code>.</td>
</tr>
<tr>
<td>Page</td>
<td>Usage</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>Pre/post compile command (for File nodes)</td>
<td>Shell scripts or other programs to be executed before/after compiling. Provide the complete path to the script or program if the path is not part of the PATH environment variable of the system.</td>
</tr>
<tr>
<td>Environment variables (for Group, Application, Library nodes)</td>
<td>Existing variables can be defined or redefined. The Genero Variables list displays a list of environment variable settings that are automatically computed by Genero Studio. These variables cannot be altered, although they can be redefined in Project variables. For example, if there are several applications that use different FGLPROFILE files, set a specific FGLPROFILE environment variable for each one. Predefined node variables can be used in the value.</td>
</tr>
</tbody>
</table>

**Related concepts**

- Add or edit environment variables on page 168
- The Environment Variable dialog is used to add and edit environment variables.
- Predefined node variables on page 436
  Each Project Manager node has a defined set of variables containing values determined at runtime.

**Import Project dialog**

Use the Import Project dialog to specify preferences when importing files from a file system into a project.

![Import project dialog](Figure 219: Import Project dialog)

**Project path**

Enter the path and name of the directory to be imported, or use the browse button to select it from the file system.

**Include files**

A complete list of the files that will be imported (for example, *.4gl indicates all files with the extension 4gl). By default, files having the Genero and Genero Studio file extensions are listed; modify this list by adding or removing entries. The entries in the lists are separated by spaces.
Define the source directory as target directory

If checked, The Target directory property is set to the relative path from $(ProjectDir) to the source directory.

Exclude files

Enter the files that are to be excluded from the project. Use the * symbol to indicate all files; for example, *.abc would exclude all files having the extension abc. The entries are separated by spaces.

Exclude directories

A list of excluded directories. By default the CVS directory is excluded. The entries are separated by spaces.

Compute dependencies

If checked, Project Manager will try to compute the dependencies between Application/Libraries when importing the project.

Example

In this example, all files with a 4gl extension in the specified path will be included, except for testform.4gl. The directories CVS and mydir will also be excluded.

![Exclude files and directories](image)

Figure 220: Import Project example

Default organization of imported files

When files are imported into Project Manager, a new Group node is created and listed in the Projects view. For each folder, the process is:

- A File node is created for each file.
- The file nodes are then added to a Library; the name of the folder is used as the name of the Library node.
- If a file contains a MAIN program block, an Application node is created and the File node is added.
- A Group node is then created, and all Library and Application nodes, and their nodes, are added to this Group.

Build Rules Configuration dialog (Languages)

The Build Rules Configuration dialog is used to edit a language (add, modify, delete build/link/execution rules and variables) and to add a new language or remove an existing language.

Add / edit a language

![Build Rules Configuration dialog](image)

Figure 221: Build Rules Configuration dialog - Genero Studio

The integrated Toolbar allows for adding, duplicating, and deleting languages.

Add a language

Select this icon to add a new, empty language.
**Duplicate the current language**

Select an existing language and click this button to duplicate it.

**Delete a language**

Select an existing language and click this button to remove it. This action deletes only the rules and variables of the current editable level, for instance if you are editing the language from Project Manager, it will delete only the project rules and variables. If the language is empty it will remove the language.
Build tab
The Build tab in the Build Rules Configuration dialog is used to add, edit, and delete build rules.

Figure 222: Build tab

Build Rules List
The Build Rules List includes the build rules Default, Template, Global, and Project build rules for the selected language. The integrated Toolbar allows for adding, duplicating, deleting and editing build rules.

Add a build rule
Select the Global category (Preferences only) or Project category (Project Manager only) and click this
icon to add a custom rule. See Add/Edit a build rule on page 414.

**Duplicate selected build rule**
Select an existing rule and click this icon; the Build rule will be added to the appropriate category. Select the newly added rule and click the **Edit** icon to modify the duplicated rule.

**Delete a selected build rule**
Select an existing rule and click this icon to remove it.

**Edit a selected build rule**
Select a custom Build rule and click this icon to modify the Build Rule fields. See Add/Edit a build rule on page 414.

**Link tab**
The Link tab in the Build Rules Configuration dialog is used to add, edit, and delete link rules. A link rule is executed when you build an application or library node.

![Build Rules Configuration dialog](image.png)

Figure 223: Link tab
<table>
<thead>
<tr>
<th>Command lines</th>
<th>Commands that will be executed during the linking of an application / library node. See <a href="#">Command line options for build, link, execution rules</a> on page 416</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output files</td>
<td>List of files generated by the link command. See <a href="#">Predefined node variables</a> on page 436</td>
</tr>
<tr>
<td>Additional dependencies</td>
<td>List of files used by the link command to generate output files. If all of the listed files are less recent than the output files, the link is up to date. If the dependencies list is empty, the link rule is executed every time. If the dependencies contain one or several variables which are all empty (for example, $(BuildOutputFilePaths)), the link rule is never executed.</td>
</tr>
<tr>
<td>Overwrite default rules</td>
<td>If a link rule is already defined at an upper level (Default, Global, Template) the link rule is not editable unless you check the Overwrite default rules box to define a new link rule for the current level (Global or Project).</td>
</tr>
</tbody>
</table>
**Execution tab**
The Execution tab in the *Build Rules Configuration* dialog is used to set execution rules. An execution rule is executed when you run, debug, or profile an application node.

![Build Rules Configuration](image)

---

**Figure 224: Execution tab**

**Run command**
Command used when user runs an application.

**Debug command**
Command used when user debugs an application.

**Profile command**
Command used when user profiles an application.

**Overwrite default rule**
If a execution rule is already defined at an upper level (Default, Global, Template), the execution rule is not editable unless you check the  *Overwrite default rules* box to define a new link rule for the current level (Global or Project).
Variables tab
The Variables tab in the Build Rules Configuration dialog is used to set environment variables.

Figure 225: Variables tab

Environment variables
The Variables tab lists environment variables set for the selected language. The integrated Toolbar allows for adding, deleting and editing environment variables.

Add a variable
Select the Global category (Preferences only) or Project category (Project Manager only) and click this button to add a new environment variable. See Add or edit environment variables on page 168.

Delete a variable
Select an existing variable and click the button to remove it.

Edit a variable
Select a variable and click this button to modify variable settings. See Add or edit environment variables on page 168.
Variable values

See Predefined node variables on page 436.

Project Manager node properties

Properties can be set for the nodes of a project, to help define the component or specify its behavior.

The Properties view displays the properties for the selected group, application, library node, or file. The assigned value for the property displays. You can add or update the property value, or use the undo button to reset the property to its default value.

Note: See Package and Directory nodes and properties on page 1180 for Genero Mobile packaging and directory node properties.

![Properties view for an application node](image)

Specific Project Manager node variables can be used in the values.

Table 137: Project Manager properties

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
<th>May be inherited</th>
<th>Read-only</th>
<th>Group</th>
<th>Application</th>
<th>Library</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A short description of the group, application or library node.</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Compiler options</td>
<td>Options to be passed to the compiler. For example, -S to dump Static SQL messages found in the source.</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Label</td>
<td>Description</td>
<td>May be inherited</td>
<td>Read-only</td>
<td>Group</td>
<td>Application</td>
<td>Library</td>
<td>File</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>-----------</td>
<td>-------</td>
<td>-------------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>Target Directory</td>
<td>Target directory for output files for all applicable nodes. Compiled modules and link results (42m, 42f, 42r, and 42x files) will be stored in this directory.</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>The default directory is a bin directory created in the current Project directory. Use the Browse button to change the directory.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If the directory value is changed, Genero Studio tries to maintain a relative path having the project directory as its base. A relative path is not possible, however, if a different drive under Windows® is selected.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Important:</strong> For portability, we recommend that Target Directory should always be defined through a relative path, with the Project Directory $(ProjectDir)$ as the starting point.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Directory</td>
<td>Directory of source files.</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>External dependencies</td>
<td>See <a href="#">Setting external dependencies</a> on page 411.</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Command line arguments</td>
<td>Arguments passed to the FGL application when it is launched (Run or Debug). This is useful when the application is written to behave differently depending on arguments that are passed on the command line.</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>For example, the value <strong>123</strong> could be entered as the argument. This value could be retrieved in the application source code using the built-in function ARG_VAL, and the application could be written to respond accordingly.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exclude from build</td>
<td>Excludes the node from the build process.</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Exclude from compilation</td>
<td>Excludes the file from compile process.</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Exclude from link</td>
<td>Excludes the file from linking process.</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Label</td>
<td>Description</td>
<td>May be inherited</td>
<td>Read-only</td>
<td>Group</td>
<td>Application</td>
<td>Library</td>
<td>File</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------</td>
<td>-----------</td>
<td>-------</td>
<td>-------------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>Language</td>
<td>Programming language to be used when building a node. See Languages on page 413.</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Linker options</td>
<td>Options to be passed when linking.</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>File Path</td>
<td>The complete path of a file, including the file name.</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Web Service</td>
<td>When checked, indicates the application is a web service application.</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Web Service URL suffix</td>
<td>The suffix used to generate the URL when starting a Web Service application.</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

In a Genero Configuration using the GAS as a front-end, the URL generated for a Web Service application is changed to:

```
http://host:port/connector/ws/t/application_name/web_service_suffix
```

where:

- "ws" is used in the URL, instead of the traditional "ua" or "wa".
- `application_name` is the name of the application node.
- `web_service_suffix` is the value of the associated property. It can be empty. Other values depends on the current GAS configuration.

This property is found under the Application Server group.

**Predefined node variables**

Each Project Manager node has a defined set of variables containing values determined at runtime.

These variables can be used in **Build rules** or **Environment variables**. The variables available on a node are the concatenation of the variables defined for the node plus the ones defined for its ancestors.

**Table 138: Predefined node variables (X indicates available for the node)**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Group</th>
<th>Application</th>
<th>Library</th>
<th>File</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(BAFilePath)</td>
<td>Absolute path of the 4ba file.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>$(BinaryName)</td>
<td>Name of the binary node.</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
<td>Group</td>
<td>Application</td>
<td>Library</td>
<td>File</td>
<td>Package</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------</td>
<td>-------------</td>
<td>---------</td>
<td>------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| $(BuildInputFileDirs)    | List of all file directories for source files used by a build rule. For example, D:\My Genero Files\test\src\.
<p>|                          |                                                                                                                                             | X     | X          |         |      |         |
| $(BuildInputFileNames)   | List of all source files for a build rule. For example, main.4gl form.4fd test.per toto.4gl.                                                                                                             |       | X          | X       |      |         |
| $(BuildInputFilePaths)   | List of absolute file paths for source files used by a build rule. For example, D:\My Genero Files\test\src\main.4gl D:\My Genero Files\test\src\form.4fd. |       | X          | X       |      |         |
| $(BuildOutputBaseNames)  | List names of all output files for a build rule, without extension or path. For example, main form test toto.                                                                                              | X     | X          |         |      |         |
| $(BuildOutputFileNames)  | List of all output files for a build rule.                                                                                                     | X     | X          |         |      |         |
| $(BuildOutputFilePaths)  | List of absolute file paths for build output files.                                                                                             |       | X          |         |      |         |
| $(CommandLineArgs)       | Command line arguments.                                                                                                                        |       | X          |         |      |         |
| $(CompilerOptions)       | Options to be passed to the compiler. For example, -S to dump Static SQL messages found in the source.                                                                                                     |       | X          | X       |      | X       |
| $(Dependencies)          | List of link output files (link results) of all dependencies binary nodes. For example, your Application depends on 2 libraries, lib1 and lib2. The lib1 link rule creates the $(TargetDir)/lib1.42x output file, and the lib2 link rule creates the $(TargetDir)/lib2.42x output file. The Application Dependencies is a list of these paths: {$(TargetDir)/lib1.42x $(TargetDir)/lib2.42x} |
| $(DistDir)               | Value of the Distribution directory property.                                                                                                  |       | X          |         |      |         |
| $(ExecutableName)        | Executable name.                                                                                                                              |       | X          |         |      |         |
| $(ExternalDependencies)  | Value of the node property External dependencies.                                                                                               |       | X          | X       |      |         |</p>
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Group</th>
<th>Application</th>
<th>Library</th>
<th>File</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(GSTDIR)</td>
<td>Genero Studio installation directory path.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$(GSTSETUPDIR)</td>
<td>Application Generator template directory path.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>$(Implicit)</td>
<td>FGL argument -implicit=none</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>$(InputBaseName)</td>
<td>Input file base name, without extension.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>$(InputDir)</td>
<td>Absolute directory of the input file.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>$(InputExtension)</td>
<td>Input file extension.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>$(InputMimeType)</td>
<td>Input file MIME type.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>$(InputName)</td>
<td>Input file name, with extension.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>$(InputPath)</td>
<td>Absolute path of the input file.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>$(IntermediateFilePaths)</td>
<td>List of intermediate file paths.</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(Language)</td>
<td>Programming language to be used when building a node. See Languages on page 413.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>$(LinkerOptions)</td>
<td>Options to be passed when linking.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>$(LinkOutputFileBaseNames)</td>
<td>Names of all output files for a link rule, without extension or path.</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(LinkOutputFileName)</td>
<td>List of all output files for a link rule.</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(LinkOutputFilePaths)</td>
<td>List of absolute file paths for output files used by a link rule.</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(PackageName)</td>
<td>Value of the PackageName property.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>$(ProjectDir)</td>
<td>The directory where the 4pw file is located.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
<td>Group</td>
<td>Application</td>
<td>Library</td>
<td>File</td>
<td>Package</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------</td>
<td>-------------</td>
<td>---------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>$(RelativeDir)</td>
<td>A list of directory names, corresponding to the path difference between the</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>source directory and the current directory. For example, the source</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>directory is $(ProjectDir)/src, and the file path is $(ProjectDir)/src/com/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>d1/Account.java. The relative directory would be the string list {com d1}.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>To use, you typically join with a separator:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• For paths that use the backslash:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$(RelativeDir</td>
<td>/)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• For a java package using a period:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$(RelativeDir</td>
<td>.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(TargetDir)</td>
<td>Target directory of the compiled files.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(TargetPaths)</td>
<td>List of target directories of all dependencies.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(XCFFilePath)</td>
<td>Path of the XCF file found for the application.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**List Expansion**

For all variables that contain a list of files, for example $(BuildInputFiles)$, special syntax is used to expand the list.

To get a list with a specific file extension, use a colon:

```
$(variableName:.extension)
```

For example, $(BuildInputFiles:.4gl) returns all files with a 4gl extension in the $(BuildInputFiles) list, separated by a space.

To get a list with a specific separator, use the pipe symbol:

```
$(variableName|separator)
```

For example, $(BuildInputFiles|#) returns all the files in the $(BuildInputFiles) list, separated by a #.
### Project Manager error messages

A list of Project Manager error messages. For messages that are not self-explanatory, additional information is provided.

**Table 139: Project Manager error messages**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
</table>
| GS-12001 | ‘%1’ could not be found on the disk.  
The file to open is not present on disk.  
Check if the file path is correct. |
| GS-12002 | ‘%1’ is not on the same drive than the project ‘%2’. The project file will not be portable.  
The file is not on the same drive as the project, resulting in no relative path between the file and the project. This causes some limitations, for example using versioning will not be possible. |
| GS-12003 | Cannot load ‘%1’ language file  
Language file loading error.  
Transform to a generic loading error message with the file path. |
| GS-12004 | Cannot load ‘%1’ platform file  
Platform file loading error.  
Transform to a generic loading error message with the file path. |
| GS-12008 | GSTSETUPDIR=’%1’ doesn't exist  
GSTSETUPDIR is relative or doesn't exist.  
Check the path in the Genero configuration. |
| GS-12009 | Environment variable '%1' should not be defined in the project but in the config only  
One invalid environment variable has been set in the project. The following list contains the forbidden variables:  
- GREDIR  
- GSTSETUPDIR  
- GSTWCDIR  
- FGLDIR  
Move its definition to the Genero Configuration. |
| GS-12010 | Some files are not on the same drive than the project file.  
The project has been saved with absolute paths and will not be portable.  
Error message and resolution should be self-explanatory. |
| GS-12011 | Unable to create a node of type ‘%1’.  
Internal error: incorrect project file format. A node could not be created. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-12012</td>
<td>For external project file format version must be equal or greater than '%1'.</td>
</tr>
<tr>
<td></td>
<td>Internal error: an external project of an unsupported format has been added.</td>
</tr>
<tr>
<td>GS-12013</td>
<td>Unable to write the file '%1'. Check path existence and permissions</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12014</td>
<td>Unknown encoding '%1'.</td>
</tr>
<tr>
<td></td>
<td>No suitable codec can be found for the specified encoding.</td>
</tr>
<tr>
<td></td>
<td>Add a new alias in the encoding map or add a new codec charmap.</td>
</tr>
<tr>
<td>GS-12015</td>
<td>Cannot load external project '%1', project already loaded.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12016</td>
<td>Removed circular dependency on %1 %2 from %3 %4.</td>
</tr>
<tr>
<td></td>
<td>During the load, a circular dependency was detected and removed.</td>
</tr>
<tr>
<td>GS-12019</td>
<td>Multiple Business Application Diagram present in the project.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12020</td>
<td>Missing dependency '%1' for node '%2'.</td>
</tr>
<tr>
<td>GS-12251</td>
<td>The file '%1' is already present in the project and has not been imported</td>
</tr>
<tr>
<td></td>
<td>The import action tries to add a file which is already present. The file is ignored (warning).</td>
</tr>
<tr>
<td>GS-12252</td>
<td>Cannot compute dependencies for function '%1' because it is defined multiple times</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12254</td>
<td>File is already present in the project</td>
</tr>
<tr>
<td></td>
<td>This warning signals that one of the saved files is already present in the project and won’t be added a second time.</td>
</tr>
<tr>
<td>GS-12255</td>
<td>A business application diagram file is already present in the project</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12401</td>
<td>Added dependency on '%1' node to '%2' library</td>
</tr>
<tr>
<td></td>
<td>Information message: the import process created a dependency.</td>
</tr>
<tr>
<td>GS-12501</td>
<td>Unable to find the wsdl compiler. Check your web services installation.</td>
</tr>
<tr>
<td></td>
<td>Fglwsdl tool is not found in the fgl install.</td>
</tr>
<tr>
<td></td>
<td>Check if the Genero configuration uses an fglgws VM.</td>
</tr>
<tr>
<td>GS-12502</td>
<td>Cannot load external dependency '%1'</td>
</tr>
<tr>
<td></td>
<td>An error occurred loading an external project (4pw).</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| GS-12503 | Cannot load project database for external dependency ‘%1’  
An error occurred loading an external project database (4pwdb).                                                                                      |
| GS-12505 | Project database: <ERRORMESSAGE>  
The content of error message GS 12505 can vary because it is an error that occurs during the build process of the project manager database, and is dependent on the external tool that is running. This will likely be a compiler (fglcomp) error.  
Error message and resolution should be self-explanatory.                                                                                             |
| GS-12506 | Cannot add circular dependency on %1 node to %2 library  
‘Compute dependencies’ could not add a dependency because it would create circular dependencies within the project.                                                                                   |
| GS-12509 | Cannot create package XML file ‘%1’  
An error occurred during $(GenerateXMLPackage) command.                                                                                               |
| GS-12510 | Unknown argument '%1' to $(generateXMLPackage) task.  
The $(GenerateXMLPackage) command arguments are incorrect.                                                                                         |
| GS-12511 | Unknown property ‘%1’  
Some unknown property is defined. This message mostly appears for AG settings.                                                                             |
| GS-12512 | Load failed  
General error: an error occurred during file load.                                                                                               
Error message and resolution should be self-explanatory.                                                                                                                                                 |
| GS-12513 | Unsupported version  
Error message and resolution should be self-explanatory.                                                                                           |
| GS-12514 | Missing version  
An error has occurred with the pm-settings.conf file in the template directory $(GSTSETUPDIR) because the version attribute is missing. Note: this message may occur with other files using a version attribute. |
| GS-12515 | Unknown node '%1'  
Error message and resolution should be self-explanatory.                                                                                           |
| GS-12516 | Platform '%1' is not defined  
Error message and resolution should be self-explanatory.                                                                                           |
| GS-12517 | Language '%1' is not defined  
Error message and resolution should be self-explanatory.                                                                                           |
| GS-12518 | Save failed.  
Check if the file path is correct, and that you have the appropriate rights.                                                                           |
| GS-12519 | Start GCC task failed.  
Check if the file path is correct, and that you have the appropriate rights.                                                                           |
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-12520</td>
<td>GGC test generation failed : no 4gl file generated. Check if the file path is correct, and that you have the appropriate rights.</td>
</tr>
<tr>
<td>GS-12521</td>
<td>Unused function <code>%1</code>. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12751</td>
<td>Configuration isn't valid. Operation canceled. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12753</td>
<td>Cannot create execution task. This is caused by an internal system error. Contact your local Four Js support center.</td>
</tr>
<tr>
<td>GS-12754</td>
<td>Cannot create profiling task. This is caused by an internal system error. Contact your local Four Js support center.</td>
</tr>
<tr>
<td>GS-12755</td>
<td>Cannot create debugging task. This is caused by an internal system error. Contact your local Four Js support center.</td>
</tr>
<tr>
<td>GS-12756</td>
<td>The debugger is already running. You have to stop the current debugger session to start a new one. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12757</td>
<td>Operation cannot be performed, no item selected Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12758</td>
<td>Unable to find the internal FGL installation. Check your <code>%1</code> installation. Internal error: Genero Studio install malfunctioned. Reinstall Genero Studio to solve the problem.</td>
</tr>
<tr>
<td>GS-12759</td>
<td>Unable to start an import on the selected node. Internal error: the import does not work on the currently selected node (only the main project and group are supported).</td>
</tr>
<tr>
<td>GS-12760</td>
<td>Unable to build an unsaved project using a remote server. Unsaved projects can only be built with local FGL installations Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12761</td>
<td>Genero installation isn't valid. Operation canceled. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12762</td>
<td>Front end isn't valid. Operation canceled. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-12763</td>
<td>File must be saved in the source directory</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12764</td>
<td>Cannot create tracing task.</td>
</tr>
<tr>
<td></td>
<td>This is caused by an internal system error.</td>
</tr>
<tr>
<td></td>
<td>Contact your local Four Js support center.</td>
</tr>
<tr>
<td>GS-12900</td>
<td>An error occurred while checking FGL tools: %1.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12901</td>
<td>Projects built successfully: %1.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12902</td>
<td>Projects built with errors: %1.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12903</td>
<td>FGLDIR environment variable doesn't correspond to an FGL installation.</td>
</tr>
<tr>
<td></td>
<td>Ensure you are in an FGL development environment.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12904</td>
<td>FGLDIR environment variable not set. Ensure you are in an FGL development</td>
</tr>
<tr>
<td></td>
<td>environment.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12905</td>
<td>You have to enter only one operation (-b, -r, -c or -force-build).</td>
</tr>
<tr>
<td></td>
<td>Specify only one gsmake command.</td>
</tr>
<tr>
<td>GS-12907</td>
<td>Wrong parameter for %1 argument.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12908</td>
<td>Error loading %1.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12909</td>
<td>Unable to open the file '%1'.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12910</td>
<td>Unable to open the project '%1'.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12911</td>
<td>Error loading Business Application diagram.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12912</td>
<td>The project has no default application.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12913</td>
<td>The project doesn't contain '%1'.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>GS-12914</td>
<td>GSMake server socket closed.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12915</td>
<td>GSMake server socket error.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-12916</td>
<td>Invalid Template directory '%1'.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
</tbody>
</table>

**Code Editor**

*Code Editor* is a programming-oriented editor. In addition to editing source code, it can handle any kind of text as well as languages such as 4GL and XML. Smart editing features like auto-completion, code templates, and robust search make coding easier and more efficient.

- Editing code files on page 445
- Using the Diff tool on page 451
- Code coverage on page 454
- Printing files on page 457
- XML Catalog files on page 457
- Code Editor Reference on page 458

**Editing code files**

The Code Editor contains basic and smart functionality for creating the code that drives your applications.

- Code Editor basics on page 445
- Smart editing - indenting, tabs, and backspace on page 446
- Bookmarks on page 447
- Auto completion (Ctrl+Space) on page 447
- Code templates (Ctrl+T) on page 447
- Split a document on page 448
- Search and replace on page 448
- Set the color and font of identifiers on page 456

**Code Editor basics**

The active document is displayed in the Document workspace. If this document is a code document, such as a 4gl file, the Code Editor functions are active.

Multiple documents can be open, in tabbed windows, with the filename on the tab.

By default, line numbers are displayed at the left side of the window.

The status bar at the bottom of the window contains the selection range (line and column of the start and end position), the format (Windows®, UNIX™, Mac®), and the mode (insert, overstrike).

The Code Structure view displays information about the structure of the active file. Clicking on an element in the Structure view will display and highlight the corresponding lines in the Editing window.
vi Editor
You can change the editor to use vi commands. Select **Edit > VI Editing Mode** to switch the editor.

Syntax highlighting
The elements of the program are visually highlighted. Language key words, strings, variables and comments are each colored differently, making the program structure easier to understand. Select **Tools > Preferences > Code Editor** to customize the behavior and color.

Syntax errors
An error mark in the gutter flags syntax errors as they occur. Select the error mark to display a message concerning the error. The error message and line number also display in the **Document Errors** tab in the output.

Menu of options
Use the Edit menu or right-click in document to display some options for selecting, searching, and editing. The available accelerator keys are documented in the **Keyboard Shortcuts** on page 472 page.

Document Format
You can change the document format (Windows®, UNIX™, MAC) by selecting **Edit > Convert to**.

Integrated diff
The **Using the Diff tool** on page 451 tool is integrated into Code Editor.

Folding text
Folding condenses portions of text based on the scope of statements.
Click the + or - symbols in the left gutter to fold or unfold all corresponding lines. Use the **View** menu or right-click context menu options to **Fold, Fold all**, or **Unfold all**.

Finding declarations
Press the CTRL key while clicking on a function or variable name to find the declaration of that function or variable.

Finding callers of a function
To find out where a function has been called, right-click on the function name and select **Callers**.

Square selection
Press the ALT key while selecting text with the mouse to select a rectangle of text, instead of entire lines. See **Selection keymap** on page 473.

Related concepts
**Code Editor preferences** on page 459
Set preferences for the Code Editor. For example, you can set the default diff mode or the color and font preferences.

Smart editing - indenting, tabs, and backspace
Code Editor has smart editing features for indenting, tabs, and backspace.

Smart Indent
The Smart Indent feature is enabled by default. After you press the Enter key, the new line is indented to align with the immediately preceding non-blank line.
For example:

```
01 LET var1 = 55
02 LET var2 = 22
```

If you do not wish your new line to be aligned with the preceding line, use the Left Arrow key to move the cursor towards the left margin of the page.

**Smart Tabs**

The Smart Tabs feature is enabled by default. When the Tab key is pressed at the beginning of a line, the cursor is aligned with the first character following the next whitespace on the immediately preceding non-blank line.

For example:

```
01 DEFINE
02    var1 INTEGER,
03    var2 INTEGER
```

If the preceding line does not have embedded whitespace, the Tab key moves the cursor the number of spaces indicated by the **Tab size** setting in the Behavior and display preferences.

**Smart Backspace**

The Smart Backspace feature is enabled by default. Pressing the Backspace key will move the cursor back in the current line to the position of the indent in the previous line.

**Disable Smart features**

Disable Smart features by selecting Tools > Preferences > Code Editor, Behavior & Display from the menu.

**Bookmarks**

Bookmarks provide quick access to areas of a document.

**Add a bookmark**

1. Place the cursor where you want to add the bookmark.
2. Right-click and select **Toggle bookmark**.

**Viewing bookmarks**

The **Bookmarks** view displays a list of bookmarks for the current project.

**Related concepts**

- **Bookmarks view** on page 117
- The **Bookmarks** view lists the bookmarks for the current project.

**Auto completion (Ctrl+Space)**

The auto complete feature helps complete a line of code or prompts for a valid keyword in the syntax.

Type the first letters of a word and then press Ctrl-space to complete the word or select a word from a list of options.

**Code templates (Ctrl+T)**

Code templates are snippets of frequently used code elements available to insert into the code to avoid repetitive typing and speed up coding.

To use a code template in your code, select one of these methods:

- Type the code template name, such as "**case**", in your code where you want it to appear; Press **Ctrl+T**, or right-click to display the contextual menu and choose **Expand Template**.
• Position the cursor where you want the code to appear, and press Ctrl+T to display a list of code templates. Select from the list using the arrow keys, and press Enter. (Press Esc to cancel)

The selected template is inserted in the document using the exact layout (tabs, linefeeds, and so on.) A corresponding number of lines are shifted.

Templates are user-customizable. Use Tools > Preferences > Code Editor > Code Templates to add or modify a template.

**Example: Adding a "case" instruction code template**

Use the popup window (Ctrl-T) to select a keyword, or type "case" and press Ctrl-T.

![Code template picture]

The code template is inserted in your code, ready for you to complete.

**Figure 227: Code template pop-up window**

![Code template code example]

**Figure 228: CASE code template**

**Split a document**

Splitting the document view allows you to see different parts of a long document at the same time.

Split/unsplit a document using the View menu option. Once a split is requested, the current document window splits into two separate windows which can be scrolled independently. Each window may be split again twice.

Highlight rules are applied to the same text in every window pane. For example, placing the cursor on the IF statement in one window also highlights the same IF statement in other windows in which it is displayed.

**Search and replace**

The search functionality in Code Editor includes wildcards, regular expressions, and function search.

- Using wildcards in search on page 449
- Using regular expressions in search on page 449
- Group capture in regular expressions on page 450
- Function search on page 451

**Related concepts**

Search/Replace view on page 467
The Search/Replace view allows you to search for and replace text in one or multiple documents.

**Using wildcards in search**
The *Use Wildcards* option will find the specified combination of characters, including combinations within a word.

**Table 140: Use Wildcards**

<table>
<thead>
<tr>
<th>Wildcard</th>
<th>Description</th>
<th>Example expressions</th>
<th>Matches any combination of characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Substitutes for any number of characters</td>
<td>1. st*e 2. *rec</td>
<td>1. that begins with st, followed by zero or more characters, followed by e: custrec, store, steady 2. that begins with zero or more characters, followed by rec: gr_custrec, RECORD</td>
</tr>
<tr>
<td>?</td>
<td>Substitutes for a single character</td>
<td>• st?e</td>
<td>• that begins with st, followed by a single character, followed by re:store</td>
</tr>
</tbody>
</table>

**Using regular expressions in search**
The Search tool will search for an exact match to text in the Find box, unless you specify match conditions using regular expressions, special meta characters, and predefined regular expressions.

Entering the string *FUNCTION*, will find *FUNCTION* or *function*, but not the string *fun*. If the Search option *case sensitive* is checked, the search will distinguish between uppercase and lowercase letters.

**Meta Characters**
- The meta character \ matches the character following it, except when followed by a left or right round bracket, a digit 1 to 9, or a left or right angle bracket.
- The special characters "\]" and "\-" have no special meaning if they appear as the first characters in the set.

**Table 141: Meta Characters**

This table provides the wildcard symbol or syntax, a description, an example expression, and examples of strings that match the example expression. For some entries, more than one example expression - string match pair are provided.

<table>
<thead>
<tr>
<th>Wildcard</th>
<th>Description</th>
<th>Example expressions</th>
<th>Matches any string</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Substitutes for any single character</td>
<td>• fla.</td>
<td>• containing four letters that begin with fla: flag, FLAG, flannel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• b.g</td>
<td>• containing three letters in the format b\xb: big, bog, Bog, bag</td>
</tr>
<tr>
<td>*</td>
<td>Substitutes for zero or more occurrences of the preceding expression/character</td>
<td>• a *b (notice the blank before the *)</td>
<td>• &quot;a&quot; followed by zero or more blanks then &quot;b&quot; &quot;a basic&quot; &quot;abasic&quot;</td>
</tr>
<tr>
<td>+</td>
<td>Substitutes for one or more occurrences of the preceding expression/character</td>
<td>• a +b (notice the blank before the +)</td>
<td>• &quot;a&quot; followed by one or more blanks then &quot;b&quot; &quot;a basic&quot; &quot;a basic&quot;</td>
</tr>
<tr>
<td>Wildcard</td>
<td>Description</td>
<td>Example expressions</td>
<td>Matches any string</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>---------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>\</td>
<td>Searches for the character following; this cancels the special significance of the meta characters including itself, allowing a search for them. When used in a set, it is treated as an ordinary character.</td>
<td>- +100</td>
<td>- containing +100; treats + as an ordinary character</td>
</tr>
<tr>
<td>\</td>
<td>- \user</td>
<td></td>
<td>- containing \user; treats \ as an ordinary character</td>
</tr>
<tr>
<td>[set]</td>
<td>Defines a set of characters enclosed in square brackets ([...,]) to be used for matching; may define character ranges, as in [a-z] and [0-9]. If the first character in the set is &quot;^&quot;, it matches any character NOT in the set.</td>
<td>- [bd]og</td>
<td>- containing bog, dog</td>
</tr>
<tr>
<td></td>
<td>- [Tt]ooltip</td>
<td></td>
<td>- containing Tooltip, tooltip</td>
</tr>
<tr>
<td></td>
<td>- b[^o]g</td>
<td></td>
<td>- containing three characters, b, &lt;any character but o&gt;, g: bag, big</td>
</tr>
<tr>
<td></td>
<td>- [A-Da-d]+</td>
<td></td>
<td>- containing one of the alpha characters a through d inclusive, in uppercase or lowercase: define, Define, age</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>Matches either expression x or expression y (composite expression)</td>
<td>- bob</td>
</tr>
<tr>
<td>xy</td>
<td>Strings multiple expressions together, finding a single string containing expression x and expression y (composite expression)</td>
<td>- def.* iti.*</td>
<td></td>
</tr>
<tr>
<td>^ $</td>
<td>Restricts the pattern matching to strings at the beginning of the line (^ character) and/or the end of the line ($ character),</td>
<td>1. ^when 1. with when at the beginning of the line 2. test.$ 2. with test. at the end of the line.</td>
<td></td>
</tr>
</tbody>
</table>

**Group capture in regular expressions**

When the Use Regular Expressions option is checked, group capture allows you to isolate groups in the expression to be matched, so they can be captured and substituted during the replacement.

Given this expression in your document:

```
4+5*6+88
22+4*555
```

You can transform this express to these patterns using group capturing: `(4+5) * (6+88)` and `(22+4) *555`.

In the first expression there are four groups. First you must capture the individual groups in the expression by enclosing them in parenthesis. Specify that the characters are integers using the regular expression `[0-9]+` (one or more integers). Use the escape character `\` to indicate that the `*` and `+` symbols are literal and not meta characters.

Replace expression indicating the desired pattern and the positions of the groups (numbered from left to right): (\1+\2)*\3\4

Result: (4+5)*(6+88)

**Function search**
In Code Editor, you can search a file for all or part of a function.

Display the Function Search view with Window > Views > Function Search.

Enter a part of the function name to return all functions in all files in the project that meet the search criteria. For example, entering ord returns a list of functions with ord in their name such as order_new() and close_order().

Double-click on a line in the results to go to the selected function.

---

**Using the Diff tool**

The **Diff** tool compares two files: a read-only base copy of the file and a working copy. It is integrated into Code Editor.

**Comparing files**
To select files to compare, select Tools > Diff.

The Diff tool automatically selects the Diff base file and flags differences between the base file and the working copy with color-coded markups in the Code Editor gutter. Colors can be changed in preferences.

- Green - added lines
- Orange - modified lines
- Red - deleted lines

The base file is selected in this order:

**Use Generated File**

The default for generated files, this option compares the document with the file generated by the application generator. This is the generated file before POINTs and BLOCKs are injected into the file. This option is made available only if the file has been generated.

**Use Repository File**

The default for versioned files, this option compares the document with the file in the repository for the current SVN version of the file. This option is enabled only if the file is versioned.

**Use File on Disk**

Compares the document with the contents of the file as it was on disk when the file was opened. If you select this option again after the file has been opened, the diff data will be refreshed by comparing the document with the actual content of the file on disk.

You can change the base file used with Diff > Base File.

Specify how the tool compares white space and case in preferences.

**Display modes**
You can select a Diff display mode from the Diff menu.

**Normal View**

This view opens the file in normal editing mode.
Diff View

This view displays differences between the base file and working copy in a single pane document. This is also called single pane diff mode. New and modified lines are clearly marked by color. This view identifies the location of deleted lines, but does not display them.

Diff View with Deleted Blocks

This view displays differences between the base file and working copy in a single pane document with deleted blocks highlighted in red.

Vertical Dual Diff View

This view displays the two files vertically, with a left and right pane. It opens the base file (read-only) in the left pane and the editable working copy in the right pane. This is also called two pane diff mode.

Horizontal Dual Diff View

This view displays the two files horizontally, with a top and bottom pane. It opens the base file (read-only) in the top pane and the editable working copy in the bottom pane. This is also called two pane diff mode.

Example: Diff View

This figure shows Diff View. Lines 28-32 are marked with green blocks to indicate newly added lines. The orange block in the gutter of line 24 indicates a modified line (a new comment, highlighted in green, has been added to the existing line). The red line between line 22 and line 23 flag the location of a deleted line/lines.

Figure 229: Diff View display mode

Example: Vertical Dual Diff View

This figure shows Vertical Dual Diff View. Side-by-side comparison of the base file in the left pane and the editable working copy in the right pane give you a before and after record of changes. Red blocks in the gutter of the base file pane show the contents of lines deleted in the working copy.
Figure 230: Vertical Dual Diff View display mode

Diff navigation
Locate the changes quickly with Diff navigation options.

- **First Difference**: Locate the first difference.
- **Previous Difference**: Locate the previous difference.
- **Next Difference**: Locate the next difference.
- **Last Difference**: Locate the last difference.

Reverting a difference
To revert a difference, place the cursor in the line of marked code that you want to revert. Select **Diff > Copy to Right**. The **Copy to Right** action replaces the difference in the working copy with the original values from the base copy. Select **Diff > Copy All to Right** to revert all changes.

Restricting the differences shown
To avoid showing certain types of differences, you can choose from the following options:

- **Ignore case**: Ignore differences between uppercase and lowercase.
- **Ignore all white space**: Ignore white space, including tab, vertical tab, form feed, carriage return, and space. For example, the following are treated as the same:

  ```
  LET monthValue=month(dateValue)
  LET monthValue=month(dateValue)
  ```
Ignore space changes

Ignore internal and trailing white spaces, but not spaces at the end.

LET monthValue=month(dateValue)

Is the same as:

LET monthValue=month(dateValue)

But not the same as:

LET monthValue=month(dateValue)

Try hard to find a smaller set of changes

Show a smaller set of differences. You might want to select this option if your Diff output is too large, although it might cause Diff to run more slowly.

You can choose these options when you first select your two files, or use the toolbar at the bottom to add or remove options.

Command line option

Use `generostudio -diff file1 file2` at the command line to open the given files in Diff mode.

Code coverage

The Code Coverage view displays the number of times each line of code has been executed, if it has never been executed, or if it is not reachable at all.

Activating code coverage

1. In your current configuration, select the Code Coverage environment set.
2. Open a source code (.4gl) file.
4. Run the application.
5. Exit the application.

The statistics are displayed in the Code Coverage View.
Figure 231: Code Coverage view

To find out more about how source code coverage works under the covers, see Source code coverage in the Genero Business Development Language User Guide.

Clearing code coverage values

The code coverage (.cov) files are created in the same location as your .42m files. To reset the code coverage values, remove these files from the directory.

Highlight identifiers

Specify the identifiers to highlight in Code Editor, and define the colors to use.

Identifiers refer to special words or terms used when coding your application. For example, identifiers can be variable names, table names, cursor names, and so on.

You have the ability to determine which identifiers Code Editor highlights, and what colors are used.

Define identifiers

Create and import a .4id file to define custom sets of extra identifiers.

By default, eight sets of extra identifiers are provided in the Language Specific tab for the Genero BDL language. You can update these eight sets using the graphical user interface, or you can manage your sets by creating and importing a .4id file.

Extra identifiers can be imported into the Code Editor preferences using the Import extra identifiers tool found in the Code Editor preferences for the Genero BDL language.

1. Create a .4id file.
The .4id file is an XML file that defines extra identifier sets.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<ExtraIdentifiers>
    <ExtraIdentifiersSet identifiers="identifiers" name="name" set="set"/>
    ...
</ExtraIdentifiers>
```

where:

- **identifiers** is a space-separated list of identifiers.
- **name** is the set name, which can be left empty. If the set name is left empty, a default set name is used.
- **set** is the set number, from 1 to 8. There is a maximum of 8 sets of extra identifiers.

2. Select the **Tools > Preferences** menu option.
3. Navigate to **Code Editor > Behavior & Display**.
4. Select **Genero BDL** in the **Language** combobox.
5. Select the **Language Specific** tab.
6. Check **Use custom settings** to enable the **Extra identifiers** section.
7. Under **Extra identifiers**, click the Import extra identifiers icon.
8. Select the .4id file to import.

### Set the color and font of identifiers

Create a custom keyword set for keywords you want to highlight using a specific font and color in a Code Editor document.

A keyword set allows you to group a specified list of keywords for the purpose of assigning a color and font for display in Code Editor. For example, if you wanted the terms FOO and BAR to appear using a specific color and/or font, you would add them to a keyword set, then define the style for that keyword set to use the color and font you desire.

1. Select **Tools > Preferences**.
2. Under **Code Editor**, select the **Behavior & Display** preference page.
3. In the **Language** combobox, select the language.
4. Check the **Use custom settings** checkbox.
5. Click on the **Language Specific** tab.
6. Update the details for an extra identifier set.

   This is the set you will add your custom keywords to, and that you will later define a style for.

**Tip**: This procedure uses the Genero Studio interface to create and manage your extra identifiers sets. These sets can also be managed by importing an XML file. See **Define identifiers** on page 455 for more information.

   a) Under **Extra identifiers**, select the name of a set.
   b) Click the **Edit identifiers** icon.
   c) Modify the set name, if desired.
   d) Use the appropriate toolbar icons in the **Edit List** dialog to add, modify, and remove keywords from the set.
   e) When you are finished, click **OK**.

   At this point, you have grouped a set of custom keywords into one of the eight provided custom keyword sets.

7. Go back to the **Pages** list along the left-hand side of the **Preferences** window. Under **Code Editor**, select the **Color & Font** preference page.
8. Under the **Style Specific Settings** section, select the **Language** and check the **Use custom settings** checkbox.
9. Scroll and select the name of the keyword set.
10. Under **Color**, use the color picker to set the foreground and background colors for the keywords in this set.
11. Under **Font**, select the font family, size, bold and italic settings. To change the font family or size, check the **Use custom font for selected style** checkbox.

**Related concepts**
- **Behavior & Display preferences** on page 460
  Set preferences for the behavior and display aspects of the Code Editor.
- **Color & Font preferences** on page 461
  Set preferences for the behavior and display aspects of the Code Editor.

**Printing files**

Print or preview the files in Code Editor.

- **File > Print**
  Prints the file using the options available to your operating system printers.

- **File > Print Preview**
  Displays a preview of the printed file in a preview window. For diagrams, scaling and number of pages to use to print can be configured.

**XML Catalog files**

XML Catalog files are used to provide an alternative path when the external entities are not accessible, or are not where the XML document specifies.

Code Editor recognizes XML documents and provides XML validation. Color coding, Smart Editing, and Code Completion features can be used with an XML file.

Many XML documents contain external links to stylesheets, schemas, DTDs, and so on, which may be stored on remote systems. If the links are absolute URLs, they only work when your network can reach them. However, the entity resolver of the XML SAX parser can be used to determine whether there is a local equivalent in your system’s **Catalog** application level cache.

When you use an XML Catalog file to map these external references to local equivalents, the task of locating the reference is shifted from the XML documents to the XML Catalog files, which can be varied for different audiences. Since local copies of the references are accessed instead of remote network resources, and the local copy can be a subset of the complete schema or DTD in the external reference, XML processing may be faster.

The XML Catalog files to be used by Genero Studio are specified through the **General Preferences** on page 128.

**Managing XML Catalog entries**

Use **Tools >> Preferences** and select **XML Schema/DTD** from the **Code Editor** page in the Pages tree. A list of the XML catalog files to be used with Genero Studio is displayed.

You can add, edit, and delete entries for these catalog files through **XML Schema/DTD Configuration**.

**XML Catalog specification**

Genero Studio provides a subset of the OASIS XML catalog specification:

- The **catalog** element to assign a name to the catalog
- **System** and **public** elements to resolve DTDs
- The **uri** element to resolve schema location for XSDs
- The **nextCatalog** element to refer to another catalog file

**Example File**

This example illustrates the features of a Genero Studio XML Catalog file:
Figure 232: Genero Studio XML Catalog file

The `catalog` element is the root of an XML Catalog. The xmlns attribute assigns a name to the catalog file.

Additional Elements in this file are:

1. `public` - resolves a Public ID for an external DTD; specifies Public external DTDs intended for broad use (public distribution of a DTD file for a wider audience).
2. `system` - resolves a relative system id for an external DTD; used to find the public DTD if it cannot be located by the public ID.
3. `system` - resolves a web-based system id for an external DTD.
4. `uri` - used to locate XSL and other files. It can be used for everything that is not a declared PUBLIC or SYSTEM identifier for a DTD or system entity file.
5. `nextCatalog` - resolves entries related to Additional Catalogs.

Related concepts
- XML Schema/DTD preferences on page 463
  Set preferences for the XML schema and DTD aspects of the Code Editor.
Code Editor preferences

Set preferences for the Code Editor. For example, you can set the default diff mode or the color and font preferences.

Select Tools > Preferences > Code Editor to access Code Editor preferences.

Shortcut keys can be customized with Tools > Preferences > User Interface, Accelerators.

VI Mode Settings

Table 142: VI Mode Settings options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open file with VI mode editing check box</td>
<td>If checked, the behavior of the editor is modified to behave like a UNIX™ editor.</td>
</tr>
</tbody>
</table>

Diff Mode Settings

Table 143: Diff Mode Settings options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default diff mode combobox</td>
<td>Set the default diff mode. This is the mode in which Code Editor will open files. This mode has no effect when comparing two files on disk; dual pane will always be opened in this case.</td>
</tr>
<tr>
<td>No diff</td>
<td>Open in normal editing mode.</td>
</tr>
<tr>
<td>Single pane</td>
<td>Open in diff mode in a single pane view.</td>
</tr>
<tr>
<td>Single pane with deleted blocks</td>
<td>Open in single pane with deleted blocks highlighted.</td>
</tr>
<tr>
<td>Horizontal dual pane</td>
<td>Open in two panes with the base file on the top and the file to compare on the bottom.</td>
</tr>
<tr>
<td>Vertical dual pane</td>
<td>Open in two panes with the base file on the left and the file to compare on the right.</td>
</tr>
</tbody>
</table>

• Encoding (i18n) on page 459
• Behavior & Display preferences on page 460
• Color & Font preferences on page 461
• Template preferences on page 462
• XML Schema/DTD preferences on page 463

Related concepts

Setting Preferences on page 128
Customize Genero Studio to meet your needs.

Encoding (i18n)

Text Editor supports internationalization (i18n), allowing you to enter special characters like Japanese, Chinese, or Indian scripts, and to enter text from right to left as in Arabic.

The characters typed at the keyboard are intercepted and changed automatically, based on the Encoding method selected in General Preferences on page 128. The default Text Encoding is Default charset.
Related concepts
Language support (text encoding) on page 189
Text encoding specifies the character set to be used by Genero Studio. You can change the default encoding to work in your preferred language.

Behavior & Display preferences
Set preferences for the behavior and display aspects of the Code Editor.

Language
You can set language preferences for all languages, then overwrite preferences for specific languages.
To set preferences for all languages, select All Languages.
To set preferences for a specific language, select the Language combobox, then select the Use custom settings checkbox. This box must be checked to make any changes in the settings for the selected language. The language is identified as having a custom setting by the addition of a pencil icon next to the language name.
When you select a language, changes to preferences will only apply to that language.

Behavior preferences tab
Tabs:
• Tab size - Size of tab in characters.
• Insert spaces for tab - If selected, tab size whitespaces are inserted into the document instead of the tab character when the Tab key is pressed.

Indentation:
• Smart tab - If selected, Code Editor will move the cursor to align with the first character following a whitespace on the previous non-blank line, when the Tab key is pressed.
• Smart backspace - If selected, Code Editor will backspace intelligently through whitespace.
• Smart indent - If selected, Code Editor will indent code based on the indentation of the previous line. New lines are indented to the first non-blank character of the line above.
• Strip trailing white spaces - If selected, Code Editor will remove all trailing whitespace from the file.

Automatically Close:
If selected, the closing symbol will be added to each of these:
• String " "
• Single quotes ''
• Parenthesis ( )
• Curly braces { }
• Square brackets [ ]
• Angle brackets < >

Smart Key Options:
• Smart home - If selected, the Home key moves the cursor to the left of the first non-blank character on the line.
• Smart end - If selected, the End key moves the cursor to the right of the last non-blank character on the line.

Copy Options:
• Copy whole line when no selection - If selected, the entire line is copied when you choose Copy and nothing is selected.

New Document Format:
Sets the default line ending for new text documents:
• **Windows®** - Use the CRLF end-of-line format, as used by Windows® and most other early non-UNIX® and non-IBM operating systems.

• **UNIX/Mac®** - Use the LR end-of-line format, as used by UNIX-like systems including macOS™.

• **Mac® 9** - Use the CR end-of-line format, as used by Mac® OS up to version 9.

**Display preferences tab**

**Editor:**

• **Show line numbers** - If selected, Code Editor displays line numbers.

• **Show Right Margin** - If selected, Code Editor displays a thin line on the right side to indicate the right margin. The spinner sets the column number where the right margin line will be located, in characters.

• **Show indentation guide** - If selected, Code Editor displays vertical lines that indicate relative indentation of text.

**Highlighting:**

• **Current line** - If selected, Code Editor highlights the line that contains the cursor.

• **Brace Match** - If selected, Code Editor highlights both of the enclosing glyphs in a set when the cursor is immediately to the right of one of the glyphs. The glyphs involved (parentheses, curly braces, square brackets, or angle brackets) are language-dependent. With Genero BDL, parentheses and square brackets are highlighted, while curly braces denote comments and are not highlighted.

• **All occurrences on double-click** - If selected, Code Editor highlights all occurrences of a selected word after the user double-clicks on it. The search is case-insensitive and matches whole words only.

**Wrapping:**

• **Line wrapping** - If selected, line wrapping is enabled.

**Language Specific preferences tab**

The **Language Specific** tab is only enabled when a language has specific options to set. It is never enabled for **All Languages**. If you select a language and the tab is not enabled, then there are no language-specific preferences to set.

For example, if you set the language to **Genero BDL**, you have these options:

• **Automatically convert all keywords to upper case**

• **Select comment operator** - Use -- or # to delineate comments from code.

• **Synchronize with current line** - If selected, the Structure view synchronizes with cursor movements in the Code Editor, to show the current function.

• **Extra identifiers** - Specify custom keywords into one or more sets of extra identifiers, then set the style for each extra identifier set on the **Color & Font preferences** on page 461 page.

**Related tasks**

Set the color and font of identifiers on page 456
Create a custom keyword set for keywords you want to highlight using a specific font and color in a Code Editor document.

**Color & Font preferences**

Set preferences for the behavior and display aspects of the Code Editor.

You can specify how specific styles of your Editor code appear.

**Global Style - applies to all languages**

• **Theme** - quickly configure the display using predefined color combinations

• **Font family** - specify the font family; when this is changed it applies to all languages, except those for which custom settings have been created.

• **Font size** - specify font size, when this is changed it applies to all languages, except those for which custom settings have been created.
Style Specific Settings (custom settings) for all Languages

**Language** - the default selection **All Languages** in the combobox displays the default styles and properties applicable to all languages. As a default, specific language pages inherit the settings defined for **All languages**.

**Style** - list of styles. Select a **style** to display these properties:

- **Color** - specifies the color for the **Foreground** and **Background** of the style, when this is changed it applies to all languages.
- **Font** - specifies the font selected from the available screen fonts installed on your system. **Use custom font for the selected style** must be checked in order to make changes in:
  - **Family** - font family associated with your selected font; changes will override the global setting.
  - **Size** - font sizes associated with your selected font. changes will override the global setting.
- **Bold** - specifies whether the selected font is Bold. Inherits the global family and size.
- **Italic** - specifies whether the selected font is Italic. Inherits the global family and size.

Style Specific Settings (custom settings) for a specific Language

**Language** - list of languages. Select the **language** from the dropdown list, to display the language-specific styles and properties.

**Use custom settings** - this checkbox displays when you select a specific language in the list. This box must be checked in order for you to make any changes in the settings for the selected language. When this box is checked, the settings for the selected language become specific, and it does not inherit the **All languages** settings any longer.

- To change the **Color** (**Background**, **Foreground**) or the font properties **Bold** or **Italic** for a specific style, you must first select the **Style** from the Style list. **Use custom settings** must be checked.
- To change the **Font Family** and **Size** properties for a specific style, you must check **Use custom font for the selected style**. This overrides the Global Font family and Font size settings.

**Related tasks**

- **Set the color and font of identifiers** on page 456
- Create a custom keyword set for keywords you want to highlight using a specific font and color in a Code Editor document.

**Template preferences**

Set preferences for the template aspects of the Code Editor.

**Language** - select the desired language. The default is Genero BDL.

**Templates**

Displays a list containing the **Name** and **Description** of existing templates. When an existing template is selected, the corresponding template code is displayed in **Preview**.

Icons allow you to:

- **Add** - Displays the **Add Code Template** dialog box, allowing you to enter the new template name and description
- **Delete** - Deletes the selected template
- **Edit** - Displays the **Edit Code Template** dialog box, allowing you to edit the name and description of the selected template.

**Add/Edit a Template:**

- For new templates, enter the template name and description.
- Type directly in the pane to add the code for a new template or to make changes in existing code.
- Use the pipe character '|', to denote the cursor position in the expanded template. For example, when you insert this template into your code, the cursor appears inside the parenthesis after the **CASE** keyword.

```plaintext
CASE ( | )
WHEN ()
```
| Code Editor | 463 |

**Important:** When you modify or create a template, you must type it exactly. For example, you must use indents.

**Preview** - Displays the code for the selected template.

**Import/Export buttons**

- **Import** - import template definitions from an XML file
- **Export** - export template definitions to an XML file

**XML Schema/DTD preferences**

Set preferences for the XML schema and DTD aspects of the Code Editor.

Many XML documents contain external links to stylesheets, schemas, DTDs, and so on, which may be stored at remote locations. XML catalog files may be used to provide an alternative path when the external entities aren't accessible.

Use **Tools>>Preferences** and select **XML Schema/DTD** from the Code Editor page in the Pages tree. You can now see a list of the XML catalog files that are provided.
XML Schema/DTD Configuration

Catalog Entries

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Key Type</th>
<th>Key Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.4js.com/ns/gst/2.40/agmodel.xsd">http://www.4js.com/ns/gst/2.40/agmodel.xsd</a></td>
<td>System Id</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td><a href="http://www.4js.com/ns/gas/2.40/xml.xsd">http://www.4js.com/ns/gas/2.40/xml.xsd</a></td>
<td>System Id</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td><a href="http://www.4js.com/ns/gas/2.40/cfextws.xsd">http://www.4js.com/ns/gas/2.40/cfextws.xsd</a></td>
<td>System Id</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td><a href="http://www.4js.com/ns/gas/2.40/cfextwa.xsd">http://www.4js.com/ns/gas/2.40/cfextwa.xsd</a></td>
<td>System Id</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td><a href="http://www.4js.com/ns/gas/2.40/cfes.xsd">http://www.4js.com/ns/gas/2.40/cfes.xsd</a></td>
<td>System Id</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td><a href="http://schemas.xmlsoap.org/wsd/soap/">http://schemas.xmlsoap.org/wsd/soap/</a></td>
<td>URI</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td><a href="http://schemas.xmlsoap.org/wsd/mime/">http://schemas.xmlsoap.org/wsd/mime/</a></td>
<td>URI</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td><a href="http://schemas.xmlsoap.org/wsd/http/">http://schemas.xmlsoap.org/wsd/http/</a></td>
<td>URI</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td><a href="http://schemas.xmlsoap.org/wsd/">http://schemas.xmlsoap.org/wsd/</a></td>
<td>URI</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td><a href="http://schemas.xmlsoap.org/soap/">http://schemas.xmlsoap.org/soap/</a></td>
<td>URI</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td>-/W3C/DTD XHTML Basic 1.0/EN Public Id</td>
<td>Public Id</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td>-/W3C/DTD XHTML 1.1/EN Public Id</td>
<td>Public Id</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td>-/W3C/DTD XHTML 1.0 Transitional/EN Public Id</td>
<td>Public Id</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
<tr>
<td>-/W3C/DTD XHTML 1.0 Strict/EN Public Id</td>
<td>Public Id</td>
<td>C:\Program Files (x86)\FourJs\Genero \genero-config.xml</td>
</tr>
</tbody>
</table>

Additional Catalog

Catalog File

- C:\Program Files (x86)\FourJs\Genero Evaluation Program 2.40.08\gst\conf\catalog-gst-defaults.xml
- C:\Program Files (x86)\FourJs\Genero Evaluation Program 2.40.08\gas\etc\catalog

Figure 233: XML Schema / DTD Configuration

Catalog Entries

This section displays the entries for the XML Catalog files that are provided.

Use the icons at the top of the Catalog Entries section to add/delete or edit catalog entries. An edit icon appears next to catalog entries that may be edited.

These values are displayed for each Catalog Entry:

- **Key name** - can be empty; already existing names can not be used.
- **Key type** - valid values are:
  - Public ID, for the DTD
  - System ID, for the DTD
  - Schema URI, for the Schema location
• **Key value**
  - For **Public/System IDs** - the absolute file location for the DTD
  - For **Schema URI** - the XSD file location; a Browse button is provided to locate the grammar file.

**Additional Catalog**

Entries in this section specify an XML catalog file to be added to the end of the current catalog. This allows one catalog to refer to another. If a reference cannot be resolved in the current catalog entry file, then the display moves to the next catalog specified in the Additional Catalogs section.

Entries defined in an **XML Catalog file** are given preference over entries that come from a Next Catalog file. This allows you to override the entries in the Next Catalog file to resolve external grammar files.

Use the icons at the top of this section to add/edit/delete Additional Catalog entries.

• **Next Catalog File** - specifies the name of the additional XML catalog fie.

See **XML Catalog files** on page 457 for additional information about Catalog files.

**Customize Diff tool preferences**

Set preferences for the Diff tool. For example, you can set the tool to ignore case or white space, or specify the color used.

Select **Tools > Preferences > Code Editor > Diff Preferences** to set preferences for the Diff tool.

**Internal tool configuration**

Select the checkbox to enable an option:

- **Ignore case**
  - Ignore differences between uppercase and lowercase.

- **Ignore all white space**
  - Ignore white space, including tab, vertical tab, form feed, carriage return, and space. For example, the following are treated as the same:
    ```
    LET monthValue=month(dateValue)
    LET   monthValue=month(dateValue)
    ```

- **Ignore space changes**
  - Ignore internal and trailing white spaces, but not spaces at the end.
    ```
    LET monthValue=month(dateValue)
    ```
    Is the same as:
    ```
    LET   monthValue=month(dateValue)
    ```
    But not the same as:
    ```
    LET monthValue=month(dateValue)
    ```

- **Try hard to find a smaller set of changes**
  - Show a smaller set of differences. You might want to select this option if your Diff output is too large, although it might cause Diff to run more slowly.
**External tool configuration**

If you have chosen to use an external tool, enter the configuration information:

- **External tool command line**
  - The path of the external tool

- **External tool arguments**
  - Use the shown variables to specify the arguments for the command line. The names of the files will replace the variables when the tool is invoked.

**Color**

Select the color button next to each option to change the color used for deleted, modified, added lines and placeholders.

**Related concepts**

- Setting Preferences on page 128
  - Customize Genero Studio to meet your needs.

**Views**

Information about Code Editor views.

- Code Editor basics on page 445
- Code Structure view on page 466
- Output view
- Search/Replace view on page 467
- Search Results view on page 471
- Bookmarks view on page 117

**Note:** For SVN-related views, see Source Code Management (SCM) on page 624.

**Code Structure view**

If the file being edited is a Genero source code file (.4gl), its structure is displayed in a tree.

![Figure 234: Code Structure view](image)

The view is organized by folders.
### Table 144: Code Structure view folders

<table>
<thead>
<tr>
<th>Folder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TODOs</td>
<td>Comments in the source code that start with <code>--TODO</code>.</td>
</tr>
<tr>
<td>Bugs</td>
<td>Comments in the source code that start with <code>--BUG</code>.</td>
</tr>
<tr>
<td>SQL</td>
<td>The program blocks and functions, together with their variables.</td>
</tr>
<tr>
<td>Module</td>
<td>Prepared SQL statement and cursor names.</td>
</tr>
<tr>
<td>Globals</td>
<td>Global variables, including records</td>
</tr>
<tr>
<td>Externals</td>
<td>Functions from another 4gl file that are called in the active file</td>
</tr>
</tbody>
</table>

![Figure 235: Example of a TODO](image)

Click an object in the tree to display the corresponding lines in the Document window.

**Tip:** A lock icon signifies that a function or variable is private.

The functions listed in the structure are automatically displayed in the order in which they appear in the file. Right-click in the Code Structure view to access options for reorganizing the structure.

- **Reset to File Order:** The default is file order, or the order in which the functions and variables appear within the file.
- **Sort Ascending:** List the folders, the functions, and the variables within the functions ascending in alphabetical order.
- **Sort Descending:** List the folders, the functions, and the variables within the functions descending in alphabetical order.

**Tip:** Click on the small arrow located under the window's title bar to change the sort to ascending or descending.

**Search/Replace view**

The Search/Replace view allows you to search for and replace text in one or multiple documents.

The **Window > Views > Search/Replace** menu option opens the Search/Replace View.
**Figure 236: Search/Replace View**

**Search/Replace Fields**

Fields allow you to set the search / replace criteria:

- **Find**: Enter a search string, or click the down arrow next to the input box to select from a list of previously entered strings.

- **Replace**: Enter a string to replace the string specified in the **Find** field. Leave this field empty if you do not wish to replace the found text. You can enter the replacement string, or click the down arrow next to the input box to select from a list of previously entered replacement strings. To replace the text with a space, type a space in this box.

- **In**: Select the scope of the search/replace:
  - **Current file**: Search in the current file only.
  - **Open files**: Search in all open files.
  - **Folder**: Search within a specified folder. You specify the folder, as well as some additional options, in the **Folder** field. Selecting **Folder** has the same effect as the **Find in Files** menu option.
  - **Current project**: Search in all files of the current project.

- **File types**: Limit your search and replace to specific file types, specified by *
  ext* where ext is a file extension.

- **Folder**: Limit your search results to the contents of a specified folder. You can use the magnifying glass icon to specify the folder. Within this folder, you can use the check boxes to identify whether to include the subfolders of the selected folder, and whether to include hidden files.

  **Note**: The **Folder** combobox is enabled when **Folder** is selected for the **In** field.

**Search/Replace Options**

A series of checkboxes allow you to further refine your search and replace options.
Case Sensitive
When selected, differentiates between uppercase and lowercase when performing a search.

Match whole word only
When selected, will not find and/or replace the string if it is contained within another string.

Use
When selected, you can choose to use Regular Expressions or Wildcards (* and?).

Note: If Use is checked, the Match whole word only option can not be used and is disabled.

Wrap around
When selected, if the search reaches the end of the file, start at the top of the document to find the next occurrence.

Incremental search
When selected, stop at each occurrence of the string

Restrict to selection
When selected, if a section of a document is selected, restricts the search to that section.

Text search
When selected, the string is searched for in the text file. When not selected, the string is searched for in the formatted Genero file.

A formatted Genero file refers to a file that displays graphically, rather than text-based. For example, when you open a .4fd file in Form Designer, you are shown a graphical view of the form. This is referred to as a formatted Genero file.

To demonstrate:

1. Open the HelloWorld demo project.
2. In the Search/Replace view:
   - Type "gr1" in the Find field.
   - Select "Current project" from the In combo box.
   - Ensure Text search is not selected.
   - Click the Find all toolbar icon.

   In the Search Results view, a single result displays for the HelloForm.4rd file.
3. Double-click on the entry.
   - The graphical view of HelloForm.4rd opens in Form Designer.
   - The searched-for string is highlighted in the Properties view. In this example, it is the value of the name property for the Grid object.

4. In the Search Results view, click the Clear all search results toolbar icon.
5. Modify your options in the Search/Replace view:
   - Select the Text Search checkbox.
   - Click the Find all toolbar icon.

   In the Search Results view, a single result displays for the HelloForm.4rd file. The listing for the result, however, is based on the XML representation of the .4fd file.
6. Double-click on the entry.
• The XML view of HelloForm.4rd opens in Form Designer.
• The searched-for string is highlighted in the XML (text-only) file.

Tip: The Text Search option works on unsaved text files, but only works on the saved formatted Genero files, as the underlying text/xml file is only generated when the formatted Genero file is saved to disk.

Search/Replace Toolbar

In the view's toolbar, icon buttons allow you to control the direction of the search and the implementation of the replace. The icons in order (left to right):

• Find previous
• Find next
• Find all
• Replace previous
• Replace next
• Replace all
• Stop

The Edit > Search/Replace menu

The Edit > Search/Replace provides a menu interface that allows you to perform many of the same tasks allow you to find/replace text in documents, as well as clearing the Search Results view.

• Edit > Search/Replace > Find (CTRL + F) opens the Search/Replace view.
• Edit > Search/Replace > Find in Files (CTRL + SHIFT + F) opens the Search/Replace view with Folders selected for the In field.

Escaping the dialog

When in the Search/Replace dialog, press the ESC key to return focus to the top-most file opened in the central work area.
Search Results view

Search results are displayed in the Search Results view.

Figure 237: Search results View

The Locator column displays the Search results using a tree view, with three levels:

- The uppermost node provides a summary of the search criteria, as well as the number of results found.
- The first child node represents a file. Each file included in the search gets its own child node. The number of results from within that file are displayed.
- The leaves of the tree view are the individual search results. The format of the locator depends on the searched file. For text files, each result is marked by four numbers: The line where the match starts, the character position where the match starts, the line where the match ends, and the character position where the match ends. For other types of files, it depends on the module used to open the file; the locator is usually an XPath to the model node, or a model node identifier. It will be the same as the locator in the Document Errors view.

The Match column provides the detail of the result in the context of the line from the file, with the search term highlighted.

If more than 10,000 results are found, search is stopped and a message is displayed on the summary line.

Double-click the highlighted text to display the string in its corresponding file.

Tasks view

The Tasks view displays a list of the current tasks started by Genero Studio. From this view, you can stop a task.

In addition to listing running applications, the Tasks view displays a task for the Genero Desktop Client or the Genero Application Server when they are started by Genero Studio.

To stop a running task:

1. Select the task row.
2. Click the Abort toolbar icon.

All running tasks are stopped when you exit Genero Studio.
**Document Errors view**
The Document Errors view displays errors related to a document.

- Select the error number and press the F1 key to display additional information about the error.
- Use **Tools>>Preferences, Genero Studio Preferences, Messages** to hide a specific information or warning message.
- View **BUG** or **TODO** notations found in your code. Enter the notation into your code files with `--KEYWORD <message>` where **KEYWORD** is **BUG** or **TODO**. After compiling, put focus in the Project Manager view to see these messages in the Document Errors tab.

**Output view**
The Output view displays messages related to the output and errors specific to the process being performed.

Click **Show as List** to toggle between text display and list display.

With a list display, you can filter the items:

- The **Channel** drop-down list filters by process.
- Selecting icons filters by severity. Each icon acts as a toggle switch. You can have more than one message type selected.

### Table 145: Message icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Error Icon" /></td>
<td>Select this icon to show/hide error messages.</td>
</tr>
<tr>
<td><img src="image" alt="Warning Icon" /></td>
<td>Select this icon to show/hide warning messages.</td>
</tr>
<tr>
<td><img src="image" alt="Info Icon" /></td>
<td>Select this icon to show/hide information messages.</td>
</tr>
</tbody>
</table>

**Keyboard Shortcuts**
Information about Code Editor keymaps.

- **Cursor movement keymap** on page 473
- **Selection keymap** on page 473
- **Text editing keymap** on page 474
- **Clipboard keymap** on page 475
- **Search and replace keymap** on page 475
- **Buffers and Files keymap** on page 475
- **Code completion keymap** on page 476
- **Code templates keymap** on page 476

**Note:** Genero Studio default accelerator keys may be customized. See **Accelerators** in **Tools>>Preferences, User Interface**.
### cursor movement keymap

Keyboard shortcuts for moving the cursor.

**Table 146: Cursor movement keymap**

<table>
<thead>
<tr>
<th>Command / Action</th>
<th>Keyboard shortcuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left one character</td>
<td>Left arrow</td>
</tr>
<tr>
<td>Right one character</td>
<td>Right arrow</td>
</tr>
<tr>
<td>Left one word</td>
<td>Ctrl + Left arrow</td>
</tr>
<tr>
<td>Right one word</td>
<td>Ctrl + Right arrow</td>
</tr>
<tr>
<td>Up one line</td>
<td>Up arrow</td>
</tr>
<tr>
<td>Down one line</td>
<td>Down arrow</td>
</tr>
<tr>
<td>Beginning of line</td>
<td>Home</td>
</tr>
<tr>
<td>End of line</td>
<td>End</td>
</tr>
<tr>
<td>Last index of current line</td>
<td>Alt + End</td>
</tr>
<tr>
<td>First index of current line</td>
<td>Alt + Home</td>
</tr>
<tr>
<td>Scroll window up one line</td>
<td>Ctrl + Up arrow</td>
</tr>
<tr>
<td>Scroll window down one line</td>
<td>Ctrl + Down arrow</td>
</tr>
<tr>
<td>Up one screen</td>
<td>Page Up</td>
</tr>
<tr>
<td>Down one screen</td>
<td>Page Down</td>
</tr>
<tr>
<td>Top of file</td>
<td>Ctrl + Home</td>
</tr>
<tr>
<td>Bottom of file</td>
<td>Ctrl + End</td>
</tr>
<tr>
<td>Next tab stop</td>
<td>Tab</td>
</tr>
<tr>
<td>Previous tab stop</td>
<td>Shift + Tab</td>
</tr>
<tr>
<td>Go to line</td>
<td>Ctrl + G</td>
</tr>
<tr>
<td>Find matching brace, bracket or parenthesis</td>
<td>Click one brace to highlight matching braces</td>
</tr>
<tr>
<td>Smart backspace</td>
<td>Shift + Backspace</td>
</tr>
</tbody>
</table>

### selection keymap

Keyboard shortcuts for form selection.

**Table 147: Selection keymap**

<table>
<thead>
<tr>
<th>Command / Action</th>
<th>Keyboard shortcuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select left one character</td>
<td>Shift + Left arrow</td>
</tr>
<tr>
<td>Select right one character</td>
<td>Shift + Right arrow</td>
</tr>
<tr>
<td>Select current word</td>
<td>Ctrl + L, W</td>
</tr>
<tr>
<td>Select to start of current word</td>
<td>Ctrl + Shift + Left arrow</td>
</tr>
<tr>
<td>Select to end of current word</td>
<td>Ctrl + Shift + Right arrow</td>
</tr>
<tr>
<td>Select current line</td>
<td>Ctrl + L, L</td>
</tr>
<tr>
<td>Select to start of line</td>
<td>Shift + Home</td>
</tr>
<tr>
<td>Command / Action</td>
<td>Keyboard shortcuts</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>Select to end of line</td>
<td>Shift + End</td>
</tr>
<tr>
<td>Select up one line</td>
<td>Shift + Up arrow</td>
</tr>
<tr>
<td>Select down one line</td>
<td>Shift + Down arrow</td>
</tr>
<tr>
<td>Select to top of window</td>
<td>Ctrl + Shift + Page Up</td>
</tr>
<tr>
<td>Select to bottom of window</td>
<td>Ctrl + Shift + Page Down</td>
</tr>
<tr>
<td>Select up one screen</td>
<td>Shift + Page Up</td>
</tr>
<tr>
<td>Select down one screen</td>
<td>Shift + Page Down</td>
</tr>
<tr>
<td>Select to top of file</td>
<td>Ctrl + Shift + Home</td>
</tr>
<tr>
<td>Select to bottom of file</td>
<td>Ctrl + Shift + End</td>
</tr>
<tr>
<td>Select all</td>
<td>Ctrl + A</td>
</tr>
<tr>
<td>Square selection</td>
<td>Alt + mouse (left)</td>
</tr>
<tr>
<td>Extend square selection to previous character</td>
<td>Alt + Shift + Left arrow</td>
</tr>
<tr>
<td>Extend square selection to next character</td>
<td>Alt + Shift + Right arrow</td>
</tr>
<tr>
<td>Extend square selection to end of current line</td>
<td>Alt + Shift + End</td>
</tr>
<tr>
<td>Extend square selection to start of current line</td>
<td>Alt + Shift + Home</td>
</tr>
</tbody>
</table>

**Text editing keymap**

Keyboard shortcuts for editing text.

**Table 148: Editing Text keymap**

<table>
<thead>
<tr>
<th>Command / Action</th>
<th>Keyboard shortcuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toggle Insert / Overstrike mode</td>
<td>Insert</td>
</tr>
<tr>
<td>Comment (comments all selected lines)</td>
<td>Ctrl + K, Ctrl + K</td>
</tr>
<tr>
<td>Block comment</td>
<td>Ctrl + K, Ctrl + C</td>
</tr>
<tr>
<td>Unblock comment</td>
<td>Ctrl + K, Ctrl + U</td>
</tr>
<tr>
<td>Delete character/selection</td>
<td>Delete</td>
</tr>
<tr>
<td>Delete previous character/selection</td>
<td>Backspace</td>
</tr>
<tr>
<td>Delete to end of word (kill-word)</td>
<td>Ctrl + D</td>
</tr>
<tr>
<td>Delete to start of word (backward-kill-word)</td>
<td>Ctrl + Backspace</td>
</tr>
<tr>
<td>Delete line (kill-line)</td>
<td>Ctrl + Del</td>
</tr>
<tr>
<td>Delete to end of line</td>
<td>Ctrl + Shift + Y</td>
</tr>
<tr>
<td>Indent block</td>
<td>Tab</td>
</tr>
<tr>
<td>Un-indent block</td>
<td>Shift + Tab</td>
</tr>
<tr>
<td>Lowercase the currently selected text</td>
<td>Alt + Shift + L</td>
</tr>
<tr>
<td>Uppercase the currently selected text</td>
<td>Alt + Shift + U</td>
</tr>
<tr>
<td>Toggle case of each character in the selected text</td>
<td>Ctrl + U</td>
</tr>
<tr>
<td>Lowercase word</td>
<td>Ctrl + Alt + W</td>
</tr>
<tr>
<td>Command / Action</td>
<td>Keyboard shortcuts</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Uppercase word</td>
<td>Ctrl + Alt + Shift + W</td>
</tr>
<tr>
<td>Undo</td>
<td>Ctrl + Z</td>
</tr>
<tr>
<td></td>
<td>Alt + Backspace</td>
</tr>
<tr>
<td>Redo</td>
<td>Ctrl + Y</td>
</tr>
<tr>
<td></td>
<td>Alt + Shift + Backspace</td>
</tr>
<tr>
<td>Insert return</td>
<td>Enter Shift + Enter</td>
</tr>
</tbody>
</table>

**Clipboard keymap**
Keyboard shortcuts for the clipboard.

**Table 149: Clipboard keymap**

<table>
<thead>
<tr>
<th>Command / Action</th>
<th>Keyboard shortcuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut selection</td>
<td>Ctrl + X, Shift + Delete, Cut</td>
</tr>
<tr>
<td>Copy selection</td>
<td>Ctrl + C, Ctrl + Insert, Copy</td>
</tr>
<tr>
<td>Paste from clipboard</td>
<td>Ctrl + V, Shift + Insert, Paste</td>
</tr>
</tbody>
</table>

**Search and replace keymap**
Keyboard shortcuts for search and replace.

**Table 150: Search and Replace keymap**

<table>
<thead>
<tr>
<th>Command / Action</th>
<th>Keyboard shortcuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td>Ctrl + F</td>
</tr>
<tr>
<td>Find next</td>
<td>F3</td>
</tr>
<tr>
<td>Find previous</td>
<td>Shift + F3</td>
</tr>
<tr>
<td>Find in files</td>
<td>Ctrl + Shift + F</td>
</tr>
<tr>
<td>Focus back to file</td>
<td>Esc</td>
</tr>
</tbody>
</table>

**Buffers and Files keymap**
Keyboard shortcuts for managing buffers and files.

**Table 151: Buffers and Files keymap**

<table>
<thead>
<tr>
<th>Command / Action</th>
<th>Keyboard shortcuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>New</td>
</tr>
<tr>
<td>File</td>
<td>Open</td>
</tr>
<tr>
<td>File</td>
<td>Print</td>
</tr>
<tr>
<td>File</td>
<td>Save All</td>
</tr>
<tr>
<td>File</td>
<td>Save</td>
</tr>
<tr>
<td>File</td>
<td>Save As</td>
</tr>
<tr>
<td>File</td>
<td>Close</td>
</tr>
<tr>
<td>Command / Action</td>
<td>Keyboard shortcuts</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Close Genero Studio</td>
<td>Ctrl + Shift + Q</td>
</tr>
<tr>
<td>Split window vertically</td>
<td>Ctrl + Alt + V</td>
</tr>
<tr>
<td>Split window horizontally</td>
<td>Ctrl + Alt + H</td>
</tr>
<tr>
<td>Unsplit window</td>
<td>Ctrl + Alt + Shift + N</td>
</tr>
<tr>
<td>Unsplit all</td>
<td>Ctrl + Alt + Shift + A</td>
</tr>
<tr>
<td>Preview view</td>
<td>Alt + Shift + P</td>
</tr>
</tbody>
</table>

### Code completion keymap
Keyboard shortcuts for code completion.

#### Table 152: Code Completion keymap

<table>
<thead>
<tr>
<th>Command / Action</th>
<th>Keyboard shortcuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Complete</td>
<td>Ctrl + Space</td>
</tr>
</tbody>
</table>

**Note:** Code Completion is for 4gl files only.

### Code templates keymap
Keyboard shortcuts for accessing code templates.

#### Table 153: Code Templates keymap

<table>
<thead>
<tr>
<th>Command / Action</th>
<th>Keyboard shortcuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Templates</td>
<td>Ctrl + T</td>
</tr>
</tbody>
</table>

### vi Commands List
A listing of vi commands currently implemented in Code Editor.

#### Table 154: Cursor movement

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>h l k j</td>
<td>character left, right; line up, down</td>
</tr>
<tr>
<td>b w</td>
<td>word/token left, right</td>
</tr>
<tr>
<td>ge e</td>
<td>end of word/token left, right</td>
</tr>
<tr>
<td>0 ^ $</td>
<td>beginning, first, last character of line</td>
</tr>
<tr>
<td>nG/:n ngg</td>
<td>line n, default the last, first</td>
</tr>
<tr>
<td>B W</td>
<td>space-separated word left, right</td>
</tr>
<tr>
<td>gE E</td>
<td>end of space-separated word left, right</td>
</tr>
<tr>
<td>H M L</td>
<td>Top, middle, bottom of screen</td>
</tr>
<tr>
<td>g0 gm</td>
<td>beginning, middle of screen line</td>
</tr>
<tr>
<td>g^ g$</td>
<td>first, last character of screen line</td>
</tr>
<tr>
<td>fc Fc</td>
<td>next, previous occurrence of character c</td>
</tr>
<tr>
<td>tc Tc</td>
<td>before next, previous occurrence of c</td>
</tr>
</tbody>
</table>
### Table 155: Insertion and Replace (insert mode)

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i a</td>
<td>insert before, after cursor</td>
</tr>
<tr>
<td>I A</td>
<td>insert at beginning of first character, end of line</td>
</tr>
<tr>
<td>gI (g + Capital I)</td>
<td>insert text in first column/go to first column</td>
</tr>
<tr>
<td>o o</td>
<td>open a new line below, above the current line</td>
</tr>
<tr>
<td>rc</td>
<td>replace character under cursor with c</td>
</tr>
<tr>
<td>R</td>
<td>replace characters starting at the cursor</td>
</tr>
<tr>
<td>cm</td>
<td>change text of movement command m</td>
</tr>
<tr>
<td>cc or S</td>
<td>change current line</td>
</tr>
<tr>
<td>C</td>
<td>change to the end of line</td>
</tr>
</tbody>
</table>

### Table 156: Deletion

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x X</td>
<td>delete character under (right), before (left) cursor</td>
</tr>
<tr>
<td>dm</td>
<td>delete text of movement command m</td>
</tr>
<tr>
<td>dd D</td>
<td>delete current line, to the end of line</td>
</tr>
<tr>
<td>J gJ</td>
<td>join current line with next, without space</td>
</tr>
<tr>
<td>:rd</td>
<td>delete range r lines</td>
</tr>
</tbody>
</table>

### Table 157: Search and Substitution

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/s# ?s#</td>
<td>search forward, backward for s</td>
</tr>
<tr>
<td>n or /#</td>
<td>repeat forward last search</td>
</tr>
<tr>
<td>N or ?#</td>
<td>repeat backward last search</td>
</tr>
<tr>
<td># *</td>
<td>search backward, forward for complete word under cursor</td>
</tr>
<tr>
<td>g# g*</td>
<td>same, but also find partial matches</td>
</tr>
<tr>
<td>gd gD</td>
<td>local, global definition of symbol under cursor</td>
</tr>
<tr>
<td>:rs/f /t/x</td>
<td>substitute f by t in range r; x: g -- all occurrences, c -- confirm changes</td>
</tr>
</tbody>
</table>

### Table 158: Copying

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ym</td>
<td>yank the text of movement command m</td>
</tr>
<tr>
<td>yy/:y or Y</td>
<td>yank current line into register</td>
</tr>
<tr>
<td>:ry</td>
<td>yank r range of lines</td>
</tr>
<tr>
<td>p P</td>
<td>put register after, before cursor position</td>
</tr>
</tbody>
</table>
Table 159: Undoing, Repeating and Registers

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u U</td>
<td>undo last command, restore last changed line</td>
</tr>
<tr>
<td>.</td>
<td>repeat last changes</td>
</tr>
<tr>
<td>.n</td>
<td>repeat last changes with count replaced by n</td>
</tr>
</tbody>
</table>

Table 160: Standard Mode Formatting / Filtering

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>~</td>
<td>switch case and advance cursor</td>
</tr>
<tr>
<td>g~m gum gUm</td>
<td>switch case, lc, uc on movement m</td>
</tr>
</tbody>
</table>

Table 161: Insert Mode

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>esc or ^]</td>
<td>abandon editing # command mode</td>
</tr>
</tbody>
</table>

Table 162: Marks, Motions, and Tags

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mc</td>
<td>mark current position with mark c # [a Z]</td>
</tr>
<tr>
<td>'c 'C</td>
<td>go to mark c in current, C in any file</td>
</tr>
</tbody>
</table>

Table 163: Misc IO related commands

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:e f</td>
<td>edit file f, reload current file if no f</td>
</tr>
<tr>
<td>:rw f</td>
<td>write range r to file f (current file if no f) (:w, :w f)</td>
</tr>
<tr>
<td>:q :q!</td>
<td>quit and confirm, quit and discard changes</td>
</tr>
<tr>
<td>:wq or :x or ZZ</td>
<td>write to current file and exit</td>
</tr>
<tr>
<td>:r f</td>
<td>insert content of file f below cursor</td>
</tr>
<tr>
<td>:n</td>
<td>next file</td>
</tr>
<tr>
<td>:p</td>
<td>previous file</td>
</tr>
<tr>
<td>:n,kw &gt;&gt; file1</td>
<td>append lines n-k into another file file1.</td>
</tr>
</tbody>
</table>

Table 164: Ranges

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>, ;</td>
<td>separates two line numbers, set to first line</td>
</tr>
<tr>
<td>:n,m</td>
<td>lines n to m</td>
</tr>
<tr>
<td>n</td>
<td>an absolute line number n</td>
</tr>
<tr>
<td>. $</td>
<td>the current line, the last line in file</td>
</tr>
<tr>
<td>Keystrokes</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>% *</td>
<td>entire file, visual area (exclude)</td>
</tr>
<tr>
<td>'t</td>
<td>position of mark t</td>
</tr>
<tr>
<td>/p/ ?p?</td>
<td>the next, previous line where p matches</td>
</tr>
<tr>
<td>+n -n</td>
<td>+n, #n to the preceding line number</td>
</tr>
</tbody>
</table>

Table 165: Folding

<table>
<thead>
<tr>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zo zc zO zC</td>
<td>open, close one fold; recursively</td>
</tr>
<tr>
<td>[z ]z</td>
<td>move to start, end of current open fold</td>
</tr>
<tr>
<td>zj zk</td>
<td>move down/up to start/end of next/previous fold</td>
</tr>
<tr>
<td>zm zM</td>
<td>fold more, close all folds</td>
</tr>
<tr>
<td>zr zR</td>
<td>fold less, open all folds</td>
</tr>
</tbody>
</table>

Code Editor error messages

A list of Code Editor error messages. For messages that are not self-explanatory, additional information is provided.

Table 166: Code Editor Error messages

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
</table>
| GS-15100 | File filename not found, unable to run the program."
          | A Quick Run task was started but the task cannot find the file. |
| GS-15101 | Failed to run the program. |
          | A Quick Run task was unable to start because either: |
          | • The file could not be compiled. |
          | • The compilation of the file failed. |
          | • The file resulting from the compilation could not be found. |
| GS-15110 | *** Quick run filename started *** |
          | Error message and resolution should be self-explanatory. |
| GS-15111 | *** Execution of filename finished. Exit code: code *** |
          | Error message and resolution should be self-explanatory. |
| GS-15112 | *** Execution of filename aborted *** |
          | Error message and resolution should be self-explanatory. |
GUI Testing

The automation of regression testing relies on the Genero Ghost Client (GGC), a tool that is provided as part of the Genero Business Development Language package.

From Genero Studio, you create a regression test by executing an application and recording what it does. It will generate a .4gl test scenario. You can run the application and record the log in both local and remote configurations. You can record against all front ends: the Genero Desktop Client (GDC), the Genero Browser Client (GBC), and both of the Genero Mobile clients (GMA and GMI).

Note: Recording a test against a mobile client (GMA or GMI) must be done in development mode. The Genero BDL code is executed on the local machine.

Quick Start: Regression testing

This quick start uses the OfficeStore demo to create a regression test for the Account application.

1. Open the OfficeStore demo.
2. Right-click on the Accounts application node and select Create test. The application launches, using the front-end specified by the Genero Studio configuration.
3. Use the application to add a new account.
   a) Click New.
   b) Enter in the details for a new imaginary user. Do not worry about the validity of the field entries at this time.
   c) Click Accept.
   Genero Studio records the details of your application use.
4. When you are done, select File > Exit to exit the application. Genero Studio creates the test scenario as a .4gl file.
5. Add an application node to your project and name it Test.
6. Set a dependency between the Test node and the node of the application being tested.
   a) Right-click on the Test node and select Advanced Properties.
   b) Select the Dependencies page.
   c) Select the application being tested and click Apply.
   d) Click OK to close the dialog.
7. Use the File > Save as ... option to save the .4gl test scenario file as a child of the Test node.
   For this quick start, name the test myAccountTest.4gl and save it in the tests subdirectory of the OfficeStore project.
8. Execute the Test application node.

The test executes according to the instructions in the test scenario file. The results appear in the Output view. Edit the test scenario to add or remove steps from the test. See the Genero Ghost Client User Guide for more information.

Related concepts
Groups, Applications, Libraries, and Packages on page 410
Projects are organized into Group, Application, Library, and Package nodes. The Projects view visually displays a project file (4pw) for easy management of project source files.

The testing framework

There are four steps in creating and executing a test.

1. Create the test.

   Create the test for your application by selecting Create test and executing the application. Your app runs locally or remotely, depending on the configuration. After navigating the interface and performing the steps to be tested, quit the application. Internally, a GUILOG is created for the app (local or remote). A .4gl test file is then generated from the GUILOG. The .4gl code is based on the interactions with the application while the log was recording. This includes opening windows, setting field values, executing actions, and so on.

2. Add the test to a project.

   Save the .4gl test file to a project. Create an application node for your test, and add the .4gl test file to the new application node. Right-click on the application test node and select Advanced Properties. Set a dependency with the application being tested.

   Your project can contain only the single test, or you can create a project to serve as a test suite inclusive of multiple tests. If you create a test suite, you must also create an application to call the generated tests.

3. Edit the test and add assertion checks.

   Assertion checking is where you verify that the application works as it did previously, repeatedly.

   When the test is generated, every possible step and test is added to the .4gl file. The generated code usually contains far more checks than what you expect; but this is intended as it is easier to remove things than to write new checks and tests by hand. Update the .4gl file to remove parts of the test that are no longer needed or to add new checks and tests.

   You may need to update initialization and finalization tasks to ensure the data is correct for the test.

4. Run the test.

   Before running the test, ensure that the test application node has a dependency set on the application being tested.

   The app is run as a child process. The test runs the app and plays the role of a GUI client (using GGC) and runs the checks set in the .4gl file.

For specific instructions on completing these tasks, see Create and run a unit test on page 481.

Create and run a unit test

Genero Studio provides a graphical user interface to create tests. It works with Genero Ghost Client.

From Genero Studio, you can:

- Generate a test scenario as a Genero application.
- Edit the scenario to modify the test.
- Add the scenario to a project.
- Run the scenario.

1. Open your Genero project.

2. Right-click on an application node and select Create test.
   
   The application launches, using the front-end specified by the Genero Studio configuration.

3. Use the application to complete the steps to include in your test.
   
   Genero Studio records the details of your application use.

4. When you are done, exit the application.

   Genero Studio creates the test scenario as a .4gl file.
5. Add an application node to your project and name it Test.
   You can use any name for this node; for this procedure we use the name Test.
   Tip: If you already have a Test node, you can skip ahead to step 7. By default, each new project you create has a Test node as part of its project structure.

6. Set a dependency between the Test node and the node of the application being tested.
   a) Right-click on the Test node and select Advanced Properties.
   b) Select the Dependencies page.
   c) Select the application being tested and click Apply.
   d) Click OK to close the dialog.

7. Save the .4gl test scenario file as a child of the Test node.

8. If you made changes to your data during the recording of the test scenario, return the data to its initial state.

9. Execute the Test application node.

The test executes according to the instructions in the test scenario file. The results appear in the Output view.

Edit the test scenario to add or remove steps from the test. See the Genero Ghost Client User Guide for more information.

Related concepts
Groups, Applications, Libraries, and Packages on page 410
Projects are organized into Group, Application, Library, and Package nodes. The Projects view visually displays a project file (.4pw) for easy management of project source files.

Viewing macro definitions
A Genero test application includes macros. You can view the macro definitions using the CTRL and ENTER keys.

If you read through the generated test file, you will notice an include statement specifying the file that contains the definition of the macros:

```plaintext
&include "GGCTestUtilsMacro.4gl"
```

The macros are written in uppercase. In this snippet from the generated test file, two macros can be found:

```plaintext
... 
WAIT_FOR_APPLICATION("app_1484", 1025)  
ASSERT_EQUALS(getWindowName(), "w_orderform")  
...
```

You can quickly view the definition for either macro by:

1. Placing your cursor in the macro keyword (WAIT_FOR_APPLICATION or ASSERT_EQUALS in our example.)
2. Hold down the CTRL key.
3. Press the ENTER key (or click the mouse).

The GGCTestUtilsMacro.4gl file opens in Code Editor, with the selected macro highlighted.

For more information on macros, refer to the Genero Ghost Client User Guide.

Code Analyzer

The Code Analyzer reverse engineers existing applications and can generate diagrams to provide an overview of the application.

- Sequence Diagrams on page 483
Sequence Diagrams

The Sequence Diagram visually displays the flow of your application logic. It shows how the functions of the application call and/or are called by other functions.

Display the Sequence Diagram for a function
To display the Sequence Diagram for a function:

1. Right-click the function name, in either the opened Genero source code (4gl) file (in Code Editor) or in the Code Structure view.
2. Select Open Sequence Diagram.

Tip: If you do not see the menu option, rebuild your application.

Display the Sequence Diagram for the application
To display the Sequence Diagram for an application:

1. Right-click the word MAIN in the MAIN program block in the opened Genero source code (4gl) file (in Code Editor) or in the Code Structure view.
2. Select Open Sequence Diagram.

Tip: If you do not see the menu option, rebuild your application.

In an open Sequence Diagram, the right-click option View source opens the source code module (4gl file) for the function, allowing you to switch back and forth between the source and the diagram.

In the Function calls view of an open Dependency Diagram, the right-click menus for Called Function and Caller Function have an option Open sequence diagram.

The Sequence Diagram

![Sequence Diagram Image]

Figure 238: Sequence Diagram
The starting point of the application is indicated by the stick-figure *actor*, the user who interacts with the application.

- Processes that exist simultaneously are represented by *parallel lines*. For a Genero application, these lines represent the *source code modules* in your application.
- Boxes on the lines represent the *functions* in each module. The sequence is indicated by the order in which the boxes are listed; functions which are called by other functions have their boxes stacked on top. Plus and minus signs on each box allow you to display or hide sub calls.
- Horizontal arrows display the interaction between functions, the messages (*calls*) that are exchanged between them, and the order in which the calls occur.

In this example, the user interacts with the **MAIN** program block (function), which calls the *query_cust* function in the *custquery.4gl* source code module. The *query_cust* function calls other functions in that module in the order indicated. For example, it calls the function *cust_select*, which calls *fetch_cust*. The *query_cust* function returns to the **MAIN**.

### Zoom

Hold down the CTRL key and use the mouse wheel to zoom in and out of the diagram.

### Structure View

The structure of the program modules is shown in a tree in the Structure view. Use the plus/minus signs to display/hide the functions in a module. Select a function in the Structure view to display its properties in the Properties view. Right-click on a node in the Structure view to see a context sensitive menu of options for that node.

### Sub Calls

Right-click on the sequence diagram to display the menu options that will display or hide all the sub calls in a program.

### Customize

Use **Tools > Preferences > Sequence Diagram** to define default maximum number of dependencies; default is 1.

### Dependency Diagrams

The Dependency Diagram displays a graphical view of the complex relationships between the various pieces of a project. It shows the components that depend on other components, and/or have components that depend on them.

**Displaying the Dependency Diagram**

To display a diagram, right-click an Application or Group node, and select **Open Dependency Diagram**.
Figure 239: Dependency Diagram

Zoom
Hold down the CTRL key and use the mouse wheel to zoom in and out of the diagram.

Structure and Function Calls Views
The structure of the project is displayed in a tree in the Structure view. Select a component in the diagram to display its properties in the Properties view. Select the link in the diagram between components to display the associated function calls in the Function calls view.

Right-click menu options in Function calls view:

- Caller function
  - Symbol definition
  - Open call location
  - Open sequence diagram
- Called function
  - Symbol definition
  - Open sequence diagram
- Caller/Called Module
  - View source
Dependency diagram context menu
The Dependency Diagram has a context menu accessible by right-clicking on the diagram or diagram item. The context menu displays actions applicable to the item selected.

Table 167: Right-click Context Menu

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expand/Collapse</td>
<td>Hide or show subcomponents.</td>
</tr>
<tr>
<td>Locate in Project</td>
<td>Bring focus to the component in the project.</td>
</tr>
<tr>
<td>Open Project/Application/Library in new tab</td>
<td>Open component in new tab.</td>
</tr>
<tr>
<td>Open in New Tab</td>
<td>Open diagram in new tab.</td>
</tr>
<tr>
<td>Filter Items ...</td>
<td>The Filter View dialog allows you to hide and show items on a diagram.</td>
</tr>
<tr>
<td>Hide selected item / Show all items</td>
<td>Selectively show or hide items.</td>
</tr>
</tbody>
</table>

Form Designer

*Form Designer* is a visual editor that supports the creation, editing, and layout of Genero forms in Genero Studio.

- Forms in Genero applications on page 486
- Quick Start: Creating a first form on page 487
- Creating the user interface on page 488
- Form Designer usage on page 523
- The gsform command on page 525
- Localizing your form on page 527
- Form Designer Reference on page 528

Forms in Genero applications

Forms are used in Genero applications to interact with the application's users primarily by allowing for input and display of data.

Built-in functionality automatically detects when a user has entered data into a form, or selected a button or other form item. Frequently the data displayed on the form has been retrieved from a database table or will be used to update the table.

A Genero form is designed in a **Form definition file**. In Form Designer, this form definition file has a `.4fd` extension. The form definition file is compiled into an XML file having an extension `.42f`. The `.42f` is used by the Genero client to display the application form to the user.

Existing `.per` forms may be imported and then updated with *Form Designer*.

Program source files (`.4gl`) use high-level Genero BDL instructions to open and display the form and to allow users to query a database or make changes in the rows of a database table, for example.
Quick Start: Creating a first form

This topic will guide you through creating a first form in Form Designer.

The easiest way to create a form for data manipulation is from an existing database.

1. Create a project.
2. Add a meta-schema.
3. Select File > New Form from Database.
4. Use the wizard to create a form.
5. Save your form.
6. Enhance the form by changing widgets and properties.
7. Select Build > Preview to preview your form.
8. Write the code to interact with the form.

**Related concepts**

Creating the user interface on page 488

The Form Designer includes two working areas, one for the Form design and layout and one for the corresponding Record.

**Related tasks**

Create a form on page 488

You can create new forms in Form Designer.

---

### Creating the user interface

The Form Designer includes two working areas, one for the Form design and layout and one for the corresponding Record.

- **Forms** on page 488
- **Containers** on page 496
- **Widgets** on page 504
- **Web components** on page 515
- **Action management (Toolbars, Topmenus)** on page 517
- **Styles** on page 522

### Forms

Information about creating and compiling forms.

- **Create a form** on page 488
- **Create a database-aware form** on page 490
- **Create form with drag and drop** on page 491
- **Form item properties** on page 492
- **Tab index** on page 492
- **Compile a form** on page 493
- **Preview a form** on page 493

#### Create a form

You can create new forms in Form Designer.

There are two types of forms: form-only forms (using only non-database widgets) and database-aware forms. If you are building a database-aware form, confirm that you have added the Meta-schema file (4db) to your project.

1. Select **File > New > Genero Files** and choose a form from **Sources**.

<table>
<thead>
<tr>
<th>Form options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form from Database (4fdm)</td>
<td>Use wizard to select schema, columns and container from which to build new form.</td>
</tr>
<tr>
<td>Form (4fd)</td>
<td>Create a blank form (4fd).</td>
</tr>
<tr>
<td>Form as Text (per)</td>
<td>Create a blank text form file (per).</td>
</tr>
</tbody>
</table>

2. From the form design tab, Add a container. Form items must be in **Containers** on page 496.

3. **Add form items**. Within a container you can drag-and-drop various form items such as **Widgets** on page 504.

4. Each form and form item has properties that control its appearance and behavior. Set properties in the **Properties view** on page 534.

5. **Save** the form to the file system and/or project with **File > Save or File > Save As...**
6. **Preview** the form with **Build > Preview** to validate that it conforms to Genero rules and see how it will look to users.

7. **Compile** the form with **Build > Compile File**. You can also compile your form from Project Manager as part of a Build of an entire application.

**Related tasks**

- **Create a database-aware form** on page 490
  You can create a form from a database using a wizard to select the schema, columns and container from which to build a new form.

- **Create form with drag and drop** on page 491
  Drag and drop items from the DB Schemas tab to the Form or Records tab.

**Add base elements (Containers)**

Draw a container from the Container menu onto a form.

All form items must be nested within a container.

1. Open or create a new form.
2. Select a container from the **Container** menu and draw out the container on the form canvas. If you are nesting other objects into the container, draw the container first, then drag the other form elements into the new container. The background of the destination container turns a pale yellow when an object can be nested within it.
3. Set properties on the container as needed. Containers can be resized by selecting one of the corner handles on the container and dragging it to a new size.

**Related concepts**

- **Widgets** on page 504
  Widgets are designed for data handling, action triggering, or decoration.

- **Containers** on page 496
  Containers are used to group items on the form. Every form item must be contained within a container. A parent container can also have child containers. If there are multiple child containers, they must be grouped in a horizontal or vertical layout.

- **Properties and Form Designer** on page 545
  The **Properties** view lists all properties that can be set for a selected form item. The type of form item determines which properties are relevant; the list of properties changes as you move between different form items.

**Add data elements (Widgets)**

Each data elements (widget) added to a form and must be nested with in a container.

Every field on the form is of a specific widget type.

1. Open or create a new form.
2. **Add a container**.
3. **Add widgets**.

To add a widget from the menu options, do one of the following:

<table>
<thead>
<tr>
<th>Do you want to ...</th>
<th>Then ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a widget based on a database table or field?</td>
<td>Select <strong>Container &gt; Data Control</strong> to access the <strong>Data Control wizard</strong> on page 535. From the wizard, select the data field or fields and the container to place on the form. Each field will have a default widget type.</td>
</tr>
<tr>
<td>Create a widget that you will configure later?</td>
<td>Select <strong>Widget</strong>, then select the specific widget from the menu options. Draw out the widget within a container on the form.</td>
</tr>
</tbody>
</table>

4. Set properties on the widgets by selecting the widget on the form and setting its properties in the **Properties** view.
Adding a field to a form adds that field to the corresponding record on the Records tab. Adding a field to the form design outside of any existing containers, results in a new record in the Business Records diagram. You can manage the record and set its properties on the Records tab.

Related concepts

Widgets on page 504
Widgets are designed for data handling, action triggering, or decoration.

Containers on page 496
Containers are used to group items on the form. Every form item must be contained within a container. A parent container can also have child containers. If there are multiple child containers, they must be grouped in a horizontal or vertical layout.

Properties and Form Designer on page 545
The Properties view lists all properties that can be set for a selected form item. The type of form item determines which properties are relevant; the list of properties changes as you move between different form items.

Business records (data sets) on page 493
Business records model the data definition, structure and table relationships of the data used in a form, report, and/or web service. Business records are designed and modified in the Business Records diagram.

Create a database-aware form
You can create a form from a database using a wizard to select the schema, columns and container from which to build a new form.

Before you begin, confirm that you have added the meta-schema file to your project.

To be able to add database-aware controls to a form, the databaseName property of the form must be set.

1. Select File > New and select a category:

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genero BAM Desktop</td>
<td>Select this category if you are working on a project involving a BAM-generated desktop application.</td>
</tr>
<tr>
<td>Genero BAM Mobile</td>
<td>Select this category if you are working on a project involving a BAM-generated mobile application.</td>
</tr>
<tr>
<td>Genero</td>
<td>Select this category if you are working on a project that does not use the Business Application Modeler.</td>
</tr>
</tbody>
</table>

2. Select the form to create. The types of forms available depend on the category selected.

<table>
<thead>
<tr>
<th>If the category is ...</th>
<th>The forms from database options include ...</th>
</tr>
</thead>
</table>
| Genero BAM Desktop     | • CRUD form from Database (.4fdm)  
|                        | • Zoom form from Database (.4fdz)  
|                        | • Custom form from Database (.4fdc)  
|                        | In addition, you can create forms when adding a SOAP or JSON Web service. See Add Web services on page 250. |
| Genero BAM Mobile      | • CRUD form from Database (.4fdm)  
|                        | • Zoom form from Database (.4fdz)  
|                        | • Custom form from Database (.4fdc)  |
| Genero                 | • Form from Database (.4fd)  |

The New Form from Database wizard will guide you through setting up the form.

3. Select the columns from one of more tables to include in the form.
4. Select the container to use.
5. Set the relationships between tables if needed.
6. Change the label and widget used for each field if desired.

**Related concepts**
- **Data Control wizard** on page 535
  The Data Control wizard is used to rapidly create a form from a database meta-schema.

**Related tasks**
- **Create a form** on page 488
  You can create new forms in Form Designer.
- **Create form with drag and drop** on page 491
  Drag and drop items from the DB Schemas tab to the Form or Records tab.

**Create form with drag and drop**

Drag and drop items from the DB Schemas tab to the Form or Records tab.

Before you begin, confirm that you have added the meta-schema file to your project.

1. Create a new, empty form by selecting **File > New > Genero Files** and select **Form (.4fd)**.
2. Select the **DB Schemas tab** on page 374 to show available meta-schemas.
3. Select a table, field, or multiple fields (Ctrl-click) from the schema and drag them to the **Form tab** on page 531 or the **Business Record diagram** on page 494.
   - A label and formfield is added for each item dragged to the form. When a table is dragged and dropped, a container is automatically added.
   - **Note:** For managed forms for generated applications (**4fdm**), if the record is created and the selection contains multiple tables linked with foreign keys, the joins corresponding to the foreign keys are created.

**Related tasks**
- **Create a form** on page 488
  You can create new forms in Form Designer.
- **Create a database-aware form** on page 490
  You can create a form from a database using a wizard to select the schema, columns and container from which to build a new form.

**Convert a non-database form to a database form**

To associate widgets to a table column, the form must be linked to a database.

Before you begin, your project must include the database schema for the database.

Complete this procedure to convert a non-database form to a database form.

1. Open the form in Form Designer.
2. In the **Form Structure** view, select the **managedForm** node.
   - The **Properties** view lists all properties relating to the managed form node.
3. In the **Properties** view, click in the **databaseName** property field to display the comboBox.
   - The comboBox lists the names of the available databases.
4. Select the database from the list of databases.
5. Save your changes.

Once you have a database specified, you can update your widgets from a NON_DATABASE field type to a database-related field type.
Form item properties
Properties can be set on form items such as containers and widgets to provide information on how the runtime system should display or handle the item.

Properties for a form or form item can be modified by selecting the form item(s) on the form or a node in the Structure view. The properties that are valid for the selected item(s) are displayed.

Some properties are used frequently and are grouped together for easier reading and management. The composition of the groups vary, depending on the selected form item(s).

Default values
Each property has a default value. To reset a default value, use the Reset value button next to the property.

Initializers
Some properties include an initializer. The initializer defines how the property's default value is determined. For example, reference information from the database meta-schema (sqltype, notNull, required, uniqueKey, labels, etc.) is inherited to the form by the initializer. Initializers are computed on schema change.

If the property has an initializer option, an Initializer button appears to the left of the Reset value button and displays the Initializer dialog to set or edit the property's initializer value.

When an Initializer property is set or modified, the property name displays in bold and the Initializer button changes symbol.

Related concepts
Properties and Form Designer on page 545
The Properties view lists all properties that can be set for a selected form item. The type of form item determines which properties are relevant; the list of properties changes as you move between different form items.

Properties view on page 534

Tab index
The Tab Index indicates the order of the form items for data entry when using the Tab key to move from field to field.

Access the Tab Index by selecting Widget > Tab Index. All items that allow entries are displayed in the form design with an index number specifying the order.

Figure 242: Tab Index
The order can be altered in a variety of ways.
• Double-click a number to change it to 1.
• Click any number to change it to the next number after the last used.
• Right-click and select an option from the context menu.
• Finish the current sequence by pressing the Esc key.
• Press the Esc key again to exit tab order.

**Related concepts**

`tabIndex` on page 567

The `tabIndex` property defines the tab order for a form item.

**Compile a form**

Compiling a form translates it into an XML file with an extension of `.42f`.

The compiled form file (`42f`) is used by the Genero client to display the application form to the user. All `42f` files must be included with your application code in distributions.

With the form open, select **Build > Compile file** from the main menu.

• A form can also be compiled in Project Manager when the form is a part of a Build of an entire application.
• A form can also be compiled at the command-line with `gsform`.

**Related concepts**

The `gsform` command on page 525

The `gsform` tool is used at the command line to compile and import forms, convert `.4fd` files, extract localized strings, and keep a copy of the `.per` file created during compilation.

Building and linking programs on page 412

Build rules compile each file in a project. Link rules create the applications and libraries. Execution rules execute the application. Build rules, link rules, and execution rules together make up a Language in Genero Studio.

**Preview a form**

View the form using Genero Desktop Client. The form is automatically validated before the preview process. Only valid forms can be previewed.

Preview a form (`4fd`, `4fdm`, or `per`) by opening it and selecting **Build > Preview**. Alternatively, from the Project Manager, right-click on the form and select **Preview**.

**Importing .per files**

A Genero text form (`per`) can be imported and converted into Form Designer form files (`4fd`).

Import an existing Genero `per` file into your project with **File > Import text form (.per)**. This action imports and converts the file to a Form Designer form (`4fd`) file. If the `per` file is already in your project, right-click on the file and choose **Import Text Form**.

If your form includes web components, ensure **Create WebComponent settings (.wcsettings) on import of a text form** in the **Form Designer preferences** on page 528 is selected. This option generates the `.wcsettings` file that describes the web components, however it is recommended that you validate the generated `.wcsettings` file. For further information, see **WebComponents and Form Designer** on page 517.

**Tip:** To use a `per` file in its text form within Genero Studio, add the file using **Project > Add files**.

**Business records (data sets)**

Business records model the data definition, structure and table relationships of the data used in a form, report, and/or web service. Business records are designed and modified in the Business Records diagram.

**Forms**

Business records for forms model the data definition, structure, and table relationships required to generate the different CRUD operations for the form. When you design a form, a record is automatically created for each container on the form. The Records tab displays the screen record for the form in the Business Records
Reports

Business records for reports (4rd) contain the data definition, structure, and table relationships required to generate a rdd (Report Data definition file). The rdd file is used in conjunction with a Genero report definition (4rp) file to automatically generate the reports. A 4rd file opens in the Business Record diagram.

Services

Business records for web services (4ws) contain the data definition, structure, and table relationships required to generate the different CRUD operations for the web service. A 4ws file opens in the Business Record diagram.

Business records inherit their default information from the meta-schema from which they were created.

Tip: Where possible, use a single data set, gathering multiple database tables into this data set. The database can then use a single SELECT statement, which allows for better performance than running multiple queries. However, multiple data sets are necessary in some cases; for example, you need multiple independent lists of data in your reports, or your query is too complex for the database engine.

Business Record diagram

The Business Record diagram is used to define the data set of the form, report, or web service.

Figure 243: Business Record diagram
Related concepts

Business records (data sets) on page 493

Business records model the data definition, structure and table relationships of the data used in a form, report, and/or web service. Business records are designed and modified in the Business Records diagram.

Business record properties

Business records have properties that are set in the Properties view.

Table 168: Business Record properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Record name.</td>
</tr>
<tr>
<td>active</td>
<td>The active property indicates that the table participates in the application code generation. It is checked by default when a managed form (4fdm) is created. When a table is inactive, it does not provide the modeling features and cannot be or be linked to a master record.</td>
</tr>
<tr>
<td>masterTable</td>
<td>A data set can be composed of several database tables. The master table is a table on which CRUD operations will apply. The generator creates the CRUD operations for a given data set only if its unique key is composed by fields of the master table.</td>
</tr>
<tr>
<td>unique key</td>
<td>A data set unique key is a field or a list of fields ensuring the uniqueness of data in the data set. It must be defined as a primary key or a secondary key in the database schema.</td>
</tr>
</tbody>
</table>

Functionality

See Enable and disable CRUD logic on page 291. Only for generated forms and web services.

Query

See Joins and Data order on page 539.

Table 169: RecordField properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Record field name.</td>
</tr>
<tr>
<td>lookup</td>
<td>See Lookup fields on page 299.</td>
</tr>
<tr>
<td>fieldType</td>
<td>Used to specify that its data type is derived from the data types in a database table or that it is Non_Database, indicating that the data type is not derived from a database columnSee formFields on page 504.</td>
</tr>
<tr>
<td>sqlTabName</td>
<td>The sqlTabName property is the name of the database table for formField form items.</td>
</tr>
<tr>
<td>colName</td>
<td>The colName property is the name of the database column.</td>
</tr>
<tr>
<td>tableAliasName</td>
<td>Table alias name.</td>
</tr>
<tr>
<td>fieldIdRef</td>
<td>Unique reference id of the field in the record.</td>
</tr>
<tr>
<td>dataType</td>
<td>See dataType on page 550.</td>
</tr>
<tr>
<td>defaultValue</td>
<td>See defaultValue on page 551.</td>
</tr>
</tbody>
</table>
### Table 170: Record Relation properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>foreignFields</td>
<td>The foreign key field(s) in the relation.</td>
</tr>
<tr>
<td>primaryFields</td>
<td>The primary key field(s) in the relation.</td>
</tr>
</tbody>
</table>

#### Add or edit a Record, RecordField, or Relation

Records, RecordFields, and Relations are managed in the Business Records diagram.

1. To add a new record, right-click on the background of the Business Records diagram and select **Add Record**.
2. Right-click on a record and select **Add Field** to add a new field to a record. The record and fields are listed in the Structure view.
   
   In Form Designer, new recordFields are also added to the form design in the Form tab.
3. Set the relationship between two records by right-clicking on the foreign field in one record and dragging it to the primary key in the other record. Select the Relation and modify the `foreignFields` and `primaryFields` properties if needed.

#### Containers

Containers are used to group items on the form. Every form item must be contained within a container. A parent container can also have child containers. If there are multiple child containers, they must be grouped in a horizontal or vertical layout.

- **HBox and VBox - layouting** on page 496
- **Grid - positioning** on page 497
- **ScrollGrid - positioning** on page 498
- **Group - grouping** on page 499
- **Folder page - stacking** on page 500
- **Table - organizing** on page 501
- **Tree - hierarchy** on page 502
- **HRec - aligning fields** on page 503
- **Matrix** on page 503
- **Data Control** on page 503

**Note:** Some containers are not supported on mobile platforms.

#### HBox and VBox - layouting

HBox and VBox containers are used to layout the containers nested within them either horizontally from left to right (HBox) or vertically from top to bottom (VBox).

Containers are packed in the HBox or VBox container in the order in which they appear in the structure. By combining HBox and VBox layouts for various containers, you can align the containers of your form in any way you choose.
Example

Figure 244: HBox Layout Container

Example

Figure 245: VBox Layout Container

Properties

name on page 562, posX, posY on page 563, gridHeight, gridWidth on page 556

Grid - positioning
The Grid container declares a formatted text block defining the dimensions and the positions of the form elements for a unique-record presentation.

With Grid, you can specify the position of labels, formFields for data entry or additional interactive objects such as buttons. Grids have no visual representation when the form is displayed.
**Figure 246: Grid Container**

**Properties (by group)**

Object: *name* on page 562, *tag* on page 567, *style* on page 567

Geometry: *posX*, *posY* on page 563, *gridHeight*, *gridWidth* on page 556

Display: *hidden* on page 556, *fontPitch* on page 552

Comment: *comment* on page 549, *localizedStr*

**ScrollGrid - positioning**

The ScrollGrid container declares a formatted text block defining the dimensions and the positions of the form elements for a multi-record presentation.

**Example**

ScrollGrid is similar to the Grid container, except that you can repeat the screen elements on several "row-templates", in order to design a multiple-record view that appears with a vertical scrollbar. A ScrollGrid may be a container of, or contained within, both Grids and ScrollGrids.
Figure 247: ScrollGrid Container

Properties (by group)

Object: name on page 562, tag on page 567, style on page 567

Geometry: posX, posY on page 563, gridHeight, gridWidth on page 556

Constraints: wantFixedPageSize on page 571, initialPageSize on page 558

Display: hidden on page 556, fontPitch on page 552, gridChildrenInParent on page 555

Comment: comment on page 549, localizedStr

Miscellaneous: doubleClick on page 552

Group - grouping

A Group container can be used to display a titled box (usually called a groupbox) around contained elements.

To display a groupbox around a set of fields, nest your Grid or other container within a Group container.

Example

Figure 248: Group Container
Folder page - stacking

A Folder container is used to display children (Pages) inside a Folder tab.

The Folder container manages the Page containers displayed as tabs. You can add or remove and order the Pages using the right-click contextual menu on the Folder or Page object on the form.

Example

![Folder and Pages](image)

Figure 249: Folder and Pages

Folder Properties (by group)

Object: name on page 562, tag on page 567, style on page 567

Geometry: posX, posY on page 563, gridHeight, gridWidth on page 556

Display: hidden on page 556, fontPitch on page 552

Comment: comment on page 549, localizedStr

Page Properties (by group)

Object: name on page 562, tag on page 567, style on page 567

Display: hidden on page 556, image on page 557

Text: text on page 568, localizedStr

Comment: comment on page 549, localizedStr

Miscellaneous: action on page 546

Usage

The text property defines the label of the folder page. The image property can be used to specify which image to use as an icon.

If needed, you can use the action property to bind an action to a folder Page. When the Page is selected, the program gets the corresponding action event.
To bring a folder page to the top, in your source code, use NEXT FIELD to one of the active fields of the page (The NEXT FIELD field-name instruction gives the focus to the specified field.) or use the ui.Form.ensureFieldVisible() method if the fields are disabled/unused or the ui.Form.ensureElementVisible() method if the page does not contain focusable elements.

With the tabIndex property of a field in a Page, you can define which field gets the focus when a Page is selected.

**Table - organizing**

The Table container defines the presentation of a list of records, bound to a screen record (also called a screen array).

Each record is displayed on a separate line. The screen record appears on the Records tab.

**Example**

![Figure 250: Table Container](image)

**Properties (by group)**

Object: name on page 562, tag on page 567, style on page 567

Geometry: posX, posY on page 563, gridHeight, gridWidth on page 556

Size: width, widthUnit, height, heightUnit

Constraints: wantFixedPageSize on page 571

Display: hidden on page 556, fontPitch on page 552

Aggregate: text on page 568, localizedStr

Comment: comment on page 549, localizedStr

Appearance: unsortableColumns on page 569, unsizableColumns on page 569, unmovableColumns on page 569, unhidableColumns on page 568

Miscellaneous: doubleClick on page 552, totalRows on page 568, rowHeight on page 564

**Usage**

Drag and drop the columns to change the initial order.

To add or delete columns or edit a column title, select the column title and right-click to display a menu of options.

The screen record definition on the Records tab must have exactly the same columns as the Table container. However, the order of the screen record fields can be different from the column order, to match the program array elements, for example when the database table defines the columns (DEFINE LIKE) in a different order as the form table.

By default, the current row in a Table is highlighted in display mode, but it is not highlighted in input mode. You can set decoration attributes of a table with a style.

If the aggregate property is set for one of the columns in the table, a summary line will be displayed.
Tree - hierarchy
The Tree container defines the presentation of a list of ordered records in a tree-view widget.

Example

![Tree Container](image)

Figure 251: Tree Container

Properties
Object: name on page 562, tag on page 567, style on page 567
Tree: parentIdColumn on page 562, idColumn on page 557, expandedColumn on page 552, isNodeColumn on page 559, imageCollapsed on page 557, imageLeaf on page 558
Geometry: posX, posY on page 563, gridHeight, gridWidth on page 556
Size: width, widthUnit, height, heightUnit
Constraints: wantFixedPageSize on page 571
Display: hidden on page 556, fontPitch on page 552
Comment: comment on page 549, localizedStr
Appearance: unsortableColumns on page 569, unsizableColumns on page 569, unmovableColumns on page 569, unhidableColumns on page 568
Miscellaneous: doubleClick on page 552, totalRows on page 568, rowHeight on page 564

Usage
Tree views are very similar to regular table containers; before reading further about tree views, you should be familiar with Table containers. The Tree container allows you to specify the layout of a graphical tree widget, displaying data in a parent-child relationship. Plus and minus icons allow the user to expand and contract a branch of the tree.

You specify the layout for the tree in Form Designer; the tree content is provided in your BDL program.

A Tree is made of rows and columns, very similar to a Table container, with the exception that the rows in a tree can be nested.

The Tree container is created by default with two Edit columns, and two Phantom columns that do not display on the screen. The Phantom columns of the Tree specify the id and the parent id of each row, enabling nesting. The other columns display data rows in a normal format without nesting, as in a table.

Properties of the tree container specify the idColumn and parentIdColumn. Additional properties specify images to be used for the expand/contract icons and leaf icons.

The rows in the tree are automatically defined in your form as a screen array. In your BDL program, define a corresponding dynamic array that matches the screen array, using the screen record in the Form to determine column names and their order. Your program must populate the rows of the container at runtime.

See the topic on Tree Views in the Genero Business Development Language User Guide for more details about tree-view programming in Genero.
**HRec - aligning fields**
An HRec item is a special container that uses spacers to align widgets in a form.

Spacers are one or more blanks defining an invisible element that expands automatically. Without spacers, blank areas are ignored and the resulting form may not display as you expect. Draw out the HRec container and then drag and drop form items into it. Once you have placed form items into the HRec container you can right-click on the form item to add additional spacer objects to the right or left of the form item. The spacer objects force the spacing between the form items within the HRec container.

**HRec Container**

![HRec Container Diagram]

**Properties**
- **name** on page 562, **posX, posY** on page 563, **gridHeight, gridWidth** on page 556

**Matrix**
A Matrix container manages a screen array, usually a repetitive array of fields in the screen layout, each containing identical groups of screen fields.

Each row of a screen array is a screen record.

Right-click on a field to **Convert to Matrix** and set its Matrix properties. The Structure View marks matrix fields with a preceding + sign.

**Table 171: Matrix Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repeat</td>
<td>Determines if field repeats. Default is checked (TRUE).</td>
</tr>
<tr>
<td>columnCount</td>
<td>Number of columns of repeated fields.</td>
</tr>
<tr>
<td>rowCount</td>
<td>Number of rows of repeated fields. Minimum value is 1. Both columnCount and rowCount cannot be 1, otherwise the field would be a standard formField and not a repeated field.</td>
</tr>
<tr>
<td>stepX</td>
<td>Number of cells (horizontally) between the repeated fields. (Relevant only if columnCount &gt; 1.)</td>
</tr>
<tr>
<td>stepY</td>
<td>Number of cells (vertically) between the repeated fields. (Relevant only if rowCount &gt; 1.)</td>
</tr>
</tbody>
</table>

**Data Control**
Although this item appears on the Container menu, it is actually a wizard that allows you to create a new form or add multiple fields to the form, based on database table columns.

**Related concepts**
- Data Control wizard on page 535
The Data Control wizard is used to rapidly create a form from a database meta-schema.

**Widgets**

Widgets are designed for data handling, action triggering, or decoration.

- **Button** on page 505
- **ButtonEdit** on page 505
- **Canvas** on page 506
- **CheckBox** on page 506
- **ComboBox** on page 507
- **DateEdit** on page 508
- **Edit** on page 509
- **Field - deprecated** on page 509
- **HLine** on page 509
- **Image** on page 510
- **Label** on page 510
- **Phantom** on page 511
- **ProgressBar** on page 511
- **RadioGroup** on page 512
- **Slider** on page 513
- **SpinEdit** on page 513
- **TimeEdit** on page 515
- **TextEdit** on page 514

**formFields**

A formField is a type of form object that can be used to display data or take input.

A formField is presented to the user through a widget; the most commonly used widget, Edit, defines a simple line edit box that allows the user to enter a value directly into the formField. Other widgets, such as the ComboBox and Checkbox, present the data contained in the field in a user-friendly way. For example, a ComboBox defines a dropdown box of values, allowing the user to select from a list of valid values for the underlying formField.

In the Properties View, the fieldType property of a formField may be set to specify that its data type is derived from the data types in a database table or that it is Non_Database, indicating that the data type is not derived from a database column.

If the data type is to be derived from a database table, the databaseName property of the form must be set to a database for which there is a database meta-schema file listed in the Db Schemas tab. The sqlTabName and colName properties of the formField must be selected from the tables and columns in the schema file. The dataType property will be automatically filled based on the data type for the table column in the schema file.

A form has built-in validation to insure that the value entered into a formField is compatible with the declared dataType in the form definition file. Additional validation routines can be specified in your BDL program.

The values entered into formFields are stored in variables in the Genero program, which may be used in the program in any way. A common use is to provide values for SQL statements that update a database.

**Related concepts**

- **fieldType** on page 552
- The **fieldType** property specifies the data type of the item.

- **colName and sqlTabName** on page 548
The **colName** property is the name of the database column, the **sqlTabName** property is the name of the database table for formField form items.

**Button**
The Button widget defines a push-button with a label or picture that can trigger an action defined in your BDL program code.

When a button is clicked, the program code defined for the action is triggered.

**Example**

![Click Me](image)

**Figure 252: Button**

**Properties**
- **action** on page 546, **comment** on page 549, **fontPitch** on page 552, **hidden** on page 556, **image** on page 557, **sample** on page 564, **sizePolicy** on page 565, **style** on page 567, **tabIndex** on page 567, **text** on page 568, **tag** on page 567

**ButtonEdit**
A ButtonEdit widget defines a line edit box with a button on the right side.

**Purpose**
The ButtonEdit editable box is a formField, which can be associated with a database column by changing the **fieldType** property to TABLE_COLUMN, and specifying the sqlTabName and colName properties. The button can trigger an action defined in your BDL program code. Frequently this action is designed to open a window that displays a list of values for the user to choose from.

**Example**

![f2](image)

**Figure 253: ButtonEdit**

**Properties**
- **action** on page 546, **autoNext** on page 547, **case** on page 548, **century** on page 548, **color** on page 548, **colorCondition** on page 548, **comment** on page 549, **defaultValue** on page 551, **Display Like** on page 551, **case** on page 548, **fontPitch** on page 552, **hidden** on page 556, **format** on page 553, **image** on page 557, **include** on page 558, **invisible** on page 559, **justify** on page 559, **notNull** on page 562, **noEntry** on page 562, **picture** on page 562, **reverse** on page 564, **sample** on page 564, **scroll** on page 565, **sizePolicy** on page 565, **style** on page 567, **required** on page 564, **tag** on page 567, **tabIndex** on page 567, **Validate Like** on page 569, **verify** on page 570
Canvas
A Canvas widget defines an area reserved for drawing.

Example

![Canvas Widget Example](image)

Figure 254: Canvas
See the topic on Canvas in the *Genero Business Development Language User Guide* for more information.

Properties
- comment on page 549, hidden on page 556, tag on page 567

CheckBox
The CheckBox widget defines a boolean entry with a box and a text label.

Example

![CheckBox Example](image)

Figure 255: CheckBox

Properties
- color on page 548, colorCondition on page 548, comment on page 549, defaultValue on page 551, fontPitch on page 552, color on page 548, include on page 558, justify on page 559, notNull on page 562, noEntry on page 562, required on page 564, sample on page 564, sizePolicy on page 565, style on page 567, tag on page 567, tabIndex on page 567, text on page 568, Validate Like on page 569, valueChecked on page 570, valueUnchecked on page 570

Usage
The text on page 568 attribute defines the label to be displayed near the check box.

The box shows a checkmark when the form field contains the value defined in the valueChecked attribute (for example: "Y"), and shows no checkmark if the field value is equal to the value defined by the valueUnchecked attribute (for example: "N"). If you do not specify the valueChecked or valueUnchecked attributes, they respectively default to TRUE (integer 1) and FALSE(integer 0).

By default, during an INPUT, a CheckBox field can have three states:
- Grayed (NULL value)
- Checked (valueChecked value)
- Unchecked (valueUnchecked value)

If the field is declared as notNull, the initial state can be grayed if the default value is NULL; once the user has changed the state of the CheckBox field, it switches only between checked and unchecked states.

During an CONSTRUCT, a CheckBox field always has three possible states (even if the field is notNull), to let the user clear the search condition:
- Grayed (No search condition)
- Checked (Condition column = valueChecked value)
- Unchecked (Condition column = valueUnchecked value)

**ComboBox**
A ComboBox is a data handling widget that defines a drop-down list of values, allowing the user to select a value.

**Figure 256: Combobox**

**Defining the underlying formField**
When you drop a ComboBox form item onto your form, it sets the fieldType property to NON_DATABASE as the default. To associate the ComboBox underlying formField with a database column, change the `fieldType` property to TABLE_COLUMN, and specifying the `tableName` and `colName` properties.

**Usage**
To hard-code the values for the ComboBox, enter each value as a member of the `items` property. When you add an item to the list, you specify a Name and Text for each item. The Name value is the value to be read from/stored into the underlying formField, while the Text value is what is displayed to the end user.

**Figure 257: Edit Items in a Combobox**
In this example, the entries represent a key-value pair for three cities. The end-user sees proper names such as "Paris", "Madrid", or "London". The underlying formField will hold the key equivalent, in our case either a "1", "2", or "3".

As an alternative, your application may include a function whose role is to populate the combobox. The `initializer` property specifies the name of that function.

If neither `items` nor `initializer` properties are specified, the form compiler automatically fills the list of items with the values of the `include` property, when specified.

The `queryEditable` property can be used to force the ComboBox to be editable during a CONSTRUCT instruction. Some front-ends support different presentation options which can be controlled by a `style` property. You can for example enable the first item to be selected when pressing keys.

For more information about programming for a ComboBox, see the *Genero Business Development Language User Guide*. 
Context Menu

The values in the list of items can be managed (add, delete) by right-clicking on the ComboBox on the form in Form Designer and selecting **Edit Items**. Edit the values in the columns of the dialog:

- **Name** - the value to be stored in the underlying formField represented by the ComboBox, for example, "W".
- **Text** - the value in the list to be displayed to the user, for example "West Region".
- **Localize string** - Specify whether the text is a localized string (true/false).

**DateEdit**

A DateEdit is a data handling widget that defines a line edit box with a button that opens a calendar window, allowing the user to select a date value.

**Purpose**

The dateEdit underlying formField can be associated with a database column by changing the **fieldType** property to **TABLE_COLUMN**, and specifying the sqltabName and columnName properties.

**Example**

![Figure 258: DateEdit](image)

**Properties**

`autoNext` on page 547, `century` on page 548, `color` on page 548, `colorCondition` on page 548, `comment` on page 549, `defaultValue` on page 551, `fontPitch` on page 552, `format` on page 553, `hidden` on page 556, `include` on page 558, `justfy` on page 559, `notNull` on page 562, `noEntry` on page 562, `required` on page 564, `sample` on page 564, `SizePolicy` on page 565, `style` on page 567, `tag` on page 567, `tabIndex` on page 567, `Validate Like` on page 569

**Usage**

Some front-ends support different presentation options which can be controlled by a `style` on page 567 attribute. For example, you can change the first day of the week or the icon of the button.

**DateTimeEdit**

The DateTimeEdit widget defines a date-time editor widget.

**Purpose**

The DateTimeEdit form item type allows the user to edit date-time values with a specific widget for date-time input. A DateTimeEdit field typically provides a calendar and clock widget, to let the end user pick a date and time from it.

The display and input precision (time part with or without seconds) of the DateTimeEdit widget depends from the front-end. On some platforms, native date-time editors do not handle the seconds. Further, some front-ends (especially on mobile devices) deny data types different from `DATETIME YEAR TO SECOND` or `DATETIME YEAR TO MINUTE`.

To store DateTimeEdit field values, consider using the appropriate data type according to the target front-end (`DATETIME YEAR TO SECOND` or `DATETIME YEAR TO MINUTE`).
**Important**: If the front-end does not support the data type used for the DateTimeEdit field, the runtime system will raise an error and stop the program. Consider testing your application with all type of front-ends.

On some front-end platforms, the native widget used for DateTimeEdit fields allows only pure date-time value input, and therefore cannot be used with a `CONSTRUCT` instruction, where it must be possible to enter search filters like ">= 2014-01-23 11:00”.

### Properties

- `autoNext` on page 547
- `century` on page 548
- `color` on page 548
- `colorCondition` on page 548
- `comment` on page 549
- `defaultValue` on page 551
- `fontPitch` on page 552
- `format` on page 553
- `hidden` on page 556
- `include` on page 558
- `justify` on page 559
- `notNull` on page 562
- `noEntry` on page 562
- `required` on page 564
- `sample` on page 564
- `SizePolicy` on page 565
- `style` on page 567
- `tag` on page 567
- `tabIndex` on page 567
- `Validate Like` on page 569

### Edit

The Edit item type defines a simple line-edit field for data input or display.

The Edit item type is a data handling widget for an underlying `formfield`. The Edit formField can be associated with a database column by changing the `fieldType` property to `TABLE_COLUMN`, and specifying the `sqltabName` and `columnName` properties.

#### Example

![Entry 1234](image)

**Figure 259: Edit item type**

### Attributes

- `autoNext` on page 547
- `century` on page 548
- `color` on page 548
- `colorCondition` on page 548
- `comment` on page 549
- `defaultValue` on page 551
- `Display Like` on page 551
- `fontPitch` on page 552
- `format` on page 553
- `hidden` on page 556
- `include` on page 558
- `justify` on page 559
- `notNull` on page 562
- `noEntry` on page 562
- `required` on page 564
- `sample` on page 564
- `SizePolicy` on page 565
- `style` on page 567
- `tag` on page 567
- `tabIndex` on page 567
- `Validate Like` on page 569
- `verify` on page 570

### Field - deprecated

The Field item type is deprecated but is retained for backwards compatibility. It defines a generic form field for data input or display that is defined in database schema files. It is recommended that you use other widgets with more specific functionality.

![Field](image)

**Figure 260: Field**

For more information, see *Genero Business Development Language User Guide*.

### HLine

The HLine item type appears in the form as a horizontal line.

#### Example

```
```

**Figure 261: HLine**
Image
Image items define areas where a picture file can be displayed.
Images can be either static or dynamic formFields.

Example

![Image Example]

Figure 262: Image

Properties
autoScale on page 547, comment on page 549, hidden on page 556, style on page 567, stretch on page 567, tag on page 567

Static Image only: image on page 557

Image Field only: color on page 548, colorCondition on page 548, fontPitch on page 552, justify on page 559, sizePolicy on page 565, sample on page 564

Usage
A Static Image displays an image that has its source file defined by the image property. This type of image item must be used to display text that does not change often, such as background pictures or logos. The item is not a formField. The image file can be changed from the BDL program by using the API provided to manipulate the user interface (see the Genero Business Development Language User Guide topic Dynamic User Interface for more details). It is not possible to change the image with a DISPLAY TO instruction.

A formField Image is a widget that gets the image file based on the underlying formField. The value of the formField is the image file specified with a URL. The formField can be associated with a database column by changing the fieldType on page 552 property to TABLE_COLUMN, and specifying the sqlTabName and colName properties. This type of image item must be used to display values that change often during program execution, like database information. The picture can be changed from the BDL program by using the DISPLAY TO instruction to set the value of the field.

Label
The Label item type defines a simple text area to display a read-only value.

Labels can be either static or dynamic formFields.

Example

![Label Example]

Figure 263: Label

Properties
color on page 548, colorCondition on page 548, comment on page 549, fontPitch on page 552, hidden on page 556, imageColumn on page 557, justify on page 559, reverse on page 564, sample on page 564, sizePolicy on page 565, style on page 567, tag on page 567
**FormField Label only:** format on page 553, sample on page 564

**Static Label only:** text on page 568

**Table Column only:** unHidable on page 568, unMovable on page 568, unSizable on page 569, unSortable on page 569, title on page 568

**Usage**

A **FormField Label** item is used to display values that change often during program execution, like database information. The text of the label is defined by the value of the corresponding form field. The text can be changed from the BDL program by using the DISPLAY TO instruction to set the value of the field, or within a list by using a DISPLAY ARRAY. This kind of Form Item does not allow data entry; it is only used to display values. The text automatically changes when the values in the table column change.

Some front-ends support different presentation options which can be controlled by a style attribute. You can for example change the text format to render HTML content.

A **Static Label** item is used to display text that does not change often, like field descriptions. The text of the label is defined by the text attribute; the item is not a form field. Double-click the widget, or right-click and select **Edit Text**, to edit the text property. The text can be also changed from the BDL program by using the API provided to manipulate the user interface (see Dynamic User Interface for more details). It is not possible to change the text with a DISPLAY TO instruction. This kind of item is not affected by instructions such as CLEAR FORM. Static labels display only character text values, and therefore do not follow any justification rule as form field labels.

**Phantom**

A Phantom field can be used to specify a formField that is listed in a screen-record, but does not have to be displayed in the form.

**Usage**

Phantom fields are never displayed to the user, although they can be used by dialog instructions of BDL programs. If you want to implement a screen-array with all the columns of a database table defined in the schema file, but you don't want to display all the columns in the table, you must use Phantom fields. With the screen-array matching the database table, you can easily write program code to fetch all columns into an array defined with a LIKE clause.

Phantom fields can be based on database columns defined in a schema file or as NON_DATABASE field.

Phantom fields are used to store the id and parent id of the nodes in a tree object.

To add a Phantom field to your form, right-click on the form object and select **Add Phantom** from the context menu.

Phantom field data is never send to the front-ends. Therefore, you can use a phantom field to store critical data that must not go out of the application server.

**ProgressBar**

A ProgressBar is a data handling widget that can indicate the current progress of an operation.

**Example**

![ProgressBar](image)

**Figure 264: ProgressBar**

**Properties**

color on page 548, colorCondition on page 548, comment on page 549, fontPitch on page 552, hidden on page 556, justify on page 559, sample on page 564, sizePolicy on page 565, style on page 567, tag on page 567, valueMin on page 570, valueMax on page 570
**Usage**

A ProgressBar item does not allow data entry; it is only used to display integer values.

The position of the progress bar is defined by the value of the corresponding form field. The value can be changed from the BDL program by using the DISPLAY TO instruction to set the value of the field.

The and `valueMin` and `valueMax` properties define respectively the lower and upper integer limit of the progress information. Any value outside this range will not be displayed. Default values are `VALUEMIN=0` and `VALUEMAX=100`.

Some front-ends support different presentation options which can be controlled by a `style` property. For example, you could display a percentage.

This widget has to be used with a SMALLINT or INTEGER variable, larger types like BIGINT or DECIMAL are not supported.

**RadioGroup**

A RadioGroup data handling widget presents a set of radio buttons from which the user may select a value.

**Purpose**

The underlying formField for the RadioGroup can be associated with a database column by changing the `fieldType` property to `TABLE_COLUMN`, and specifying the `sqltabName` and `colName` properties. Adding/deleting values for the radio buttons is managed from the contextual menu.

**Example**

**Figure 265: RadioGroup**

![RadioGroup Image]

**Property**

- `color` on page 548
- `colorCondition` on page 548
- `comment` on page 549
- `defaultValue` on page 551
- `fontPitch` on page 552
- `hidden` on page 556
- `include` on page 558
- `items` on page 559
- `justify` on page 559
- `notNull` on page 562
- `noEntry` on page 562
- `orientation` on page 562
- `required` on page 564
- `sample` on page 564
- `sizePolicy` on page 565
- `style` on page 567
- `tag` on page 567
- `tabIndex` on page 567
- `Validate Like` on page 569

**Usage**

This item type defines a set of radio buttons where each button is associated with a value defined in the `items` on page 559 property.

The text associated with each value will be used as the label of the corresponding radio button.

If the `items` property is not specified, the form compiler automatically fills the list of items with the values of the `include` property, when specified. However, the item list will not automatically be populated with include range values (i.e. values defined using the TO keyword). The item property can be specified directly in the form or indirectly in the schema files.

During an INPUT, a RadioGroup field value can only be one of the values specified in the `items` property. During an CONSTRUCT, a RadioGroup field allows to uncheck all items (even if the field is `null`), to let the user clear the search condition.
If one of the items is explicitly defined with NULL and the notNull property is omitted; In INPUT, selecting the corresponding radio button sets the field value to null. In CONSTRUCT, selecting the radio button corresponding to null will be equivalent to the = query operator, which will generate a "colname is null" SQL condition.

Use the orientation property to define if the radio group must be displayed vertically or horizontally.

Some front-ends support different presentation options which can be controlled by a style property. For example, you can define what item has to be selected first when pressing keys.

### Slider

The Slider item type defines a horizontal or vertical slider.

#### Example

![Slider Example](image)

**Figure 266: Slider**

#### Properties

- color on page 548
- colorCondition on page 548
- comment on page 549
- defaultValue on page 551
- fontPitch on page 552
- hidden on page 556
- include on page 558
- justify on page 559
- orientation on page 562
- sample on page 564
- sizePolicy on page 565
- step on page 566
- style on page 567
- tabIndex on page 567
- tag on page 567
- Validate Like on page 569
- valueMin on page 570
- valueMax on page 570

#### Usage

The slider item type is a data handling widget for the underlying formField. The formField can be associated with a database column by changing the fieldType property to TABLE_COLUMN, and specifying the sqltabName and colName properties.

This item type defines a classic widget for controlling a bounded value. It lets the user move a slider along a horizontal or vertical groove and translates the slider's position into a value within the legal range.

The valueMin and valueMax properties define respectively the lower and upper integer limit of the slider information. Any value outside this range will not be displayed; the step between two marks is defined by the step property. The orientation property defines whether the Slider is displayed vertically or horizontally. If valuemin and/or valuemax are not specified, they default respectively to 0 (zero) and 5.

This widget has to be used with a SMALLINT or INTEGER variable, larger types like BIGINT or DECIMAL are not supported.

**Note:** This widget is not designed for CONSTRUCT, as you can only select one value.

### SpinEdit

The SpinEdit item type defines a spin box widget.

#### Example

![SpinEdit Example](image)

**Figure 267: SpinEdit**

#### Properties

- autoNext on page 547
- color on page 548
- colorCondition on page 548
- comment on page 549
- defaultValue on page 551
- fontPitch on page 552
- hidden on page 556
- include on page 558
- justify on page 559
- notNull on page 562
- noEntry on page 562
- required on page 564
- sample on page 564
Usage

This item type allows the user to choose a value either by clicking the up/down buttons to increase/decrease the value currently displayed, or by typing the value directly into the spin box.

The step between two values is defined by the step attribute.

The valueMin and valueMax attributes define respectively the lower and upper integer limit of the spin-edit range. There is no default min or max value for the SPINEDIT widget.

This widget has to be used with a SMALLINT or INTEGER variable, larger types like BIGINT or DECIMAL are not supported.

This widget is not designed for CONSTRUCT, as you can only enter an integer value.

TextEdit

The TextEdit item type defines a multi-line edit field for data input or display.

Example

Figure 268: TextEdit

Properties

case on page 548, color on page 548, colorCondition on page 548, comment on page 549, defaultValue on page 551, fontPitch on page 552, hidden on page 556, include on page 558, justify on page 559, noEntry on page 562, notNull on page 562, required on page 564, sample on page 564, scrollbars on page 565, sizePolicy on page 565, stretch on page 567, style on page 567, tabIndex on page 567, tag on page 567, ValidateLike on page 569, wantTabs on page 571, wantNoReturns on page 571

Usage

This kind of form field allows the user to enter a long text on multiple lines.

By default, when the focus is in a TextEdit field, the TAB key moves to the next field, while the RETURN key adds a newline (ASCII 10) character in the text. To control the user input when the TAB and RETURN keys are pressed, you can specify the wantTabs and wantNoReturns properties. When you specify wantTabs, the TAB key is consumed by the TextEdit field, and a TAB character is added to the text. The user can still jump out of the field with the Shift-TAB combination. When you specify wantNoReturns, the RETURN key is not consumed by the TextEdit field, and the action corresponding to the RETURN key is triggered. The user can still enter a newline character with Shift-RETURN or Control-RETURN.

You can use the scrollbars property to define vertical and/or horizontal scrollbars. By default, this attribute is set to Vertical. The stretch property can be used to force the TextEdit field to stretch when the parent container is re-sized. Values can be NONE, X, Y or BOTH. By default, this attribute is set to NONE. Note that using either the SCROLLBARS or the STRETCH attribute will automatically set the SCROLL attribute. For more details about size limitation, see the scroll property.
Some front-ends support different text formats which can be controlled by a *style* property. You can for example display and input HTML content in a TextEdit.

Since Genero 2.20, a TextEdit can also be used to edit rich text format. Depending on the front-end, different formatting options are available (bold, font size, and so on) and can be controlled using either an integrated toolbox or via local actions. In this case, the value of the field will be an HTML representation of the text and its decoration.

**Note:** Each front-end uses its underlying technology to provide this feature and the HTML representation may vary between front-ends. They are most of the time compatible but not every time, and the HTML representation may change depending on the version.

**Note:** When using rich text, `FGL_DIALOG_SET_CURSOR()` and `FGL_DIALOG_SET_SELECTION()` functions must be called carefully. The rich text format, having a corresponding cursor position/selection between displayed text and HTML representation, may make it difficult.

### TimeEdit

The TimeEdit item type defines a time editor widget.

#### Example

![Figure 269: TimeEdit item type](image)

#### Properties

- `autoNext` on page 547
- `color` on page 548
- `colorCondition` on page 548
- `comment` on page 549
- `defaultValue` on page 551
- `fontPitch` on page 552
- `hidden` on page 556
- `include` on page 558
- `justify` on page 559
- `noEntry` on page 562
- `notEmpty` on page 562
- `required` on page 564
- `sample` on page 564
- `sizePolicy` on page 565
- `style` on page 567
- `tabIndex` on page 567
- `tag` on page 567
- `Validate Like` on page 569

#### Usage

This item type allows the user to edit times by using the keyboard or the arrow keys to increase/decrease time values.

With this widget, the user can only enter a `DATETIME HOUR TO SECOND` value.

This widget is not designed for `CONSTRUCT`, as you can only enter time.

### Web components

Web Components are usually complex widgets displaying detailed information on the screen, such as charts, graphs, or calendars.

- Web component widget on page 516
- Add a WebComponent to a form on page 516
- WebComponents and Form Designer on page 517

#### Related concepts

GSTWCDIR on page 168
The GSTWCDIR environment variable defines the directory in which the WebComponent XML files (.wcsettings) and optional image files are stored. By default this environment variable is set in the Web Components environment set.

**Web component widget**
The WebComponent item type defines a generic form field that can receive an external widget.

**Properties**
- `color` on page 548, `colorCondition` on page 548, `componentType` on page 550, `comment` on page 549, `defaultValue` on page 551, `fontPitch` on page 552, `hidden` on page 556, `include` on page 558, `justify` on page 559, `noEntry` on page 562, `notNull` on page 562, `properties` on page 564, `scrollbars` on page 565, `sizePolicy` on page 565, `style` on page 567, `stretch` on page 567, `tabIndex` on page 567, `tag` on page 567, `Validate Like` on page 569

**Usage**
The WebComponent item type defines a form field which can be implemented with a plug-in mechanism on the front-end side.

You must define the type of the widget with the `componentType` property. This property is mandatory to identify the external widget that will be used for this field.

The `scrollbars` and `stretch` properties can be used to define the behavior of the widget regarding sizing.

The `properties` property is typically used to define properties that are specific to a given WebComponent. For example, a chart component might have properties to define x-axis and y-axis labels.

The value of a WebComponent field is usually (XML) formatted, and holds the data that will be rendered by the external widget through the JavaScript™ shell.

**Note:** In order for a WebComponent to be listed in the `componentType` list, it must be described in an XML file contained in the directory that you specified in GSTWCDIR.

See the topic on WebComponents in the Genero Business Development Language User Guide for more information.

**Related concepts**
- **GSTWCDIR** on page 168

The GSTWCDIR environment variable defines the directory in which the WebComponent XML files (.wcsettings) and optional image files are stored. By default this environment variable is set in the Web Components environment set.

**Add a WebComponent to a form**
Each WebComponent widget is described in an XML description file having the extension .wcsettings.

All the description files must be stored in a single directory. This directory must also contain any icon files to be displayed on the form design page for the WebComponent. Specify the directory using the GSTWCDIR on page 168 environment variable in a Web Component environment set. See Environment sets on page 164. Once you have set this directory, you may add the widgets to your form design documents.

1. Select **Widget > WebComponent** from the Genero Studio menu and add the widget to the form.
2. Choose the specific WebComponent from the combobox for the `componentType` property.

When you add a WebComponent to your form design, a standard image is displayed in the form, indicating that the object is a WebComponent. The `componentType` property in the WebComponent section of the Properties View allows you to select the specific WebComponent that you wish to add. Once you select the component, the properties specific to that component are also listed in the Properties view; if an icon associated with this component has been created for the form design page, it will be displayed.
WebComponents and Form Designer

To make Web Components available for Genero Studio form designers, the description of the Web Component must be described in an XML file with the extension .wcsettings.

Create a separate XML file to describe each WebComponent using the format name.wcsettings; storecalendar.wcsettings for example. The componentType property in the Properties View is a combobox that displays the names of the available WebComponent widgets ("storecalendar" for example).

Tip: Update the preferences for the Form Designer in order to generate the .wcsettings file for WebComponents included in imported text form (.per) files. See Form Designer preferences on page 528.

You can display an icon on the form design document when a web component is selected; otherwise the default icon is displayed. The name of the icon file must be the same as that of the wcsettings file. For example, the storecalendar WebComponent would have storecalendar.wcsettings and storecalendar.jpg.

The form designer must specify the directory that contains the wcsettings and icon files. See Add a WebComponent to a form on page 516.

WebComponent description files

Example file chart.wcsettings:

```xml
<?xml version="1.0" encoding="utf-8"?>
<WebComponent>
    <DynamicProperty name="type" type="TEXT" label="type"
        description="type of chart"
        initialValue="" />
    <DynamicProperty name="caption" type="TEXT" label="caption"
        description="caption"
        initialValue="" />
    <DynamicProperty name="subcaption" type="TEXT" label="subcaption"
        description="subcaption"
        initialValue="" />
    <DynamicProperty name="xaxisname" type="TEXT" label="X label"
        description="label of X axis"
        initialValue="" />
    <DynamicProperty name="yaxisname" type="TEXT" label="Y label"
        description="label of Y axis"
        initialValue="" />
    <DynamicProperty name="numberPrefix" type="TEXT" label="numberPrefix"
        description="number prefix"
        initialValue="" />
    <DynamicProperty name="labels" type="STRINGLIST" label="labels"
        description="labels"
        initialValue="" />
    <DynamicProperty name="values" type="STRINGLIST" label="value"
        description="values"
        initialValue="" />
</WebComponent>
```

The XML schema description for the .wcsettings file is contained in an XSD file named wcsettings.xsd, located in the Genero Studio installation directory GSTDIR/conf/schema.

Action management (Toolbars, Topmenus)

Information about adding Toolbars and Topmenus to forms.

- Action views on page 517  
- Action defaults on page 518  
- Topmenus on page 519  
- Toolbars on page 520

Action views

Action views on a form such as Toolbars, Topmenus, and buttons, trigger actions defined in your program code.

The value of the name or the action property, must exactly match the name of an action in an ON ACTION clause of an interactive statement, such as MENU.
For example, a ButtonEdit on the form with the action property value set to "search" would trigger the action "search" in this BDL program code, when the MENU statement is active and the user clicks the button.

```
MENU
  ON ACTION search
    CALL find_customer()
  ...
```

**Action defaults**
Action defaults can be used to define the appearance of action views on the form (buttons, menu items, and Toolbar items, for example).

For more information see "Understanding action defaults" in the *Genero Business Development Language User Guide*.

**Add action defaults**
Action defaults can be added to a form.

1. Open form.
2. Right-click the form object in the Structure view and select Add Action Default List from the context menu. The form structure now includes an ActionDefaultList node.
3. Right-click on the ActionDefaultList node in the form structure view and select Add Action Default. Repeat to add additional ActionDefaults.
4. For each ActionDefault, specify the properties associated with text, images, accelerators, and comments. These properties will be automatically applied to any action views on the form that have the same name as the property of the ActionDefault in the list.

**Related concepts**
Properties and Form Designer on page 545
The Properties view lists all properties that can be set for a selected form item. The type of form item determines which properties are relevant; the list of properties changes as you move between different form items.

**Import action defaults**
An existing action defaults file (4ad) can be imported into a form.

1. Open form.
2. Right-click the form object in the Structure view and select Import Action Defaults from the context menu.
3. Navigate the file system to find the action default file (4ad) to be used. Select Open to import the file into your form. The form structure now includes an ActionDefaultList node with ActionDefault nodes for each action default.
4. These properties will be automatically applied to any action views on the form that have the same name as the property of the ActionDefault in the list.

Additional ActionDefaults added to the list will be used for the form, but will not effect the 4ad file imported.

**Related concepts**
Properties and Form Designer on page 545
The Properties view lists all properties that can be set for a selected form item. The type of form item determines which properties are relevant; the list of properties changes as you move between different form items.

**Related tasks**
Add action defaults on page 518
Action defaults can be added to a form.

**Topmenus**

A Topmenu presents a pull-down menu on the form associated with actions defined by the current interactive instruction in the BDL code.

When a Topmenu command is selected by the user, the BDL program triggers the action to which the Topmenu command is bound.

A Topmenu can be defined directly in the form or in a resource file with the extension 4tm.

See the topic on Topmenus in the *Genero Business Development Language User Guide* for more information.

![Diagram of Form Structure tab showing a Topmenu](image)

**Figure 270: Form Structure tab showing a Topmenu**

**Related tasks**

Add a Topmenu to a form on page 519

Adding a Topmenu at the form level will make the Topmenu available only to that form.

Add a Topmenu to a form

Adding a Topmenu at the form level will make the Topmenu available only to that form.

You can only add one top menu to the form. If a top menu exists, this procedure will not work.

**Tip:** To create a Topmenu the can be used for all forms or as needed, create a 4tm file and add it to your project.

1. Open form.
2. Right-click the form object in the **Form Structure** view and select **Add Top Menu** from the context menu. The form structure now includes an **TopMenu** node. Topmenu structure is added to the form design.

3. Build your Topmenu:
   
   - Build your Topmenu directly in the form design by using the icons at the top of the form.
   
   ![Figure 271: Topmenu editor](image)

   **Figure 271: Topmenu editor**

   - Or, right-click on the **Topmenu** node in the form structure view and add the elements of the Topmenu.

<table>
<thead>
<tr>
<th>Menu options</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Group</td>
<td>Adds a TopMenuGroup, the text and parent node for the pull-down menu group.</td>
<td></td>
</tr>
<tr>
<td>Add Command</td>
<td>Adds a TopMenuCommand and specifies the action the menu option must be bound to.</td>
<td></td>
</tr>
<tr>
<td>Add Separator</td>
<td>Adds a TopMenuSeparator, an optional horizontal line to be displayed on the Topmenu, visually separating some of the commands.</td>
<td></td>
</tr>
</tbody>
</table>

   **Note:** The TopMenuCommands's name property must exactly match the name of an action specified in your BDL program.

4. Set properties for each of the elements in the Topmenu.

**Related concepts**

**Form item properties** on page 492
Properties can be set on form items such as containers and widgets to provide information on how the runtime system should display or handle the item.

**Import a Topmenu**
An existing Topmenu (4tm) can be imported into a form.

1. Open form.
2. Right-click the form object in the Structure view and select **Import Top Menu** from the context menu.
3. Navigate the file system to find the Topmenu file (4tm) to be used. Select **Open** to import the file into your form. The form structure now includes a **Topmenu** node with nodes for each of the elements defined in the Topmenu. The Topmenu also appears at the top of the form in the form design window.
4. Set properties for each of the elements in the Topmenu as needed.

   Changes made to the Topmenu will be used for the form, but will not effect the 4tm file imported.

**Related concepts**

**Form item properties** on page 492
Properties can be set on form items such as containers and widgets to provide information on how the runtime system should display or handle the item.

**Toolbars**
A Toolbar presents buttons on the form associated with actions defined by the current interactive instruction in the BDL code.

When a Toolbar button is selected by the user, the program triggers the action to which the Toolbar button is bound.
The Toolbar object appears in the Form Structure View, but it does not appear in the design window. It implements a Toolbar on the form that is displayed to the user.

A Toolbar can be defined directly in a form or in a resource file with the extension .4tb.

See the topic on Toolbars in the Genero Business Development Language User Guide for more information.

---

**Figure 272: Form with Toolbar displayed using the Genero Desktop Client**

**Add a Toolbar**

A Toolbar can be added to a form.

Adding a Toolbar at the form level will make the Toolbar available only to that form. To create a Toolbar the can be used for all forms or as needed, create a .4tb file and add it to your project and program.

1. Open form.
2. Right-click the form object in the Structure view and select **Add Tool Bar** from the context menu.
   
   The form structure now includes an **ToolBar** node.
3. Right-click on the **ToolBar** node in the form structure view and add the elements of the Toolbar.

<table>
<thead>
<tr>
<th>Menu options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add Item</strong></td>
<td>Adds a ToolBarItem node and specifies the action the Toolbar button must be bound to. Note: The ToolBarItem’s name property must exactly match the name of an action specified in your BDL program.</td>
</tr>
<tr>
<td><strong>Add Separator</strong></td>
<td>Adds a ToolBarSeparator, an optional vertical line to be displayed on the Toolbar, visually separating some of the commands.</td>
</tr>
</tbody>
</table>

4. Set properties for each of the elements in the Toolbar.

**Related concepts**

**Toolbars** on page 520

A Toolbar presents buttons on the form associated with actions defined by the current interactive instruction in the BDL code.

**Import a Toolbar**

An existing Toolbar (.4tb) can be imported into a form.

1. Open form.
2. Right-click the form object in the Structure view and select **Import Tool Bar** from the context menu.
3. Navigate the file system to find the Toolbar file (.4tb) to be used. Select **Open** to import the file into your form.
   
   The form structure now includes a **ToolBar** node with nodes for each of the elements defined in the Toolbar.
4. Set properties for each of the elements in the Toolbar as needed.
   Changes made to the Toolbar will be used for the form, but will not effect the .4tब file imported.

**Related concepts**

- **Form item properties** on page 492

Properties can be set on form items such as containers and widgets to provide information on how the runtime system should display or handle the item.

**Styles**

Use styles to centralize properties related to the appearance of user interface elements.

Style properties can define appearance by defining font properties, foreground colors, and background colors. Some style properties are specific to a given class of widgets (like the first day of week in a DATEEDIT).

Styles can be defined directly in the form or in a presentation style file (a resource file with a .4st extension). The presentation style file must be distributed with other runtime files.

For more detail about Genero presentation style files and their use, see *Presentation styles* in the *Genero Business Development Language User Guide*.

**Create a style file**

Create a Genero presentation style file based on a starting template.

Creating a presentation style file (.4st) requires an in-depth understanding of presentation styles and the presentation style file syntax. See *Presentation styles* in the *Genero Business Development Language User Guide*.

1. Select **File > New**.
2. Select the option to create the style file you desire.

<table>
<thead>
<tr>
<th>If you are ...</th>
<th>You should ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>working in a Genero BAM Desktop project and want to create a style file based on dbapp.4st</td>
<td>Select <strong>Genero BAM Desktop &gt; Style (.4st)</strong></td>
</tr>
<tr>
<td>working in a Genero BAM Mobile project and want to create a style file based on dbapp.4st</td>
<td>Select <strong>Genero BAM Mobile &gt; Style (.4st)</strong></td>
</tr>
<tr>
<td>working on a Genero project that does not involve the Business Application Modeler and want to create a style file based on default.4st</td>
<td>Select <strong>Genero &gt; Style (.4st)</strong></td>
</tr>
</tbody>
</table>

**Apply a style to a form item**

Styles defined in a style file (.4st) can be applied to forms and form items for form previews and style dialogs.

Before you begin, you need to have identified which style file you are using. You may have created your own, you may be using a shared style file with other developers, or you may be relying on a default style file. Understand which file you are using before you start this procedure.

**Important**: The styleFile attribute is only used by the Form Designer when previewing the form and for Form Designer-related style dialogs. To apply a style file with your Genero application, you must load the style file using the `ui.Interface.loadStyles` method. See the *Genero Business Development Language User Guide* for more information regarding the use of style files by Genero applications.

1. Specify the style file to use. For a form, this is accomplished by setting the **styleFile property** on the form node.
   a) In the **Form Structure** view, select the **Form** node.
   b) In the **Properties** view, select the **styleFile property**.
   c) Click the ... button to browse for and select the style file to use.

   If no style file is specified in the **styleFile property**, a default style file will be used.
If default.4st is found in the path specified by the FGLRESOURCEPATH environment variable, it will be used.

If you add the default.4st file to a library in your project, and a dependency to this library exists, then the directory is automatically added to FGLRESOURCEPATH.

If no default.4st file is found in FGLRESOURCEPATH, the default.4st file from <FGLDIR>/lib is used.

This provides flexibility in the use of style files. You can:

• Specify the style file in the form file.
• Add a default.4st style file to a group node, and set a dependency to the form application node.
• Put a default.4st in a directory, and add the path to the FGLRESOURCEPATH environment variable of an active environment set.
• Automatically use the FGLDIR/lib/default.4st file.

2. Set the style for each form item.

Tips: You can apply a style to multiple form items at one time. Simply select multiple form items before starting this procedure.

a) Select the form item.
b) In the Properties view, select the style property.
c) Click the ... button.
   The Style Selection dialog displays.
d) Select a style or styles for the form item.
e) Select OK.
   Selected styles are listed, separated by a space, in the style property for the form item.

Related concepts
Style Selection dialog on page 541
The Style Selection dialog displays available styles from the form's style file.

Related tasks
Create a style file on page 522
Create a Genero presentation style file based on a starting template.

Form Designer usage
Information about designing a form with Form Designer.

• Drawing on page 523
• Selecting, moving, resizing on page 524
• Aligning on page 524
• Transforming on page 525

Drawing
Containers and widgets can be drawn onto the form.

Using the Toolbar of containers and widgets, you can add one of more container or widget of a specific type onto the form.

Single-click the Toolbar icon
The item is selected, and one occurrence can be placed on the form.

Double-click the Toolbar icon
The item remains selected until explicitly deselected by clicking in the main window, another icon, or the select
Nesting
All items on the form are in a hierarchy identified in the Structure view. All form items must be nested within a container. Some containers can be nested within other containers. For example, a table container could be nested within a folder container. To place a widget within a container on the form design, draw the widget in the container, or drag the drawn widget into a container. The parent container changes color indicating that the widget can be nested within it. The structure view will show the widget within the parent container. You can also use the structure view to change the parenting of containers to containers and widgets within containers by dragging and dropping the selected item into its new location in the structure.

Selecting, moving, resizing
Form items can be selected individually or in groups, moved, and resized.

Selecting
Select an item
Use the mouse to select an item. A selected item on the form design is displayed with handles around it.

Double-click an item
Double-clicking on an item executes the default action, which depends on the widget. For example, double-clicking on a button label or group opens the Edit text dialog. Double-clicking on a combobox or radiogroup, opens the Edit items dialog.

Multiple selection
To select multiple items, select and item and then hold down the Ctrl key while selecting additional items.

Square selection
While holding down the Shift key, drag a rectangle touching elements to select.

Select all
Right click on the form and select Select all Form Fields. This selects all form fields within the current container.

Moving and resizing
Selected items can be moved with the mouse or by using the keyboard arrows (left, right, up, down). This changes the value of the item's posX and/or posY attributes.

Selected items can be re-sized by holding down the Shift key while using the keyboard arrows. This changes the value of the item's gridWidth and/or gridHeight attributes.

Aligning
- Align widgets on page 524
- Layout containers on page 525

Align widgets
Widgets within the same container may be aligned.

The parent container size is limited to the form's maximum width and height.

Align items by using the mouse to select. Use Ctrl-mouse click to select multiple items. Select an option from the Alignment menu.
Related concepts
- **Alignment menu** on page 533
  The Alignment menu provides options for aligning widgets on a form.
- **Alignment dialog** on page 543
  The Alignment dialogs provide advanced settings for item alignment in the form.

**Layout containers**
If a parent container has more than one child container, the child containers must be grouped and aligned using layouts.

Select the containers to be grouped. Use the Layout menu to group them. The form must be large enough to contain both the child containers in their desired position, as well as the layout container.

- **Horizontal Layout**
  Aligns the containers horizontally in a horizontal layout container.
- **Vertical Layout**
  Aligns the containers vertically in a vertical layout container.
- **Break Layout**
  Removes layout, ungrouping the containers; if you have grouped containers in a layout, you must break the layout in order to move the containers.

**Transforming**
Form items can be transformed from one type to another.

- **Convert a widget** on page 525
- **Convert a container** on page 525
- **Convert to matrix** on page 525

**Convert a widget**
Right-click on the form item to select the menu option **Convert Widget**. Select a new widget from the list.

**Convert a container**
Right-click on the container object in the Form Design window to select the menu option **Convert Container**. Select a new container type and options in the **Convert Container** dialog.

**Convert to matrix**
Right-click on a field to convert it to a matrix container.

**The gsform command**
The gsform tool is used at the command line to compile and import forms, convert .4fd files, extract localized strings, and keep a copy of the .per file created during compilation.

The Genero Studio form definition file (.4fd) can also be compiled from the operating system command line, using the tool gsform, which is located in the GSTDIR/bin directory. The gsform tool executable file should not be moved to another directory.

Use gsform if you want to:
- compile a .4fd file from the command line
- import a .per file into Genero Studio Form Designer format
- convert .4fd files in the old Genero Studio format to the new .4fd format
- extract localized strings
- keep a copy of the .per file created during compilation
gsform Syntax

```
gsform [options] file1[4fd] file2[4fd] ...
```

where `fileN` is a Genero Studio form file, with or without the extension being specified.

- You can specify a single command, with or without options. Not all options work with all commands.
- You can specify options without any commands.
- You can omit commands and options completely. To compile a form, for example, you can simply enter: `gsform formname`.

**Table 172: gsform - Help and version information**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td>Display help.</td>
</tr>
<tr>
<td>-V</td>
<td>Display version information.</td>
</tr>
</tbody>
</table>

Examples:

- `gsform -h` shows help.
- `gsform -V` displays the version.

**Table 173: gsform - Commands**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-m</td>
<td>Extracts localized messages.</td>
</tr>
<tr>
<td></td>
<td>See the Localized Strings topic in the Genero Business Development Language User Guide for more information.</td>
</tr>
<tr>
<td>-c</td>
<td>Converts old format <code>.4fd</code> files to new format.</td>
</tr>
<tr>
<td></td>
<td>Old format <code>filename.4fd</code> files are backed up as <code>filename.bak</code>; new format files are saved as <code>filename.4fd</code>. If the backup fails, the conversion is aborted.</td>
</tr>
<tr>
<td>-clean</td>
<td>Cleans orphan settings.</td>
</tr>
<tr>
<td>-import</td>
<td>Imports a <code>.per</code> file as a Genero Studio <code>.4fd</code> file.</td>
</tr>
</tbody>
</table>

Examples:

- `gsform -m myfile.4fd` extracts localized messages.
- `gsform -c myfile.4fd` converts the form file to a new format.
- `gsform -clean myfile.4fd` cleans to orphan settings.
- `gsform -import myfile.per` imports a `.per` file.
Table 174: gsform options

<table>
<thead>
<tr>
<th>Option</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-keep</td>
<td></td>
<td>Keep the intermediate .per file. By default, intermediate .per files are deleted. The compilation process uses temporary .per files. This option avoids the temporary files deletion so that the file can be read. This can be useful in cases where there are unexpected (internal) errors. <strong>Important:</strong> Form Designer and gsform do not allow compilation when .per files and .4fd files with the same file name are present in the same directory.</td>
</tr>
<tr>
<td>-dbname</td>
<td>DBNAME</td>
<td>Specify the database to use for form compilation. If your form file uses a database, it is strongly recommended that you provide the database name using this option to reduce computation time.</td>
</tr>
<tr>
<td>-i</td>
<td></td>
<td>Ignore and overwrite existing .per file.</td>
</tr>
<tr>
<td>-importTextAsLabel</td>
<td></td>
<td>Convert text strings into labels during a text form import.</td>
</tr>
<tr>
<td>-saveWCSettings</td>
<td></td>
<td>Save WebComponent settings files during text form import.</td>
</tr>
<tr>
<td>-M</td>
<td></td>
<td>Display all compilation messages. See the Compiling Programs topic of the Genero Business Development Language User Guide for additional information.</td>
</tr>
<tr>
<td>-W</td>
<td>WARNINGS</td>
<td>Display warning messages. Adding the parameter all displays all messages.</td>
</tr>
<tr>
<td>-wcDir</td>
<td>WEBCOMPONENTS_DIR</td>
<td>Specify the WebComponents directory path.</td>
</tr>
<tr>
<td>-ag-templatePath</td>
<td></td>
<td>Specify which Application Generator template directory used to load the Application Generator settings (file types, properties).</td>
</tr>
<tr>
<td>-translate</td>
<td>LOCALE</td>
<td>Specify the translation file.</td>
</tr>
</tbody>
</table>

Examples:

- `gsform -M -W all myfile.4fd` compiles with warnings and errors displayed in the output.
- `gsform -keep myfile.4fd` keeps the .per file generated during the form file compilation.

**Localizing your form**

Localized Strings allow you to customize the text displayed by your application, for internationalization or site-specific text.

The `localizedStr` property allows you to set localization for an individual widget or form item, where applicable. Setting the checkbox for localized strings to "True" will mark the related strings for localization.

Once the form definition is completed, you can use the `gsform` utility from the command line to generate a text file of the localized strings. For example:

```
gsform -m myform.4fd > myfile.str
```
The format of the file will be "string key" = "string text". You can copy the file and change the string text as needed for localization. Do not change the string key, as this is generated by Genero Studio.

The localized strings file must be added to a library node in your project in Project Manager, and the library linked to your application node. When the application is built, the localized string will be compiled and linked into your application.

See the topic on Localized Strings in the Genero Business Development Language User Guide.

**Form Designer Reference**

Reference information for Form Designer.

- **Form Designer preferences** on page 528
- **Form tab** on page 531
- **Business Record diagram** on page 494
- **Menus** on page 532
- **Views** on page 534
- **Dialogs** on page 535
- **Properties and Form Designer** on page 545
- **Form Designer error messages** on page 572
- **Business Records error messages** on page 336
- **XML validation error messages** on page 586

**Form Designer preferences**

The Form Designer Preferences page sets the preferences for form elements in Form Designer.

The Form Designer Preferences page consist of two sections.

**New Form**

Under New Form, you specify dimension defaults for a new form.

- **Default width**: Default width, in number of characters.
- **Default height**: Default height, in number of lines.

**Import**

Under Import, you specify the defaults for forms that are imported from the .per format.

**Table 175: Import preferences**

<table>
<thead>
<tr>
<th>Checkbox</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open form after import</td>
<td>If checked, form will be opened in Form Designer after being imported.</td>
</tr>
<tr>
<td>Convert Text into Label</td>
<td>If checked, forms imported will have text converted into a static label.</td>
</tr>
</tbody>
</table>

**Tip:** It is preferred to use static labels instead of text to provide field and widget labels on a form. When you use static labels, you gain the use of all properties supported by the static label item type, to include the use of localized strings and internationalization. It is recommended that you leave this option selected.
<table>
<thead>
<tr>
<th>Checkbox</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create WebComponent settings (.wcsettings) on import of a text form</td>
<td>If checked, creates the .wcsettings file required for each WebComponent on an imported text (.per) form. See WebComponents and Form Designer on page 517. If a .wcsettings file already exists, it is kept unchanged. If the GSTWCDIR environment variable is not set, the .wcsettings file is saved in the source directory.</td>
</tr>
</tbody>
</table>

Form Designer preferences are also specified on these pages:

- Form Elements preferences on page 529
- Database Form preferences on page 529

**Form Elements preferences**
The Form Elements Setting page sets the preferences for form elements in Form Designer.

The Form Elements Settings page consist of two sections.

**Table / Tree Settings**
Under Table/Tree Settings, you specify defaults a table or tree container.

- **Default rows**: Number of rows.
- **Default columns**: Number of columns.
- **Default column width**: Size for a new column, in characters.

**Matrix Settings**
Under Matrix Settings, you specify defaults for a matrix.

- **Default rows**: Number of rows in a matrix.
- **Default columns**: Number of columns in a matrix
- **StepX**: Number of cells between the fields repeated horizontally.
- **StepY**: Number of cells between the fields repeated vertically.

**Related concepts**
Table - organizing on page 501
The Table container defines the presentation of a list of records, bound to a screen record (also called a screen array).

Tree - hierarchy on page 502
The Tree container defines the presentation of a list of ordered records in a tree-view widget.

Matrix on page 503
A Matrix container manages a screen array, usually a repetitive array of fields in the screen layout, each containing identical groups of screen fields.

**Database Form preferences**
The Database Forms Settings page sets the preferences for the placement of elements on a form in Form Designer, when the wizard selects fields from a database using a Form Designer wizard.

The Database Forms Settings page consists of three sections.

**Container**
Specify the default container to be used for the form.
Grid and ScrollGrid

Under **Label and Field Alignment**, click on the icon to identify the default layout of the labels and fields. As you select an icon, a textual description displays to confirm how the labels and fields will align.

Under **Label and Field Properties**, you specify defaults for the labels and fields.

- **Maximum width**: Maximum width of a field or label, in characters.
- **Number of fields**: Number of fields per line.
- **Top border**: Top border, in lines.
- **Bottom border**: Bottom border, in lines.
- **Left margin**: Left margin, in characters.
- **Right margin**: Right margin, in characters.
- **Field gutter**: Number of spaces between a label and field, in characters.
- **Field gap**: Gap between two fields on the same line, in characters.

**Matrix**

Under **Matrix**, you specify whether fields can become matrixes.

- **Repeat**: If selected, the fields can repeat and become matrixes.

**Related concepts**

- **Containers** on page 496
  Containers are used to group items on the form. Every form item must be contained within a container. A parent container can also have child containers. If there are multiple child containers, they must be grouped in a horizontal or vertical layout.

- **Grid - positioning** on page 497
  The Grid container declares a formatted text block defining the dimensions and the positions of the form elements for a unique-record presentation.

- **ScrollGrid - positioning** on page 498
  The ScrollGrid container declares a formatted text block defining the dimensions and the positions of the form elements for a multi-record presentation.

- **Matrix** on page 503
A Matrix container manages a screen array, usually a repetitive array of fields in the screen layout, each containing identical groups of screen fields.

**Form tab**
The Form tab displays the form in design mode.

**Figure 273: Form tab**

**Related concepts**
Forms on page 488
Information about creating and compiling forms.

**Related tasks**
Quick Start: Creating a first form on page 487
This topic will guide you through creating a first form in Form Designer.

**Business Record diagram**
The Business Record diagram is used to define the data set of the form, report, or web service.

![Business Record diagram](image)

**Figure 274: Business Record diagram**

**Related concepts**
- *Business records (data sets)* on page 493
  Business records model the data definition, structure and table relationships of the data used in a form, report, and/or web service. Business records are designed and modified in the Business Records diagram.

**Menus**
Information about Form Designer menus.
- *Form Designer context menu* on page 532
- *Alignment menu* on page 533

**Form Designer context menu**
In Form Designer, each container and widget on a form has a context menu accessible by right-clicking on the object.

The context menu displays actions applicable to the object selected.
Table 176: Right-click Context Menu

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Text</td>
<td>Edit the text property.</td>
</tr>
<tr>
<td>Add Phantom</td>
<td>Add a phantom field.</td>
</tr>
<tr>
<td>Convert Container</td>
<td>Convert container to a different container type.</td>
</tr>
<tr>
<td>H-Layout</td>
<td>Nest container in Horizontal layout.</td>
</tr>
<tr>
<td>V-Layout</td>
<td>Nest container in Vertical layout.</td>
</tr>
<tr>
<td>Break Layout</td>
<td>Remove Horizontal or Vertical layout.</td>
</tr>
<tr>
<td>Select All Form Fields</td>
<td>Select all form fields in the current container.</td>
</tr>
<tr>
<td>Add Page</td>
<td>Add Page to Folder container.</td>
</tr>
<tr>
<td>Add Page Before</td>
<td>Add Page before selected Page.</td>
</tr>
<tr>
<td>Add Page After</td>
<td>Add Page after selected Page.</td>
</tr>
<tr>
<td>Add Record</td>
<td>Add field to existing record or add field to a new record.</td>
</tr>
<tr>
<td>Add Column</td>
<td>Add column to table.</td>
</tr>
<tr>
<td>Add Column Before</td>
<td>Add column to Table before selected column.</td>
</tr>
<tr>
<td>Add Column After</td>
<td>Add column to Table after selected column.</td>
</tr>
<tr>
<td>Convert Widget</td>
<td>Convert widget to a different widget type.</td>
</tr>
<tr>
<td>Convert to Text</td>
<td>Convert widget to a Text widget.</td>
</tr>
<tr>
<td>Convert to Matrix</td>
<td>Convert a formField to matrix.</td>
</tr>
<tr>
<td>Edit Items</td>
<td>Edit the items property.</td>
</tr>
<tr>
<td>Locate in design view</td>
<td>From form structure node, locate and select item in design.</td>
</tr>
</tbody>
</table>

Alignment menu
The Alignment menu provides options for aligning widgets on a form.

Table 177: Alignment Menu

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>Aligned with the leftmost widget selected.</td>
</tr>
<tr>
<td>Right</td>
<td>Aligned with the rightmost widget selected.</td>
</tr>
<tr>
<td>Center</td>
<td>Centered vertically based on the average centering of all selected items.</td>
</tr>
<tr>
<td>Top</td>
<td>Aligned with the topmost widget selected.</td>
</tr>
<tr>
<td>Bottom</td>
<td>Aligned with the bottommost widget selected.</td>
</tr>
<tr>
<td>Middle</td>
<td>Centered horizontally to the average centering of all selected items.</td>
</tr>
<tr>
<td>Advanced horizontal</td>
<td>See Alignment dialog on page 543</td>
</tr>
<tr>
<td>Advanced vertical</td>
<td>See Alignment dialog on page 543</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Distribute Horizontally</td>
<td>Aligned with the topmost widget selected (= top alignment) AND items are evenly distributed.</td>
</tr>
<tr>
<td>Distribute Vertically</td>
<td>Aligned with the leftmost widget selected (= left alignment) AND items are evenly distributed.</td>
</tr>
</tbody>
</table>

### Views
Information about Form Designer views.

- [Properties view on page 534](#)
- [Structure view on page 534](#)

### Properties view
Selecting one item from the design window makes this item current, and its properties are displayed in the **Properties view** and available for updating. Each form item, as well as the form itself, has properties that can be set.

![Properties view](image)

**Figure 275: Properties view**

Multiple properties can be selected using Ctrl-click. If multiple properties have been selected, only the common properties will display in the Properties view.

**Related concepts**

[Properties and Form Designer on page 545](#)

The **Properties** view lists all properties that can be set for a selected form item. The type of form item determines which properties are relevant; the list of properties changes as you move between different form items.

### Structure view
The structure of the form is displayed in the Form Structure view. As you add form items to the Form Design window, the elements are automatically added to a tree structure in the Form Structure view.

Expand the nodes in the tree to display or hide the different form items.

Selecting a form item node also selects the corresponding object in the Form Design window, and displays the item's properties in the Properties view.
Figure 276: Structure View

- **Managed Form**
  Lists a node for each record and for the form.
- **Record(s)**
  Lists a node for each field in the record.
- **Forms**
  Lists a node for each item on the form such as containers and widgets.

Right-click on the Form node in the Structure view to add or import other form elements like a Toolbar, Topmenu, Action Defaults and Styles.

**Dialogs**
Information about Form Designer dialogs.

- **Data Control wizard** on page 535
- **Style Selection dialog** on page 541
- **Convert Container dialog** on page 542
- **Alignment dialog** on page 543
- **Dynamic properties** on page 540

**Data Control wizard**
The Data Control wizard is used to rapidly create a form from a database meta-schema.

Once you have added a database schema to your project, you can add fields for database columns to your form.

Access the Data Control wizard from the Form Designer or Business Application diagram:

- **Form Designer:** Container > Data Control
- **File > New > Design > CRUD Form from Database** or Genero Files > Form from Database
- **BA diagram:** Right-click on a Form entity and select Implement Form from Database
- **BA diagram:** Right-click on a Zoom entity and select Implement Zoom from Database.

**Column selection**
Use the Column selection dialog to choose a database and select fields for the form.

**To use Column Selection**
1. Select the database schema from the list of schemas that you have added to your project.
2. Select a table or view name in the schema list to see its columns in the **Column description** list. If a schema is not available in the list, add it to the project first.
3. Expand and/or filter the Tables list.
4. Select the desired columns and use the right arrows to transfer the columns to the **Selected Fields** list.
5. Use the up/down arrows to rearrange the column order. Use the left arrows to remove columns from the **Selected Fields** list.
6. Click **Next** to continue to the **Container Selection** page.

---

**Figure 277: Data Control Wizard**

**Database**
- Available schemas.

**Tables**
- Tables are listed in alphabetical order.
- Tables that have defined relationships in the schema can be expanded to show the tables to which they relate and how they relate. The **Incoming** arrow and **Outgoing** arrow buttons at the top of the **Tables**
list filter the list to show the outgoing and incoming relationships:

- **Incoming** - Identifies those tables that have a relationship with the selected table where the selected table includes the foreign key(s).
- **Outgoing** - The default filter. Identifies those tables include a foreign key relationship to the selected master table.

The joins are automatically built based on the schema relationships and can be viewed or modified in the Joins and Data order on page 539 page of the wizard.

The **Filter items** field (located at the bottom of the Tables list) allows you to filter by table or category names. Identifies the master table for the selected fields.

**Master table**

**Container selection**

The **Container selection** dialog lets you choose the type of container for the form.
**Container**

Select the container to hold the fields for the database columns:

- **Grid** - positioning on page 497
- **ScrollGrid** - positioning on page 498
- **Table** - organizing on page 501
- **Tree** - hierarchy on page 502

**Grid and ScrollGrid**

Label and Field Alignment:

- Label left and field left
- Label right and field right
- Label left and field right
- Label right and field left

Label and Field Properties:

- **Maximum width** - in characters
- **Number of fields** - number of fields per line
- **Top border** - in lines
- **Bottom border** - in lines
- **Left margin** - in characters
- **Right margin** - in characters
- **Field gutter** - spaces between label and data for a field
- **Field gap** - gap between two fields on the same line

**Matrix**

- **Repeat** - check to repeat the fields
- **Row count** - number of rows in matrix
- **Column count** - number of columns in matrix
Joins and Data order

The **Edit query** dialog allows you to specify the joins between the tables in a record used for a form, report, or service. Joins between tables referenced in the form are set up in the *query* property of the business record.

![Edit query dialog](image)

**Figure 278: Edit query dialog**

**Join**

If two tables will be involved in the query, this section is enabled to enter an SQL Join condition for the SQL Query.

**Additional tables**

Add a table here that does not have any fields in the record. For example, list the orders table here if the orders table is required to enable the join between the customer and items tables.

**Where**

Add additional conditions to the SQL Where clause, to restrict the rows returned by the SQL Query to a subset of the data.

**Order by**

Change the default order in which the SQL Query will return the rows.
**Dynamic properties**

The **Dynamic properties** dialog lets you change the label displayed for a formField or the widget used to display the value.

![Dynamic properties Dialog](image)

**Figure 279: Dynamic properties Dialog**

Select a field's label or widget property to change it.

Click **Finish** and the database columns and container are added to the form.
**Style Selection dialog**

The **Style Selection** dialog displays available styles from the form's style file.

**Figure 280: Style dialog**

<table>
<thead>
<tr>
<th>Name</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>dialog3</td>
<td>actionPanelPosition = &quot;none&quot;; errorMessagePosition = &quot;popup&quot;; ringMenuPosition = &quot;none&quot;;</td>
</tr>
<tr>
<td>dialog4</td>
<td>actionPanelPosition = &quot;bottom&quot;; errorMessagePosition = &quot;popup&quot;; ringMenuPosition = &quot;none&quot;;</td>
</tr>
<tr>
<td>dialog_dbapp</td>
<td>actionPanelPosition = &quot;bottom&quot;; errorMessagePosition = &quot;popup&quot;; ringMenuPosition = &quot;none&quot;;</td>
</tr>
<tr>
<td>naked</td>
<td>actionPanelPosition = &quot;none&quot;; border = &quot;tool&quot;; ringMenuPosition = &quot;none&quot;; statutoryStyle = &quot;yes&quot;;</td>
</tr>
<tr>
<td>viewer</td>
<td>actionPanelPosition = &quot;none&quot;; ringMenuPosition = &quot;none&quot;; sizable = &quot;yes&quot;; statutoryStyle = &quot;yes&quot;;</td>
</tr>
<tr>
<td>emphasize</td>
<td>fontFamily = &quot;sans-serif&quot;; fontSize = &quot;14&quot;; styleName = &quot;emphasize&quot;; styleWidget = &quot;&quot;;</td>
</tr>
<tr>
<td>required</td>
<td>fontWeight = &quot;bold&quot;; styleName = &quot;required&quot;; styleWidget = &quot;&quot;</td>
</tr>
<tr>
<td>noborder</td>
<td>border = &quot;none&quot;; styleName = &quot;noborder&quot;; styleWidget = &quot;&quot;</td>
</tr>
</tbody>
</table>

**Result:**

```
actionPanelPosition = "none"; fontFamily = "sans-serif"; fontSize = "14"; ringMenuPosition = "none"; styleName = "main2";
styleWidget = "Window"; windowType = "normal"
```

Check the styles you want to use. Check or uncheck the file name to select/de-select all styles.

Use the Up/Down arrows to change the priority order of the styles if the same property is defined in several styles selected.

See the Presentation Styles topic in the *Genero Business Development Language User Guide*. 
**Convert Container dialog**

The Convert Container dialog displays available containers to which the selected container can be converted.

**Figure 281: Convert Container dialog**

- **Container**
  - Select the container to hold the fields for the database columns:
    - Grid - positioning on page 497
    - ScrollGrid - positioning on page 498
    - Group - grouping on page 499
    - Table - organizing on page 501
    - Tree - hierarchy on page 502

- **Matrix**
  - Repeat - check to repeat the fields
  - Row count - number of rows in matrix
  - Column count - number of columns in matrix

- **Table/Tree to other conversion**
  - Number of fields - indicates the number of formfield columns on each line
Alignment dialog
The Alignment dialogs provide advanced settings for item alignment in the form.

![Horizontal Alignment dialog](image)

**Figure 282: Horizontal alignment dialog**

**Alignment Types**
- Align tops
- Align bottoms
- Align centers

**Container Alignment**
- Align with parent container

**Spacing**
- Do not use spacing
- Use default spacing
- User defined spacing: [ ]

Choose whether to align tops, bottoms, or centers of selected items.
Check to align selected item(s) with parent container's border.
Choose whether to not use spacing, use default spacing, or define spacing by the number of columns (horizontally) to put between selected items.
Figure 283: Vertical alignment dialog

Alignment Types
Choose whether to align lefts, rights, or centers of selected items.

Container Alignment
Check to align selected item(s) with parent container’s border.

Spacing
Choose whether to not use spacing, use default spacing, or define spacing by the number of lines (vertically) to put between selected items.
Dynamic properties
The **Dynamic properties** dialog lets you change the label displayed for a formField or the widget used to display the value.

![Dynamic properties Dialog](image)

**Figure 284: Dynamic properties Dialog**

Select a field's label or widget property to change it.

Click **Finish** and the database columns and container are added to the form.

Properties and Form Designer
The **Properties** view lists all properties that can be set for a selected form item. The type of form item determines which properties are relevant; the list of properties changes as you move between different form items.

**Tip**: In Genero Studio, the term *property* is used. In Genero BDL, these same items are referred to as **ATTRIBUTES**. In most cases, the names are the same - the *accelerator* property in Genero Studio is the same as the `ACCELERATOR` attribute in Genero BDL.

In many cases, you can get more detailed information about a property (or attribute) by referring to the related topics in the *Genero Business Development Language User Guide*.

**accelerator**
The *accelerator* property defines the **primary** accelerator key of an action default item.

For additional information, see the *ACCELERATOR attribute* topic in the *Genero Business Development Language User Guide*. 
Related concepts
Action defaults on page 518
Action defaults can be used to define the appearance of action views on the form (buttons, menu items, and Toolbar items, for example).

accelerator2
The accelerator2 property defines the secondary accelerator key of an action default item.

For additional information, see the ACCELERATOR2 attribute topic in the Genero Business Development Language User Guide.

Related concepts
Action defaults on page 518
Action defaults can be used to define the appearance of action views on the form (buttons, menu items, and Toolbar items, for example).

accelerator3
The accelerator3 property defines the third accelerator key of an action default item.

For additional information, see the ACCELERATOR3 attribute topic in the Genero Business Development Language User Guide.

Related concepts
Action defaults on page 518
Action defaults can be used to define the appearance of action views on the form (buttons, menu items, and Toolbar items, for example).

accelerator4
The accelerator4 property defines the fourth accelerator key of an action default item.

For additional information, see the ACCELERATOR4 attribute topic in the Genero Business Development Language User Guide.

Related concepts
Action defaults on page 518
Action defaults can be used to define the appearance of action views on the form (buttons, menu items, and Toolbar items, for example).

action
The action property defines the name of the action to be sent to the program when the user activates the form item.

There is not a related ACTION attribute in Genero BDL. For related topics, search on Action views in the Genero Business Development Language User Guide.

aggregate
The aggregate property provides an aggregate summary at the end of the column.

For additional information, see the Aggregate Item definition topic in the Genero Business Development Language User Guide.

aggregateText
The aggregateText property can be used to define the label to be displayed for aggregate fields.

The aggregateText property can be specified at the aggregate field level, or globally at the Table - organizing on page 501 level, to define a label for the whole summary line. When defining the aggregateText property at the aggregate field level, the text will be anchored to the value cell. If the aggregateText property is specified at the Table level, the label will appear on the left in the summary line. When an aggregate text is defined at both levels, the global aggregate text of the table will be ignored.

For additional information, see the AGGREGATETEXT attribute topic in the Genero Business Development Language User Guide.
aggregateType
The aggregateType property defines how the aggregate field value is computed.

PROGRAM specifies that the aggregate value will be computed and displayed by the program code.

An aggregate type different from PROGRAM specifies that the aggregate value is computed automatically:

- **SUM** computes the total of all values of the corresponding numeric column.
- **AVG** computes the average of all values of the corresponding numeric column.
- **MIN** displays the minimum value of the corresponding numeric column.
- **MAX** displays the maximum value of the corresponding numeric column.
- **COUNT** computes the number of rows.

The SUM and AVG aggregate types apply to data types that can be used as operand for an addition, such as INTEGER, DECIMAL, INTERVAL.

The MIN and MAX aggregate types apply to data types that can be compared, such as INTEGER, DECIMAL, INTERVAL, CHAR, DATETIME.

For additional information, see the AGGREGATETYPE attribute topic in the Genero Business Development Language User Guide.

autoNext
The autoNext property causes the cursor to automatically advance during input to the next field when the current field is full.

If data values entered in the field do not meet the requirements of other field properties like INCLUDE or PICTURE, the cursor does not automatically move to the next field but remains in the current field, and an error message displays.

The autoNext property is particularly useful with character fields in which the input data is of a standard length, such as numeric postal codes or the abbreviations in the state table. It is also useful if a character field has a length of 1 because only one keystroke is required to enter data and move to the next field.

For additional information, see the AUTONEXT attribute topic in the Genero Business Development Language User Guide.

autoScale
The autoScale property causes the form element contents to automatically scale to the size given to the item.

For an Image on page 510 this property forces the image to be stretched to fit in the area reserved for the image.

For additional information, see the AUTOSCALE attribute topic in the Genero Business Development Language User Guide.

blink
The blink property specifies whether the field should blink.

This property is provided for backward compatibility. It is valid in conjunction with the color on page 548 property.

For additional information, see the COLOR attribute topic in the Genero Business Development Language User Guide.

buttonTextHidden
The buttonTextHidden property indicates that the labels of the buttons of the element should not be displayed.

Use in a Toolbar definition to hide the labels of buttons.

For additional information, see the BUTTONTEXTHIDDEN attribute topic in the Genero Business Development Language User Guide.
The `case` property forces character input to uppercase or lowercase letters.

Assign the `case` property to a character field when you want the runtime system to convert to uppercase or lowercase letters entered, both on the screen and in the corresponding program variable.

Because uppercase and lowercase letters have different values, storing character strings in one or the other format can simplify sorting and querying a database.

Characters entered by the user are converted in INPUT, INPUT ARRAY, and CONSTRUCT instructions.

The results of conversions between uppercase and lowercase letters are based on the locale settings (LANG). When using single byte runners, the conversion of ASCII characters >127 is controlled by the LC_CTYPE environment variable.

The `case` property is unique to Genero Studio. Genero BDL does not provide a corresponding attribute.

The `century` property specifies how to expand abbreviated one- and two-digit year specifications in a DATE and DATETIME field.

Expansion is based on this setting (and on the year value from the system clock at runtime).

The `century` property can specify any of four algorithms to expand abbreviated years into four-digit year values that end with the same digits (or digit) that the user has entered.

`century` supports the same settings as the DBCENTURY environment variable, but with a scope that is restricted to a single field.

If the `century` and DBCENTURY settings are different, CENTURY takes precedence.

Unlike DBCENTURY, the `century` property is not case sensitive. However, we recommend that you use uppercase letters in the property.

For additional information, see the `CENTURY attribute` topic in the Genero Business Development Language User Guide.

The `colName` property is the name of the database column, the `sqlTabName` property is the name of the database table for formField form items.

There are not explicitly-named COLNAME or SQLTABNAME attributes in Genero BDL. Columns and tables are defined within a form's ATTRIBUTES section. For related topics, search on ATTRIBUTES section in the Genero Business Development Language User Guide.

The `color` property defines the foreground color of the text displayed by a form element.

The color property defines the logical color of a value displayed in a field. Value can be BLACK, BLUE, CYAN, GREEN, MAGENTA, RED, WHITE, and YELLOW.

For backward compatibility, the value an be combined with an intensity keyword: REVERSE, LEFT, BLINK, and UNDERLINE.

For additional information, see the `COLOR attribute` topic in the Genero Business Development Language User Guide.

The `colorCondition` property defines a condition to set the foreground color dynamically, though it is recommended that you use styles to implement conditional colors.

The `colorCondition` property defines the logical color of the text of a field when the value satisfies the conditional expression.

The Expression Editor allows you to create the expression for which the `colorCondition` is evaluated.
The condition in `colorCondition` can only reference the field for which the property is set. The Boolean expression is automatically evaluated at runtime to check when the color property must be set.

There is not a related `COLORCONDITION` attribute in Genero BDL. For related topics, search on `COLOR WHERE` in the *Genero Business Development Language User Guide*.

**Example Expressions**

```
COLOR = GREEN WHERE today
```

To refer to the value in the expression, use the keyword `$VALUE`:

```
COLOR = RED WHERE $VALUE > 100
```

**Related concepts**

- [style](#) on page 567
  
  The `style` property specifies a style for a form element.

**Related tasks**

- [Apply a style to a form item](#) on page 522

  Styles defined in a style file (`4st`) can be applied to forms and form items for form previews and style dialogs.

**columnCount**

The `columnCount` property specifies the number of columns of repeated fields.

Minimum value is 1.

Both `columnCount` on page 549 and `rowCount` on page 564 cannot be 1, otherwise the field would be a standard `formField` and not a repeated field.

This property is found in the [Matrix details](#) section. A matrix element is valid within a grid or scrollgrid container.

There is not a related `COLUMNCOUNT` attribute in Genero Business Development Language (BDL).

**Related concepts**

- [Matrix](#) on page 503
  
  A Matrix container manages a screen array, usually a repetitive array of fields in the screen layout, each containing identical groups of screen fields.

**comment**

The `comment` property defines text that can be shown in the comment line when the form item becomes current or when a button receives the focus.

The most common use of the `comment` property is to give information or instructions to the user. This is particularly appropriate when the field accepts only a limited set of values.

The screen location where the message is displayed depends on external configuration. It can be displayed in the `COMMENT LINE`, or in the `STATUSBAR` when using a graphical user interface.

The `comment` property can, in some situations, be a default `placeholder` on page 563. If the value of the `comment` property cannot be rendered in a dedicated area (typically, on mobile devices), the `comment` property value is used as the default placeholder value.

For additional information, see the `STATUSBAR` and `COMMENT attribute` topics in the *Genero Business Development Language User Guide*.

**completer**

The `completer` property enables autocompletion for the edit field.

Form fields with `COMPLETER` attribute provide suggestions while the end-user types text into the field, it can be used in text edit fields such as `EDIT` and `BUTTONEDIT` item types.
For more information, see the *COMPLETER attribute* topic in the *Genero Business Development Language User Guide*.

**componentType**
The *componentType* property defines a name identifying the external widget.

This property is used to define the type of a *WebComponent* form item.

The value of this property will be mapped to a specific widget definition on the front-end side. See front-end specific documentation related to Web Components.

For additional information, see the *COMPONENTTYPE attribute* topic in the *Genero Business Development Language User Guide*.

**contextMenu**
The *contextMenu* property defines the action default property whether a context menu option must be displayed for an action.

*contextMenu* values:

1. **NO** indicates that no context menu option must be displayed for this action.
2. **YES** indicates that a context menu option must always be displayed for this action, if the action is visible (setActionHidden method).
3. **AUTO** means that the context menu option is displayed if no explicit action view is used for that action and the action is visible (setActionHidden method).

The default is **YES**.

This property applies to the actions defined by the current dialog in the current window.

For additional information, see the *CONTEXTMENU attribute* topic in the *Genero Business Development Language User Guide*.

**Related concepts**

*Action defaults* on page 518

Action defaults can be used to define the appearance of action views on the form (buttons, menu items, and Toolbar items, for example).

**databaseName**
The *databaseName* property specifies the name of the database used by the widgets of the form.

When no databaseName is listed in the property, configure Genero Studio or add your database in the project.

There is not an explicitly-named *DATABASENAME* attribute in Genero BDL. The database is defined within a form's *SCHEMA* section. For related topics, search on *SCHEMA* in the *Genero Business Development Language User Guide*.

**Related concepts**

*Access a database* on page 189

To execute an application that accesses a relational database, Genero Studio uses the Genero runtime system and the specific database client software, which must be properly configured.

**dataType**
The *dataType* property specifies the data type of the item.

For NON_DATABASE formField items, the data type has to be specified. For other formField items, data type is determined from the database column type and cannot be modified.

There is not an explicitly-named *DATATYPE* attribute in Genero BDL. A widgets data type is defined within a form's *ATTRIBUTES* section. For related topics, search on *ATTRIBUTES section* in the *Genero Business Development Language User Guide*. 


**defaultValue**
The `defaultValue` property assigns a default value to a field during data entry.

The effect of the `defaultValue` property depends on the `WITHOUT DEFAULTS` configuration option of the dialog using the form:

- With the `INPUT` statement, form default values have are ignored when using the `WITHOUT DEFAULTS` option. With this option, the runtime system displays the values in the program variables to the screen. Otherwise, the form default values will be displayed when the dialog starts.

- With the `INPUT ARRAY` statement, the form default values are always used for new rows inserted by the user. With `INPUT ARRAY`, the `WITHOUT DEFAULTS` option indicates if the existing program array elements have to be used.

Defaults values can also be specified in the database schema file, for `formFields` defined with database column reference.

If the field is formonly (NON_DATABASE), you must also specify a data type when you assign the default property to a field.

If both the default property and the `required` property are assigned to the same field, the required property is ignored. `DATETIME` and `INTERVAL` literals are not supported.

The related attribute is simply `DEFAULT` in Genero BDL. See the `DEFAULT` attribute topic in the *Genero Business Development Language User Guide*.

**defaultView**
The `defaultView` property defines the action default property whether a default view (i.e. button) must be displayed for a given action.

Valid values:

- **NO** indicates that no default action view must be displayed for this action.
- **YES** indicates that a default action view must always be displayed for this action, if the action is visible (`setActionHidden`).
- **AUTO** means that a default action view is displayed if no explicit action view is used for that action and the action is visible (`setActionHidden`).

The default is **AUTO**.

This property applies to the actions defined by the current dialog in the current window.

For additional information, see the `DEFAULTVIEW` attribute topic in the *Genero Business Development Language User Guide*.

**Related concepts**

**Action defaults** on page 518

Action defaults can be used to define the appearance of action views on the form (buttons, menu items, and Toolbar items, for example).

**disclosureIndicator**
The `disclosureIndicator` property defines whether a disclosure indicator must be shown for the default view of an action.

A disclosure indicator is a drill-down decoration. It provides a visual hint to the user, to show that the selection of the action will drill down in the application screens.

For additional information, see the `DISCLOSUREINDICATOR action attribute` topic in the *Genero Business Development Language User Guide*.

**Display Like**
The `Display Like` property takes column properties defined in the database schema files and applies them to a field.

Specifying this property is equivalent to listing all the properties that are assigned to `table.column` in the database schema file generated from the `syscolatt` table.
Supply the `displayTabName` for the table name and `displayColName` for the column name.

Display properties are automatically taken from the schema file if the field is linked to a `table.column`.

The `Display Like` value is evaluated at compile time, not at runtime. If the database schema file changes, you might need to recompile a program that uses the `LIKE` clause. Even if all of the fields in the form are `FORMONLY`, this property requires the form compiler to access the database schema file that contains the description of `table`.

For additional information, search for `DISPLAY LIKE` in the *Genero Business Development Language User Guide*.

**doubleClick**

The `doubleClick` property defines the name of the action to be sent when the user double-clicks on a Table row.

This property is typically used in a `table` container, to define the action to be sent when the user double-clicks on a row.

For additional information, see the `DOUBLECLICK` attribute topic in the *Genero Business Development Language User Guide*.

**expandedColumn**

The `expandedColumn` property specifies the formField that indicates whether a tree node is expanded (opened). This property is optional.

This property is used in the definition of a `container`, see Tree Views for more details.

For additional information, see the `EXPANDEDCOLUMN` attribute topic in the *Genero Business Development Language User Guide*.

**fieldId**

The `fieldId` property is the internal ID of the formField.

This property is read-only.

**fieldIdRef**

The `fieldIdRef` property is the internal ID of the formField corresponding to the record field.

This property is read-only.

**fieldType**

The `fieldType` property specifies the data type of the item.

Valid values include:

- `TABLE_COLUMN`: the value is defined in terms of a database column. Provide the `sqlTabName`, the name of the database table. Provide the `colName`, the name of the database column.
- `NON_DATABASE`: the value is not defined in terms of a database column.
- `COLUMN_LIKE`: the value is like `NON_DATABASE`, but takes its description from a database column.
- `TABLE_ALIAS`: the value is defined in terms of an alias that is assigned to a database table.

There is not an explicitly-named `FIELDTYPE` attribute in Genero BDL. The field type is defined by the syntax of the entry in the `ATTRIBUTES` section. For related topics, search on `ATTRIBUTES section` in the *Genero Business Development Language User Guide*.

**Related concepts**

`colName` and `sqlTabName` on page 548

The `colName` property is the name of the database column, the `sqlTabName` property is the name of the database table for formField form items.

**fontPitch**

The `fontPitch` property defines the character font type as fixed or variable when the default font is used, though it is recommended that you use styles to define font types.

By default, most front ends use variable width character fonts, but in some cases you might need to use a fixed font.

When using `FIXED`, you force the characters to have a fixed size.
When using VARIABLE, you allow the characters to have a variable size.

For additional information, see the FONTPITCH attribute topic in the Genero Business Development Language User Guide.

**Related tasks**

Apply a style to a form item on page 522

Styles defined in a style file (4st) can be applied to forms and form items for form previews and style dialogs.

**format**

The **format** property controls the format of numeric and date time fields for output displays.

Supply the format-string for the format property.

**Note:**

1. *format-string* is a string of characters that specifies a data display format.
2. You must enclose *format-string* within quotation marks (".
3. If *format-string* is smaller than the field width, you get a compile-time warning, but the form is usable.

The *format* property can be set to define a display format for numeric and date fields. When this property is not used, environment variable settings define the default format. For MONEY and numeric fields such as DECIMAL fields, a global format can be specified with the DBMONEY or DBFORMAT environment variables. For DATE fields, the global format is defined by the DBDATE environment variable.

Understand that the *format* property is applied when displaying program variable data to formFields. In order to control user input with a mask, you must use the picture property instead. The picture property is typically used to specify an input mask for formatted character string fields.

If *format-string* is smaller than the field width, you get a compile-time warning, but the form is usable.

For additional information, see the FORMAT attribute topic in the Genero Business Development Language User Guide.

**Numeric formats**

For DECIMAL, MONEY, SMALLFLOAT, and FLOAT data types, *format-string* consists of a set of place holders that represent digits, currency symbols, thousands and decimal separators. For example, "###.##" defines three places to the left of the decimal point and exactly two to the right, plus a currency symbol at the end of the string.

When used with numeric values, the *format-string* must use normalized place holders described in format. The place holders will be replaced by the elements defined in the DBMONEY or DBFORMAT environment variables.

Field input cannot be supported if the format is not defined with normalized place holders.

If the numeric value is too large to fit in the number of characters defined by the format, an overflow text is displayed (**

If the actual number displayed requires fewer characters than *format-string* specifies, numbers are right-aligned and padded on the left with blanks.

If necessary to satisfy the *format-string* specification, the number values are rounded before display.

**Table 178: Format-string symbols for Numeric data types**

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>The star placeholder fills with asterisks any position that would otherwise be blank.</td>
</tr>
<tr>
<td>&amp;</td>
<td>The ampersand placeholder is used to define the position of a digit, and is replaced by a zero if that position would otherwise be blank.</td>
</tr>
</tbody>
</table>
Character | Description
--- | ---
# | The sharp placeholder is used to define the position of a digit, it is used to specify a maximum width for the resulting string. This wildcard character does not change any blank positions in the display: The character is replaced by a blank if no digit is to be displayed at that position.
< | Consecutive less than characters cause left alignment and define digit positions.
- | Displays a minus sign or a blank at that position. USING displays a minus sign when the expression is lower than zero, and otherwise a blank character. When you group several minus signs in the format string, a single minus sign floats immediately to the left of the displayed number.
+ | Displays a plus or minus sign at that position. USING displays a plus sign when the expression is greater than or equal to zero, and a minus sign when the value is less than zero. When you group several plus signs in the format string, a single plus sign floats immediately to the left of the displayed number.
\( | Displayed as left parenthesis for negative numbers. It is used to display accounting parentheses instead of a minus sign for negative numbers. Consecutive left parentheses display a single left parenthesis to the left of the number being printed.
\) | Displayed as right parenthesis for negative numbers. This wildcard character is used in conjunction with a open brace to display accounting parentheses for negative numbers.
, (comma) | The comma placeholder is used to define the position for the thousand separator defined in DBFORMAT. The thousand separator will only be displayed if there is a number on the left of it.
. (period) | The period placeholder is used to define the position for the decimal separator defined in DBMONEY or DBFORMAT. You can only have one decimal separator in a number format string.
$ | The dollar sign is the placeholder for the front currency symbol defined in DBMONEY or DBFORMAT. When you group several consecutive dollar signs, a single front currency symbol floats immediately to the left of the number being printed. The front currency symbol can be defined in DBFORMAT with more than one character.
@ | The at sign is the placeholder for the back currency symbol defined in DBMONEY or DBFORMAT. Put several consecutive @ signs at the end of the format string to display a currency symbol defined in DBFORMAT with more than one character.

<table>
<thead>
<tr>
<th>FORMAT property</th>
<th>Numeric value</th>
<th>DBFORMAT</th>
<th>Result string</th>
</tr>
</thead>
<tbody>
<tr>
<td>---,--&amp;.&amp;&amp;</td>
<td>-1234.56</td>
<td>:.;,:</td>
<td>-1.234.56</td>
</tr>
<tr>
<td>$---,--&amp;.&amp;&amp;</td>
<td>-1234.56</td>
<td>E.;,:</td>
<td>E -1.234,56</td>
</tr>
<tr>
<td>---,--&amp;.@&amp;</td>
<td>-1234.56</td>
<td>:.;,:E</td>
<td>-1,234.56E</td>
</tr>
</tbody>
</table>

When the user enters numeric or currency values in fields, the runtime system behaves as follows:
• If a symbol is entered that was defined as a decimal separator in DBFORMAT, it is interpreted as the decimal separator.
• For MONEY fields, it disregards any front (leading) or back (trailing) currency symbol and any thousands separators that the user enters.
• For DECIMAL fields, the user must enter values without currency symbols.

Date formats

Table 180: Format-string symbols for DATE and DATETIME data type

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dd</td>
<td>Day of the month as a 2-digit integer.</td>
</tr>
<tr>
<td>ddd</td>
<td>Three-letter English-language abbreviation of the day of the week. For example: Mon, Tue.</td>
</tr>
<tr>
<td>mm</td>
<td>Month as a 2-digit integer.</td>
</tr>
<tr>
<td>mmm</td>
<td>Three-letter English-language abbreviation of the month. For example: Jan, Feb.</td>
</tr>
<tr>
<td>yy</td>
<td>Year, as a 2-digits integer representing the 2 trailing digits.</td>
</tr>
<tr>
<td>yyyy</td>
<td>Year as a 4-digit number.</td>
</tr>
</tbody>
</table>

Any other character is interpreted as a literal and will be displayed as is in the field.

Table 181: Format-string examples and corresponding display formats for a DATE field

<table>
<thead>
<tr>
<th>FORMAT property</th>
<th>Date value</th>
<th>DBDATE</th>
<th>Result string</th>
</tr>
</thead>
<tbody>
<tr>
<td>None (DBDATE applies)</td>
<td>1999–09–23</td>
<td>DMY4/</td>
<td>23/09/1999</td>
</tr>
<tr>
<td>(ddd.) mmm. dd, yyyy</td>
<td>1999–09–23</td>
<td>DMY4/</td>
<td>(Thu.) Sep. 23, 1999</td>
</tr>
</tbody>
</table>

Example

"mm/dd/yyyy"

gridChildrenInParent

The gridChildrenInParent property is used for a container to align its children to the parent container.

By default, child elements of a container are aligned locally inside the container layout cells. With this property, you can force children to be aligned according to the layout cells of the parent container of the container to which you assign this property.

This is useful, for example, when you want to align fields across Group containers inside aGrid.

For additional information, see the GRIDCHILDRENINPARENT attribute topic in the Genero Business Development Language User Guide.
gridHeight, gridWidth
The gridHeight and gridWidth properties define the height and width of the form or form item.

There are not explicitly-named GRIDHEIGHT or GRIDWIDTH attributes in Genero BDL. When using a .per file to define a form, the grid and form dimensions are based on the defined layout. For related topics, search on Grid-based layout in the Genero Business Development Language User Guide.

height
The height property specifies the height of an object.

Specify the units using the heightUnit on page 556 property.

This property is found in the Size section for some form objects, such as tables and trees.

For additional information, see the HEIGHT attribute topic in the Genero Business Development Language User Guide.

heightUnit
The heightUnit property specifies the units used when defining the height of an object.

This property works with the height on page 556 property to define the height of an object.

Valid units include:

- characters
- lines
- pixels
- points

This property is found in the Size section for some form objects, such as tables and trees.

There is not an explicitly-named HEIGHTUNIT attribute in Genero BDL. The unit is defined in the HEIGHT attribute. For additional information, see the HEIGHT attribute topic in the Genero Business Development Language User Guide.

hidden
The hidden property indicates that the element should not be displayed.

hidden property values:

1. true sets the underlying item property to 1.
2. user sets the underlying item property to 2.
3. false sets the underlying item property to 0.

By default, all elements are visible. You can use the hidden property to hide an element, such as a formField or a groupbox. The runtime system handles hidden formFields. If you write an INPUT statement using a hidden field, the field is ignored (as if it was declared as noEntry). Programs may change the visibility of formFields dynamically with the ui.form built-in class.

When set to true, the underlying item property is set to 1. The value 1 indicates that the element is hidden to the user without the possibility of showing the element, for example with the context menu of table headers. In this hidden mode, the unHidable property is ignored by the front end.

When set to user, the underlying item property is set to 2. The value 2 indicates that the element is hidden by default, but the user can show/hide the element as needed. For example, the user can change a hidden column back to visible. Form elements like table columns that are hidden by the user might be automatically reshowed (hidden=0) by the front-end if the program dialog gives the focus to that field for input. In such case the program dialog takes precedence over the hidden property.

When you set for a FormField, the model node gets the hidden property, not the view node. FormFields hidden with the user option (value 2) might be shown anyway if the field is needed by a dialog for input.

For additional information, see the HIDDEN attribute topic in the Genero Business Development Language User Guide.
**idColumn**

The `idColumn` property specifies the formField that contains the identifier of a tree node.

This property is mandatory. This property is used in the definition of a `Tree`, see Tree Views for more details.

For additional information, see the `IDCOLUMN attribute` topic in the *Genero Business Development Language User Guide*.

**image**

The `image` property defines the image resource to be displayed in the form item.

This property exists for form items such as `button`, `buttonEdit`, and `static image`. It includes an `initializer`, which defines how the property's default value is determined.

You must define the `FGLIMAGEPATH` environment variable to centralize image files on the application server in a directory created specifically for images. For portability reasons, it is recommended to use `.png`, `.jpg`, or `.svg` image file formats.

You can also specify an image from a URL (`http` or `https`). The network access to the Web server must exist and network bandwidth must be sufficient to rapidly download the images.

**Supported image formats**

Here is the list of image file formats supported by the different front-ends:

<table>
<thead>
<tr>
<th>Suffix (case insensitive)</th>
<th>Front-ends supporting the file format</th>
</tr>
</thead>
<tbody>
<tr>
<td>.BMP</td>
<td>GDC, GBC, GMA, GMI</td>
</tr>
<tr>
<td>.GIF</td>
<td>GDC, GBC, GMA, GMI</td>
</tr>
<tr>
<td>.ICO</td>
<td>GDC, GBC, GMA, GMI</td>
</tr>
<tr>
<td>.JPG</td>
<td>GDC, GBC, GMA, GMI</td>
</tr>
<tr>
<td>.PNG</td>
<td>GDC, GBC, GMA, GMI</td>
</tr>
<tr>
<td>.SVG</td>
<td>GDC, GBC</td>
</tr>
<tr>
<td>.TIFF</td>
<td>GDC, GBC, GMA, GMI</td>
</tr>
</tbody>
</table>

For additional information, see the `IMAGE attribute` and `Providing the image resource` topics in the *Genero Business Development Language User Guide*.

**imageCollapsed**

The `imageCollapsed` property sets the global icon to be used when a tree node is collapsed.

This property is optional. This property defines the icon to be used for nodes that are collapsed. It overwrites the program array image defined by `imageColumn`, if both are used.

This property is used in the definition of a `Tree`, see Tree Views for more details.

For additional information, see the `IMAGECOLLAPSED attribute` topic in the *Genero Business Development Language User Guide*.

**imageColumn**

The `imageColumn` property defines the formField containing the image of a field.

This property is used in the definition of a `Tree`, see the Tree View page for more details.

The images defined by the `imageCollapsed`, `imageExpanded`, `imageLeaf` properties take precedence over the images defined by the `imageColumn` cell.

For additional information, see the `IMAGECOLUMN attribute` topic in the *Genero Business Development Language User Guide*. 
imageExpanded
The **imageExpanded** property sets the global icon to be used when a tree node is expanded.

This property is optional. This property defines the icon to be used for nodes that are expanded. It overwrites the program array image defined by **imageColumn** on page 557, if both are used.

This property is used in the definition of a Tree - hierarchy on page 502, see Tree Views for more details.
For additional information, see the **IMAGEEXPANDED attribute** topic in the *Genero Business Development Language User Guide*.

imageLeaf
The **imageLeaf** property defines the global icon for leaf nodes of a Tree container.

This property is optional. This property defines the icon to be used for all leaf nodes of the tree. It overwrites the program array image defined by **imageColumn** on page 557, if both are used.

This property is used in the definition of a Tree - hierarchy on page 502, see Tree Views for more details.
For additional information, see the **IMAGELEAF attribute** topic in the *Genero Business Development Language User Guide*.

include
The **include** property specifies acceptable values for a field and causes the runtime system to check the data before accepting an input value.

If the field is formonly, you must also specify a data type when you assign the **include** property to a field.

DATETIME and INTERVAL literals are not supported.
For additional information, see the **INCLUDE attribute** topic in the *Genero Business Development Language User Guide*.

initializer
The **initializer** property allows you to specify an initialization function that will be automatically called by the runtime system to set up the form item.

The initialization function must exist in the program using the form file and must be defined with a ui.Combobox parameter.
For additional information, see the **INITIALIZER attribute** topic in the *Genero Business Development Language User Guide*.

**Related concepts**
**ComboBox** on page 507
A ComboBox is a data handling widget that defines a drop-down list of values, allowing the user to select a value.

**initialPageSize**
The **initialPageSize** property defines the initial page size of a list element.

Some list containers, such as a resizable ScrollGrid container, displays by default a single row, if no other elements in the form layout forces the container size implicitly. For these containers, the **initialPageSize** property can be used to specify a default initial number of rows to be displayed.

For a ScrollGrid, this property only displays when the **wantFixedPageSize** property is not selected.
For more information, see the **INITIALPAGESIZE attribute** topic in the *Genero Business Development Language User Guide*.

**Related concepts**
**wantFixedPageSize** on page 571
The **wantFixedPageSize** property gives a fixed height to a list container.
The ScrollGrid container declares a formatted text block defining the dimensions and the positions of the form elements for a multi-record presentation.

**invisible**
The *invisible* property prevents user-entered data from being echoed on the screen during an interactive statement.

Characters that the user enters in a field with the *invisible* property are not displayed during data entry. Depending on the front end type, the typed characters are displayed using the blank, star, underscore or dot characters.

For additional information, see the *INVISIBLE attribute* topic in the *Genero Business Development Language User Guide*.

**isNodeColumn**
The *isNodeColumn* property specifies the formField that indicates whether a tree node has children.

This property is optional. Even if the program node does not contain child nodes for this tree node, this property may be used, to implement dynamic filling of tree views.

This property is used in the definition of a Tree, see Tree Views for more details.

For additional information, see the *ISNODECOLUMN attribute* topic in the *Genero Business Development Language User Guide*.

**items**
The *items* property defines a list of possible values that can be used by the form item.

This property is not used by the runtime system to validate the field, you must use the *include* property to force the possible values.

You can specify item labels with localized strings.

You can define a NULL value for an item (An empty string is equivalent to NULL).

**Example**

![Items Editor](image)

**Figure 285: Items Editor**

For additional information, see the *ITEMS attribute* topic in the *Genero Business Development Language User Guide*.

**justify**
The *justify* property defines the justification of the content of a field and the alignment of table column headers.

With the justify property, you specify the justification of the content of a field as LEFT, CENTER or RIGHT when the field is in display state.
The justify property can be used with all form item types. Additionally to the field content/data alignment, justify defines the alignment of table column headers indirectly (i.e. table column header follows the alignment of field data). However, column header alignment in tables may not be enabled by default; Check the front-end headerAlignment Style attribute.

For additional information, see the JUSTIFY attribute topic in the Genero Business Development Language User Guide.

**keyboardHint**

The keyboardHint property gives an indication on the kind of data the form field contains, to let the front-end adapt the keyboard accordingly.

The keyboardHint property can be used to give a hint to the front-end, regarding the kind of data the form field will contain. According to this hint, the front-end will open the virtual keyboard adapted to the data type, especially useful when designing application forms for mobile platforms.

Valid values for keyboardHint are:

- **default**: No hint, the only hint is the data type of the program variable bound to the form field.
- **email**: The field is used to enter an e-mail address.
- **number**: The field is used to enter a numeric value.
- **phone**: The field is used to enter a phone number.

For example, when defining a numeric field with the keyboardHint property set to number, the iOS device will display a numeric keyboard when entering data into that field.

![Keyboard Hint Example](image)

**Figure 286: Mobile application using a numeric keyboard**

For additional information, see the KEYBOARDHINT attribute topic in the Genero Business Development Language User Guide.

**keys**

The keys property defines labels to be given to ON KEY or COMMAND KEY action handlers when a field is made current.

By defining a label for a key, the runtime system will show a default button for the key action.

**Note:** This feature is supported for backward compatibility. Consider using action attributes to define accelerator keys and decorate actions.

This property exists for formfields which can get the keyboard focus. It appears in the FormField property group. Click the ellipse to add, delete or edit keys. The Edit Keys property editor opens.
There is not an explicitly-named KEYS attribute in Genero BDL. For related topics, see the KEY attribute topic in the Genero Business Development Language User Guide.

**Related concepts**

Action defaults on page 518
Action defaults can be used to define the appearance of action views on the form (buttons, menu items, and Toolbar items, for example).

**left**
The **left** property specifies whether the field should be left-aligned.

This property is provided for backward compatibility. It is valid in conjunction with the **color** property.

For additional information, see the COLOR attribute topic in the Genero Business Development Language User Guide.

**length**
The **length** property defines the length of a FormField.

For additional information, see the length semantics topics in the Genero Business Development Language User Guide.

**lookup**
The **lookup** property contains a name for a field in a record that is a lookup field.

This property assists in defining composite fields in a lookup. It is used by the Business Application Modeler.

For more information, see Lookup fields on page 299.

**minHeight, minWidth**
The **minHeight** and **minWidth** properties define the minimum height and width of a form.

The **minHeight** and **minWidth** properties are used to define a minimum height of the form/window. This property is set on the Form.

For additional information, see the MINHEIGHT attribute and MINWIDTH topics in the Genero Business Development Language User Guide.
name
The name property identifies the name of the item.

There is not an explicitly-named NAME attribute in Genero BDL. The name is defined by the syntax of the entry for the item in the ATTRIBUTES section. For related topics, search on ATTRIBUTES section in the Genero Business Development Language User Guide.

noEntry
The noEntry property prevents data entry in the field during an INPUT or INPUT ARRAY statement.

For additional information, see the NOENTRY attribute topic in the Genero Business Development Language User Guide.

notEditable
The notEditable property disables the text editor.

For additional information, see the NOTEDITABLE attribute topic in the Genero Business Development Language User Guide.

.notNull
The notNull property specifies that the field does not accept NULL values.

There is not an explicitly-named NOTNULL attribute in Genero BDL. The attribute instead presents as NOT NULL (two words). Search on NOT NULL attributes in the Genero Business Development Language User Guide.

orientation
The orientation property defines whether an element displays vertically or horizontally.

The orientation property is typically used in the definition of a RadioGroup form item, to specify how radio items have to be displayed.

For additional information, see the ORIENTATION attribute topic in the Genero Business Development Language User Guide.

parentIdColumn
The parentIdColumn property specifies the formField that contains the identifier of the parent node of a tree node.

This property is mandatory. This property is used in the definition of a Tree, see Tree Views for more details.

For additional information, see the PARENTIDCOLUMN attribute topic in the Genero Business Development Language User Guide.

PER comments
All comments imported from a per file are grouped in the PER comments properties.

This property does not have a corresponding attribute listed in Genero BDL.

picture
The picture property specifies a character pattern for data entry in a text field, and prevents entry of values that conflict with the specified pattern.

format-string can be any combination of characters, where the characters "A", "#" and "X" have a special meaning.

• The character "A" specifies any letter (alpha-numeric) character at a given position.
• The character "#" specifies any digit character at a given position.
• The character "X" specifies any character at a given position.

Any character different from "A", "X" and "#" is treated as a literal. Such characters automatically appear in the field and do not have to be entered by the user.

The picture property does not require data entry into the entire field. It only requires that whatever characters are entered conform to format-string.

When picture specifies input formats for DATETIME and INTERVAL fields, the form compiler does not check the syntax of format-string, but your form will work if the syntax is correct. Any error in format-string, however, such as an incorrect field separator, produces a runtime error.
The typical usage for the picture property is for (fixed-length) CHAR fields. It is not recommended to use picture for other data types, especially numeric or date/time fields: The current value of the field must always match (i.e. be formatted according to) picture.

Understand that the picture property defines a mask for data entry. In order to format fields when data is displayed to the field, use the FORMAT property instead. FORMAT is typically used for numeric and date fields, while picture is typically used for formatted character string fields requiring input control.

For additional information, see the PICTURE attribute topic in the Genero Business Development Language User Guide.

**placeholder**
The **placeholder** property defines a hint for the user when the field contains no value.

The **placeholder** property can be used to show a hint text in an input field, when the field contains no value. This property is typically used for mobile and web applications, to display a grayed text inside empty input fields. Widgets such as the CheckBox, RadioGroup and Slider do not support the **placeholder** property.

**Note:** The **comment on page 549** property can, in some situations, be a default placeholder.

For additional information, see the PLACEHOLDER attribute topic in the Genero Business Development Language User Guide.

**Related concepts**

**comment** on page 549
The **comment** defines text that can be shown in the comment line when the form item becomes current or when a button receives the focus.

**posX, posY**
The **posX** and **posY** properties define the position of the upper left corner of the form item on the x and y axis of the form.

There are not explicitly-named **POSX** or **POSY** attributes in Genero BDL. When using a .per file to define a form, the grid and form dimensions are based on the defined layout. For related topics, search on Grid-based layout in the Genero Business Development Language User Guide.

**program**
The **program** property can specify an external application program to work with screen fields of data type TEXT or BYTE.

For additional information, see the PROGRAM attribute topic in the Genero Business Development Language User Guide.

**queryEditable**
The **queryEditable** property makes a combobox field editable during a CONSTRUCT statement.

This property is useful when the display values match the real values in the **items** property.

For additional information, see the QUERYEDITABLE attribute topic in the Genero Business Development Language User Guide.

**repeat**
The **repeat** property determines if field repeats.

This property is found in the Matrix details section. A matrix element is valid within a grid or scrollgrid container. When the **repeat** property is set to TRUE, the remaining matrix-related properties are enabled in the Properties view. There is not an explicitly-named REPEAT attribute in Genero BDL.

**Related concepts**
**Matrix** on page 503
A Matrix container manages a screen array, usually a repetitive array of fields in the screen layout, each containing identical groups of screen fields.

required
The required property forces the user to modify the content of a field during an INPUT or INPUT ARRAY statement.

To insist on a non-null entry, use the notNull property instead.

For additional information, see the REQUIRED attribute topic in the Genero Business Development Language User Guide.

reverse
On character terminals, the reverse property displays any value in the field in reverse video (dark characters in a bright field).

For additional information, see the REVERSE attribute topic in the Genero Business Development Language User Guide.

rowCount
The rowCount property specifies the number of lines (rows) of repeated fields.

Minimum value is 1.

Both columnCount on page 549 and rowCount on page 564 cannot be 1, otherwise the field would be a standard formField and not a repeated field.

This property is found in the Matrix details section. A matrix element is valid within a grid or scrollgrid container.

There is not a related ROWCOUNT attribute in Genero BDL.

Related concepts
Matrix on page 503
A Matrix container manages a screen array, usually a repetitive array of fields in the screen layout, each containing identical groups of screen fields.

rowHeight
The rowHeight property forces a specific height for all rows in a Table container.

There is not an explicitly-named ROWHEIGHT attribute in Genero BDL. For information on tables and rows, search on TABLE item type in the Genero Business Development Language User Guide.

Related concepts
Table - organizing on page 501
The Table container defines the presentation of a list of records, bound to a screen record (also called a screen array).

sample
The sample property defines the text to be used to compute the width of a formField.

By default, formFields are rendered by the client with a size determined by the current font and the number of characters used in the layout grid. The field width is computed so that the largest value can fit in the widget.

Sometimes the default computed width is too wide for the typical values displayed in the field. For example, numeric fields usually need less space as alphanumeric fields. If the values are always smaller, you can use the sample property to provide a hint for the front end to compute the best width for that formField.

When specifying the sample property, you do not have to fill the sample string up to the width of the corresponding field tag: The front-ends will be able to compute a physical width by applying a ratio to fit the best visual result. For example, for a sample of 'XY' used for a field defined with 10 characters, is equivalent to specifying a sample of 'XYYYYYYYYY'.

For additional information, see the SAMPLE attribute topic in the Genero Business Development Language User Guide.
**Related concepts**

defaultValue on page 551

The `defaultValue` property assigns a default value to a field during data entry.

**scroll**

The `scroll` property can be used to enable horizontal scrolling in a character field with character input.

For additional information, see the `SCROLL attribute` topic in the *Genero Business Development Language User Guide*.

**scrollbars**

The `scrollbars` property can be used to specify scrollbars for a form item.

This property defines scrollbars for the form item, such as a `TextEdit` on page 514. Options are vertical, horizontal, or both.

For additional information, see the `SCROLLBARS attribute` topic in the *Genero Business Development Language User Guide*.

**sizePolicy**

The `sizePolicy` property is a sizing directive to display form elements.

This property defines how the front-ends will compute the size of some form elements in grids. The `sizePolicy` applies only to leaf elements, not to containers. The default value of `sizePolicy` is `initial`.

The `sizePolicy` property is ignored for the widgets used in `Table` and `Tree` columns, because in tables, the size policy is implicitly defined by the cell as fixed (i.e. the size of the column in the form layout).

When the `sizePolicy` is `fixed`, the form elements size is exactly the one defined in the Form Specification File. The size of the element is computed from the width and height in the form grid and the font used on the front-end side.

When `sizePolicy` is `dynamic`, the size of the element grows and shrinks according to the width of the wider during the life time of the application. This can be used for `ComboBox` or `RadioGroup` fields, when the size of the widget must fit exactly to its content, which can vary during the program execution. With `sizePolicy=DYNAMIC`, `Buttons`, `Labels`, `CheckBoxes`, `Images`, and `RadioGroups` can shrink and grow all the time, while `ComboBoxes` can only grow.

When `sizePolicy` is `initial`, the size is computed the first time the element appears on the screen. Once the widget is displayed, its size is frozen. This is typically used when the size of the element must be fixed but is not known at design time (for instance, when populating a `ComboBox` item list from a database table). This option is also useful when the text of labels is unknown at design time because of Internationalization. With `sizePolicy=initial`, the behavior differs depending on the form element type. Keep in mind that after the first display, the element size will be frozen:

- **Buttons**: The size defined in the form is a minimum size. If the text is bigger, the size grows (width and height).
- **ComboBoxes**: The width defined in the form is a minimum width. If one of the items in the value list is bigger, the size grows in order for the combobox to display the largest item fully.
- **Labels, CheckBoxes, and RadioGroups** can shrink or grow. The size defined in the form is ignored. The fields are sized according to the element text.
- **Images** can shrink and grow according to the picture displayed. Images can use the `stretch` property, so that the size of the widget can be dependant from the parent container, overriding the `sizePolicy` property. If the width and height properties have to be used, the `sizePolicy` property must be set to `FIXED`.
- **Other items** such as `Edit` or `widget without items` like `ProgressBar` are not sensitive to the `sizePolicy` property.

The `sizePolicy` property is supported for `WebComponent` fields, however as the content and behavior is defined by the front-end, this property may have no effect. See the front-end web-component specific documentation for more details.

For additional information, see the `SIZEPOLICY attribute` topic in the *Genero Business Development Language User Guide*. 
sliderOrientation
The sliderOrientation property determines the vertical or horizontal orientation of the slider.

There is not an explicitly-named SLIDERORIENTATION attribute in Genero BDL; it is called the ORIENTATION attribute. For related topics, search on SLIDER item type and ORIENTATION attribute in the Genero Business Development Language User Guide.

spacing
The spacing property is a spacing directive to display form elements.

This property defines the global distance between two neighboring form elements. In NORMAL mode, the front end displays form elements consistent with the desktop spacing, which is, for example, 6 and 10 pixels on Microsoft™ Windows® platforms. Some overcrowded forms may need to be displayed with less space between elements, to let them fit to the screen. In this case you can use the COMPACT mode.

By default, forms are displayed with COMPACT spacing.

For additional information, see the SPACING attribute topic in the Genero Business Development Language User Guide.

splits
The splits property forces the container to use a splitter widget between each child element.

This property indicates that the container (typically, HBox and VBox) must have a splitter between each child element held by the container. If a container is defined with a splitter and if the children are stretchable (like Table or TextEdit), users can re-size the child elements inside the container.

For additional information, see the SPLITTER attribute topic in the Genero Business Development Language User Guide.

step
The step property specifies how a value is increased or decreased in one step (by a mouse click or key up/down).

This property is typically used with form items allowing the user to change the current integer value by a mouse click like Slider and SpinEdit.

For additional information, see the STEP attribute topic in the Genero Business Development Language User Guide.

stepX
The stepX specifies the number of characters (COLUMNS) between 2 repetitions.

This property is relevant only if columnCount on page 549 > 1.

This property is found in the Matrix details section. A matrix element is valid within a grid or scrollgrid container.

There is not an explicitly-named STEPX attribute in Genero BDL. For related topics, search on GRID and SCROLLGRID in the Genero Business Development Language User Guide.

Related concepts
Matrix on page 503
A Matrix container manages a screen array, usually a repetitive array of fields in the screen layout, each containing identical groups of screen fields.

stepY
The stepY specifies the number of characters (LINES) between 2 repetitions.

This property is relevant only if rowCount on page 564 > 1.

This property is found in the Matrix details section. A matrix element is valid within a grid or scrollgrid container.

There is not an explicitly-named STEPY attribute in Genero BDL. For related topics, search on GRID and SCROLLGRID in the Genero Business Development Language User Guide.

Related concepts
Matrix on page 503
A Matrix container manages a screen array, usually a repetitive array of fields in the screen layout, each containing identical groups of screen fields.

**stretch**
The *stretch* property specifies how a widget must re-size when the parent container is re-sized.

This property is typically used with form items that can be re-sized like Image or TextEdit fields. By default such form items have a fixed width and height, but in some cases you may want to force the widget to re-size vertically, horizontally, or in both directions.

For additional information, see the *STRETCH attribute* topic in the *Genero Business Development Language User Guide*.

**style**
The *style* property specifies a style for a form element.

This property specifies a presentation style to be applied to a form element. The presentation style can define decoration properties such as a background color, a font type, and so on.

For additional information, see the *STYLE attribute* topic in the *Genero Business Development Language User Guide*.

**Related tasks**
Apply a style to a form item on page 522
Styles defined in a style file (*4st*) can be applied to forms and form items for form previews and style dialogs.

**styleFile**
The *styleFile* property specifies a style file to apply to a form for form previews and style dialogs.

**Important:** The *styleFile* attribute is only used by the Form Designer when previewing the form and for Form Designer-related style dialogs. To apply a style file with your Genero application, you must load the style file using the `ui.Interface.loadStyles` method. See the *Genero Business Development Language User Guide* for more information regarding the use of style files by Genero applications.

**Related tasks**
Apply a style to a form item on page 522
Styles defined in a style file (*4st*) can be applied to forms and form items for form previews and style dialogs.

**tabIndex**
The *tabIndex* property defines the tab order for a form item.

This property can be used to define the order in which the form items are selected as the user "tabs" from field to field when the program is using the *formField order option*.

It can also be used to define which field must get the focus when a *Page* is selected.

By default, form items get a tab index according to the order in which they appear on the form.

**Tip:** *tabIndex* can be set to zero in order to exclude the item from the tabbing list. The item can still get the focus with the mouse.

For additional information, see the *TABINDEX attribute* topic in the *Genero Business Development Language User Guide*.

**Related concepts**
Tab index on page 492
The Tab Index indicates the order of the form items for data entry when using the Tab key to move from field to field.

**tag**
The *tag* property can be used to identify the form item with a specific string.

This property is used to identify form items with a specific string. It can be queried in the program to perform specific processing.
You are free to use this property as you need. For example, you can define a numeric identifier for each field in the form in order to show context help, or group fields for specific input verification.

If you need to handle multiple data, you can format the text, for example, by using a pipe separator.

For additional information, see the `TAG attribute` topic in the *Genero Business Development Language User Guide*.

**text**
The `text` property defines the label associated with a form item, such as the text of a checkbox item.

For additional information, see the `TEXT attribute` topic in the *Genero Business Development Language User Guide*.

**title**
The `title` property defines the title of a form item used in a table container.

For additional information, see the `TITLE attribute` topic in the *Genero Business Development Language User Guide*.

**totalRows**
The `totalRows` property defines the total number of rows in a Table container.

There is not an explicitly-named `TOTALROWS` attribute in Genero BDL. For information on tables and rows, search on *Controlling table rendering* in the *Genero Business Development Language User Guide*.

**Related concepts**

- **Table - organizing** on page 501
The Table container defines the presentation of a list of records, bound to a screen record (also called a `screen array`).

**underline**
The `underline` property specifies whether the field should be underlined

This property is provided for backward compatibility. It is valid in conjunction with the `color` on page 548 property.

For additional information, see the `COLOR attribute` topic in the *Genero Business Development Language User Guide*.

**unHidable**
The `unHidable` property indicates that the element cannot be hidden or shown by the user with the context menu.

By default, a Table container allows the user to hide the columns by a right-click on the column header. Use this property to prevent the user from hiding a specific column.

For additional information, see the `UNHIDABLE attribute` topic in the *Genero Business Development Language User Guide*.

**unhidableColumns**
The `unhidableColumns` property indicates that the columns of the table cannot be hidden or shown by the user with the context menu.

Same effect as `unHidable`, but at the Table level, to make all columns not hideable.

For additional information, see the `UNHIDABLECOLUMNS attribute` topic in the *Genero Business Development Language User Guide*.

**unMovable**
The `unMovable` property prevents the user from moving a defined column of a table.

By default, a Table container allows the user to move the columns by dragging and dropping the column header. Use this property to prevent the user from changing the order of a specific column. Typically, `unMovable` is used on at least two columns to prevent the user from changing the order of the input on these columns.

For additional information, see the `UNMOVABLE attribute` topic in the *Genero Business Development Language User Guide*.
unmovableColumns
The unmovableColumns property prevents the user from moving columns of a table.

By default, a Table container allows the user to move the columns by dragging and dropping the column header. Use this property to prevent the user from changing the order of columns.

For additional information, see the UNMOVABLECOLUMNS attribute topic in the Genero Business Development Language User Guide.

unSizable
The unSizable property prevents the user from resizing the element.

By default, a Table container allows the user to re-size the columns by a drag-click on the column header. Use this property to prevent a re-size on a specific column.

For additional information, see the UNSIZABLE attribute topic in the Genero Business Development Language User Guide.

unsizableColumns
The unsizableColumns property prevents the user from resizing columns of the table.

By default, a Table container allows the user to size the columns. Use this property to prevent the user from sizing any of the columns in the table.

For additional information, see the UNSIZABLECOLUMNS attribute topic in the Genero Business Development Language User Guide.

unSortable
The unSortable property prevents the user from sorting on a specific column.

By default, a Table container allows the user to sort the columns by a left-click on the column header. Use this property to prevent a sort on a specific column.

For additional information, see the UNSORTABLE attribute topic in the Genero Business Development Language User Guide.

unsortableColumns
The unsortableColumns property prevents the user from selecting any column of the table for sorting.

By default, a Table container allows the user to sort on the columns. Use this property to prevent the user from sorting on any columns in the table.

For additional information, see the UNSORTABLECOLUMNS attribute topic in the Genero Business Development Language User Guide.

validate
The validate property is an Action Defaults property defining the data validation level for a given action.

The action default property VALIDATE = NO indicates that no data validation must occur for this action. However, current input buffer contains the text modified by the user before triggering the action.

For additional information, see the VALIDATE attribute topic in the Genero Business Development Language User Guide.

Related concepts
Action defaults on page 518
Action defaults can be used to define the appearance of action views on the form (buttons, menu items, and Toolbar items, for example).

Validate Like
The Validate Like property instructs the form compiler to set the field properties that are defined in the .val database schema file for the specified column.

Specifying the Validate Like property is equivalent to writing in the field definition all the properties that are assigned to table.column in the .val database schema file generated from the syscolval table.
Note that .val properties are taken automatically from the schema file if the field is linked to table.column in the field name specification. The Validate Like property is usually specified for FORMONLY fields.

The Validate Like property is evaluated at compile time, not at runtime. If the database schema file changes, you should recompile all your forms.

Even if all of the fields in the form are FORMONLY, the Validate Like property requires the form compiler to access the database schema file that contains the description of table.column.

For additional information, search for VALIDATE LIKE in the Genero Business Development Language User Guide.

**valueChecked**
The valueChecked property defines the value associated with a checkbox item when it is checked.

This property is used in conjunction with the valueUnchecked property to define the values corresponding to the states of a CHECKBOX.

This property is not used by the runtime system to validate the field, you must use the include property to control value boundaries.

See CheckBox for more details.

For additional information, see the VALUECHECKED attribute topic in the Genero Business Development Language User Guide.

**valueMax**
The valueMax property defines a upper limit of values displayed in widgets (such as progress bars).

This property is typically used in ProgressBar, SpinEdit, Slider fields to define the upper limit.

This property is not used by the runtime system to validate the field, you must use the include property to control value boundaries.

For additional information, see the VALUEMAX attribute topic in the Genero Business Development Language User Guide.

**valueMin**
The valueMin property defines a lower limit of values displayed in widgets (such as progress bars).

This property is typically used in ProgressBar, SpinEdit, Slider fields to define the lower limit.

This property is not used by the runtime system to validate the field, you must use the include property to control value boundaries.

For additional information, see the VALUEMIN attribute topic in the Genero Business Development Language User Guide.

**valueUnchecked**
The valueUnchecked property defines the value associated with a checkbox item when it is not checked.

This property is used in conjunction with the valueChecked property to define the values corresponding to the states of a CHECKBOX.

This property is not used by the runtime system to validate the field, you must use the include property to control value boundaries.

See CheckBox for more details.

For additional information, see the VALUEUNCHECKED attribute topic in the Genero Business Development Language User Guide.

**verify**
The verify property requires users to enter data in the field twice to reduce the probability of erroneous data entry.

This property supplies an additional step in data entry to ensure the integrity of your data. After the user enters a value into a verify field and presses RETURN, the runtime system erases the field and requests reentry of the value. The user must enter exactly the same data each time, character for character: 15000 is not exactly the same as 15000.00.
For additional information, see the VERIFY attribute topic in the Genero Business Development Language User Guide.

**version**
The `version` property is used to specify a user version string for the form.

This property specifies a version string to distinguish different versions of a form. You can specify an explicit version string or use the TIMESTAMP keyword to force the form compiler to write a timestamp string into the 42 file.

Typical usage is to specify a version of the form to indicate if the form content has changed. This property is used by the front-end to distinguish different form versions and to avoid reloading window/form settings into a new version of a form.

You should use the TIMESTAMP only during development.

For additional information, see the VERSION attribute topic in the Genero Business Development Language User Guide.

**wantFixedPageSize**
The `wantFixedPageSize` property gives a fixed height to a list container.

By default, the height of a list container (such as a Table or ScrollGrid container) is re-sizeable. Use this property to freeze the number of rows to the number of screen lines defined by the form design.

For additional information, see the WANTFIXEDPAGESIZE attribute topic in the Genero Business Development Language User Guide.

**wantNoReturns**
The `wantNoReturns` property forces a text field to reject newline characters when the user presses the RETURN key.

By default, text fields like TextEdit on page 514 insert a newline (ASCII 10) character in the text when the user presses the RETURN key. As the RETURN key is typically used to fire the accept action to validate the dialog, you can force the field to reject RETURN keys with this property.

The user can still enter newline characters with Shift-RETURN or Control-RETURN, if these keys are not bound to actions.

For additional information, see the WANTNORETURNS attribute topic in the Genero Business Development Language User Guide.

**wantTabs**
The `wantTabs` property forces a text field to insert TAB characters in the text when the user presses the TAB key.

By default, text fields like TextEdit on page 514 do not insert a TAB character in the text when the user presses the TAB key, since the TAB key is used to move to the next field. You can force the field to use TAB keys with this property.

The user can still jump out of the field with Shift-TAB, if this key is not bound to an action.

For additional information, see the WANTTABS attribute topic in the Genero Business Development Language User Guide.

**width**
The `width` property specifies the width of an object.

Specify the units using the `widthUnit` on page 571 property.

This property is found in the Size section for some form objects, such as tables and trees.

For additional information, see the WIDTH attribute topic in the Genero Business Development Language User Guide.

**widthUnit**
The `widthUnit` property specifies the units used when defining the width of an object.

This property works with the `width` on page 571 property to define the width of an object.

Valid units include:
This property is found in the **Size** section for some form objects, such as tables and trees.

There is not an explicitly-named **WIDTHUNIT** attribute in Genero BDL. The unit is defined in the **WIDTH** attribute. For additional information, see the **WIDTH attribute** topic in the *Genero Business Development Language User Guide*.

**windowStyle**

The **windowStyle** property defines the style to be used by the parent window of a form.

The windowStyle property can be used to specify the style of the parent window that will hold the form. This property is specific to the form. Do not confuse with the **STYLE** property, which is used to specify decoration style of the form elements.

For additional information, see the **WINDOWSTYLE attribute** topic in the *Genero Business Development Language User Guide*.

### Form Designer error messages

A list of Form Designer error messages. For messages that are not self-explanatory, additional information is provided.

**Table 183: Form Designer Error messages**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-10001</td>
<td>Field widget used when no database is attached to form.</td>
</tr>
<tr>
<td></td>
<td>The current form does not use a database. The Field element is an abstract Formfield that can exist only when a database is attached to a form (the widgets depends on the database column datatype).</td>
</tr>
<tr>
<td></td>
<td>• Change the form <strong>databaseName</strong> property to the corresponding database.</td>
</tr>
<tr>
<td></td>
<td>• Change the <strong>Formfield widget</strong> property from Field to another widget (Edit, for example).</td>
</tr>
<tr>
<td>GS-10002</td>
<td>Property value %1 not available, upgrade to genero version %2 or later.</td>
</tr>
<tr>
<td></td>
<td>The value is not supported in this version of Genero.</td>
</tr>
<tr>
<td></td>
<td>Change the property value or upgrade Genero.</td>
</tr>
<tr>
<td>GS-10003</td>
<td>Columns with 'aggregate' must have a widget of type 'Edit'.</td>
</tr>
<tr>
<td></td>
<td>The column using an aggregate is not an 'Edit' widget type.</td>
</tr>
<tr>
<td></td>
<td>• Change the widget type to 'Edit'.</td>
</tr>
<tr>
<td></td>
<td>• Remove the aggregate.</td>
</tr>
<tr>
<td>GS-10004</td>
<td>Error due to fgl not set well.</td>
</tr>
<tr>
<td></td>
<td>When using gsform, the compiler is not working correctly.</td>
</tr>
<tr>
<td></td>
<td>• Check whether FGLDIR is set.</td>
</tr>
<tr>
<td></td>
<td>• Check whether BDL is licensed and working.</td>
</tr>
<tr>
<td>GS-10005</td>
<td>Property %1, does not exist in its wcsettings in the WebComponent directory.</td>
</tr>
<tr>
<td></td>
<td>Open the wcsettings file in Code Editor and add the missing property then reload the directory and the form file.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-10006</td>
<td>Invalid WebComponent directory.</td>
</tr>
<tr>
<td></td>
<td>• Check if the WebComponent directory is blank.</td>
</tr>
<tr>
<td></td>
<td>• Check if the directory exists on disk.</td>
</tr>
<tr>
<td></td>
<td>• Check if there are any .wcsettings files present in the directory.</td>
</tr>
<tr>
<td>GS-10007</td>
<td>Property %1 cannot contain '.'. Please upgrade to genero version %2 or later.</td>
</tr>
<tr>
<td>GS-10008</td>
<td>Children are out of container bounds.</td>
</tr>
<tr>
<td></td>
<td>The children of the parent are out of its boundaries.</td>
</tr>
<tr>
<td></td>
<td>• Increase the size of the parent</td>
</tr>
<tr>
<td></td>
<td>• Decrease the size of the children</td>
</tr>
<tr>
<td>GS-10009</td>
<td>Webcomponent '%1' is not defined in the webcomponent directory</td>
</tr>
<tr>
<td>GS-10010</td>
<td>Stack layout not supported.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-10011</td>
<td>Widget shares border or intersects with another widget.</td>
</tr>
<tr>
<td></td>
<td>Widgets share border or overlap with each other.</td>
</tr>
<tr>
<td></td>
<td>• Move them so they no longer intersect or are adjacent.</td>
</tr>
<tr>
<td></td>
<td>• Delete them.</td>
</tr>
<tr>
<td>GS-10012</td>
<td>Invalid widget, upgrade Genero version.</td>
</tr>
<tr>
<td></td>
<td>The SpinEdit, TimeEdit, Slider and Field widgets were not available prior to</td>
</tr>
<tr>
<td></td>
<td>Genero version 2.00. The widget WebComponent was introduced in Genero version</td>
</tr>
<tr>
<td></td>
<td>2.30. The Tree container and Phantom widget were introduced in Genero version</td>
</tr>
<tr>
<td></td>
<td>2.20</td>
</tr>
<tr>
<td></td>
<td>• Change version to corresponding valid version or later.</td>
</tr>
<tr>
<td></td>
<td>• Change widgets to existing widgets with respect to your current version (</td>
</tr>
<tr>
<td></td>
<td>Edit, for example), or do not use these widgets.</td>
</tr>
<tr>
<td>GS-10013</td>
<td>Invalid property, upgrade Genero version.</td>
</tr>
<tr>
<td></td>
<td>The unitWidth/unitHeight property was added in Genero version 2.00. The</td>
</tr>
<tr>
<td></td>
<td>doubleClick/splitter/minWidth/minHeight properties were added in Genero</td>
</tr>
<tr>
<td></td>
<td>version 2.10. The image/contextMenu properties were added in Genero version</td>
</tr>
<tr>
<td></td>
<td>2.20. The valueMin/valueMax properties were added in Genero version 2.21.</td>
</tr>
<tr>
<td></td>
<td>The justify/style properties were added in Genero version 2.30.</td>
</tr>
<tr>
<td></td>
<td>• Reset to a supported property.</td>
</tr>
<tr>
<td></td>
<td>• Change version to corresponding valid version or later.</td>
</tr>
<tr>
<td>GS-10014</td>
<td>'Table', 'Tree', 'ScrollGrid' can not be defined in a ScrollGrid.</td>
</tr>
<tr>
<td></td>
<td>Corresponds to fglform error (-6846): The screen tag 'TABLE'/TREE/'SCROLLGRID'</td>
</tr>
<tr>
<td></td>
<td>can not be defined in a SCROLLGRID.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-10016</td>
<td>Layout required for containers.</td>
</tr>
<tr>
<td></td>
<td>Cannot have two or more containers without a layout in a parent container,</td>
</tr>
<tr>
<td></td>
<td>except for layout tags (containers in grids).</td>
</tr>
<tr>
<td></td>
<td>Layout the containers vertically or horizontally.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| GS-10017 | ScrollGrid and/or Group layout tags cannot be nested.  

Nested grid / ScrollGrid (layout tag ScrollGrid) cannot contain these children: Table, Group, ScrollGrid, GridNested Group /Scrollgrid (layout tag Group) cannot contain containers other than: HRec, Table, Tree  

Re-arrange your elements using layouts. |
| GS-10018 | Matrix in resizable ScrollGrid.  

The formField is a form designer matrix, which cannot fit in a stretchable scrollgrid. Corresponds to fglform error(-6843): A resizable SCROLLGRID requires the definition of exactly one template.  

To fix, convert the form designer matrix to a formField. |
| GS-10019 | Parent container required.  

A container is required between the form and the element: |

**Figure 288: Form without container**  
- Draw a grid or scrollgrid around the element.  
- Move the element into a Grid, Group, scrollGrid. |

**Figure 289: Form with container** |
| GS-10020 | Character not supported in '%1' encoding.  

- Change the characters to a supported one.  
- Change the encoding. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-10021</td>
<td>No element below layout tag before Genero 2.01.</td>
</tr>
</tbody>
</table>

**Figure 290: Form with element inside grid below layout tag**

You cannot have elements inside the grid below a layout tag (container inside grid.)
Move the elements to another grid and layout both grids vertically.

<table>
<thead>
<tr>
<th>GS-10022</th>
<th>Layout tag width is name length plus 4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Increase layout tag width.</td>
<td></td>
</tr>
<tr>
<td>• Shorten name attribute.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GS-10023</th>
<th>Saved WebComponent settings in GSTWCDIR directory.</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is an informational message. The import process has generated .wcsettings file(s) corresponding to the form. If the GSTWCDIR is not well set, the files are saved in the form directory and you must move the file(s) to the correct directory.</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>GS-10026</td>
<td>Non container widgets not beside table in grid before Genero 2.10. Cannot have widget beside layout tag table, this feature has been implemented in version 2.10.</td>
</tr>
</tbody>
</table>

**Figure 291: Form with widget beside layout tag table**

- Change Genero configuration fgl to 2.10 or later
- Move the widgets into another grid and layout both grids horizontally

**Figure 292: Form with widgets into another grid and layout both grids horizontally**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-10027</td>
<td>Empty container. This node is missing its required children. Item is missing in the TopMenu or TopMenuGroup or Toolbar or ActionDefaultList. Widget is missing in group, hrec etc</td>
</tr>
</tbody>
</table>
- Add an valid child.
- Remove the container.
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-10028</td>
<td>ActionDefault %1 doesn't contain modified attributes applicable to current Genero version. ActionDefaultGroup is defined without any attribute applicable to current Genero version.</td>
</tr>
<tr>
<td></td>
<td>• Delete it.</td>
</tr>
<tr>
<td></td>
<td>• Set one of these attributes: title, image, accelerators ...</td>
</tr>
<tr>
<td></td>
<td>• The <strong>contextmenu</strong> attribute is available since version Genero 2.20, but not before that.</td>
</tr>
<tr>
<td></td>
<td>• The <strong>validate</strong> attribute is available since Genero 2.1x, not before that.</td>
</tr>
<tr>
<td></td>
<td>• The <strong>Accelerator4</strong> is available since Genero 2.00, not before that.</td>
</tr>
<tr>
<td>GS-10029</td>
<td>Redundant table records.</td>
</tr>
<tr>
<td></td>
<td>There are two or more records of the table container that are exactly the same, i.e. they have the same order of recordFields.</td>
</tr>
<tr>
<td></td>
<td>• Reorder the recordField(s) in record that is duplicate.</td>
</tr>
<tr>
<td></td>
<td>• Delete the duplicate record.</td>
</tr>
<tr>
<td>GS-10031</td>
<td>Left edge cannot be shared with table right edge.</td>
</tr>
<tr>
<td></td>
<td>Widget left edge cannot be shared with table right edge. For any Genero version, Genero form compiler needs a space to the right of the table.</td>
</tr>
<tr>
<td></td>
<td>Enter a space to the right of the table.</td>
</tr>
<tr>
<td>GS-10033</td>
<td>Requires minimum location.</td>
</tr>
<tr>
<td>GS-10034</td>
<td>Requires minimum size.</td>
</tr>
<tr>
<td>GS-10035</td>
<td>Old widget '%1' was removed during import.</td>
</tr>
<tr>
<td></td>
<td>An old widget (canvas) that is no longer supported was present in the imported form and removed from the document.</td>
</tr>
<tr>
<td></td>
<td>Create another element for this widget, or ignore the error.</td>
</tr>
<tr>
<td>GS-10036</td>
<td>Old widget %1 was transformed to %2 during import.</td>
</tr>
<tr>
<td></td>
<td>An old widget that is no longer supported was present in the imported form. It's been transformed into the form element.</td>
</tr>
<tr>
<td></td>
<td>Check that the transformation was correct, or fix it (some attributes may be incorrectly set).</td>
</tr>
<tr>
<td>GS-10040</td>
<td>Expected ValueChecked to be different from valueUnChecked.</td>
</tr>
<tr>
<td></td>
<td>Checkbox valueChecked and valueUnchecked attributes cannot have the same value.</td>
</tr>
<tr>
<td></td>
<td>Change one of the attribute values.</td>
</tr>
<tr>
<td>GS-10042</td>
<td>Requested operation cannot be performed, please check the Genero configuration settings.</td>
</tr>
<tr>
<td></td>
<td>The Genero configuration is not valid.</td>
</tr>
<tr>
<td></td>
<td>Check Genero license, directory, and Genero Desktop Client.</td>
</tr>
<tr>
<td>GS-10044</td>
<td>Internal error.</td>
</tr>
<tr>
<td></td>
<td>An unexpected internal error occurred during compilation.</td>
</tr>
<tr>
<td></td>
<td>Contact your local support Center.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-10045</td>
<td>Could not delete temporary per file.&lt;br&gt;A temporary per file cannot be deleted during the compilation process, probably because the user opened the file or changed the access rights.&lt;br&gt;Close the temporary file, or set the access rights to read write.</td>
</tr>
<tr>
<td>GS-10046</td>
<td>Cannot generate unique tagName, gridwidth is too small.&lt;br&gt;All available field identifiers are used during per file generation.&lt;br&gt;Enlarge some field's gridwidth.</td>
</tr>
<tr>
<td>GS-10047</td>
<td>Unable to create per file during form compilation.</td>
</tr>
<tr>
<td>GS-10049</td>
<td>Value not compatible with data Type.&lt;br&gt;The Value is not compatible with data type set. This error occurs when defaultValue and include property value are incompatible with dataType property.&lt;br&gt;Change defaultValue or include; enter a value compatible to data type format.</td>
</tr>
<tr>
<td>GS-10052</td>
<td>Record cannot start with phantom field.&lt;br&gt;Rearrange record fields.</td>
</tr>
<tr>
<td>GS-10054</td>
<td>idColumn/parentIdColumn must be defined for Tree.&lt;br&gt;Set a valid value, i.e. a name of one of the tree's children, for these properties.</td>
</tr>
<tr>
<td>GS-10055</td>
<td>Invalid value set for %1 property.&lt;br&gt;Check the documentation for the property, and set a valid value. For example, a valid value for the name property is comprised of alphanumeric characters and underscores, but can not start with a number:</td>
</tr>
<tr>
<td></td>
<td>• For accelerator, accelerator2, accelerator3, accelerator4; value should be a valid accelerator name.</td>
</tr>
<tr>
<td></td>
<td>• For imageColumn, idColumn, parentIdColumn, expandedColumn, isNodeColumn; value should be one of the children of the table/tree.</td>
</tr>
<tr>
<td></td>
<td>• For expandedColumn, isNodeColumn, imageColumn; value could be blank, i.e. reset to default.</td>
</tr>
<tr>
<td></td>
<td>• For Action; value should be a valid action name.</td>
</tr>
<tr>
<td></td>
<td>• For FieldType; the Field widget cannot be used with FORM_ONLY or FORM_LIKE fieldType; the expected fieldType is TABLE_COLUMN or TABLE_ALIAS.</td>
</tr>
<tr>
<td></td>
<td>• For formFields; value should be tableName.colName.</td>
</tr>
<tr>
<td></td>
<td>• For formonly formFields; value should be formonly.name.</td>
</tr>
<tr>
<td></td>
<td>• For tableAlias type formFields; value should be tableAliasName.colName.</td>
</tr>
<tr>
<td></td>
<td>• For non-formFields; value should be a valid name.</td>
</tr>
<tr>
<td>GS-10056</td>
<td>Include property doesn't support dataType.&lt;br&gt;An include string is set while dataType is not supported (for example, interval). Change data type, or remove include string.</td>
</tr>
<tr>
<td>GS-10057</td>
<td>Value is out of data Type range.&lt;br&gt;Some data types have a limitation of size (for example, integer). Change data type (for example, from integer to bigint), or modify value so that it is valid.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-10061</td>
<td>Screen record array has different component sizes.</td>
</tr>
<tr>
<td></td>
<td>Matrices corresponding to the Screen Record need to have the same repeat count (number of rows * number of columns).</td>
</tr>
<tr>
<td></td>
<td>• Change the row count or the column count.</td>
</tr>
<tr>
<td></td>
<td>• If the current screen record is an array and any of the screenRecordItems correspond to a FormField, convert it to a matrix.</td>
</tr>
<tr>
<td></td>
<td>• If the current screen record is not an array, and any of the screenRecordItems correspond to a Matrix, convert it to a FormField.</td>
</tr>
<tr>
<td>GS-10062</td>
<td>All members of \Record\ must reference the same container.</td>
</tr>
<tr>
<td></td>
<td>The formFields corresponding to the record field in the same record must belong to the same container.</td>
</tr>
<tr>
<td></td>
<td>• Move the record field to the correct record.</td>
</tr>
<tr>
<td></td>
<td>• Move the formField to the right container.</td>
</tr>
<tr>
<td></td>
<td>• Remove the record field or formField.</td>
</tr>
<tr>
<td>GS-10063</td>
<td>First tree column must have a widget of type Edit or Label.</td>
</tr>
<tr>
<td></td>
<td>The first Tree widget column is not an Edit or a Label.</td>
</tr>
<tr>
<td></td>
<td>• Insert an Edit or Label as the first column.</td>
</tr>
<tr>
<td></td>
<td>• Change the first column to an Edit or Label.</td>
</tr>
<tr>
<td></td>
<td>• Re-order the columns to have an Edit or Label as the first column.</td>
</tr>
<tr>
<td>GS-10064</td>
<td>Invalid/Duplicate name transformed during import.</td>
</tr>
<tr>
<td></td>
<td>An invalid name that was present in the imported form was transformed to a valid and unique name. A valid name is not blank, and is made up of these characters: A-Z, 0-9, &quot;.&quot;. You may edit the transformed name.</td>
</tr>
<tr>
<td>GS-10065</td>
<td>Multiple lines in Form title.</td>
</tr>
<tr>
<td></td>
<td>Genero clients don't support multiple lines.</td>
</tr>
<tr>
<td></td>
<td>Remove the carriage return.</td>
</tr>
<tr>
<td>GS-10070</td>
<td>Cannot save 4fd file.</td>
</tr>
<tr>
<td></td>
<td>The 4fd file cannot be saved due to file system permissions.</td>
</tr>
<tr>
<td></td>
<td>Check permissions, or move the per file and import it again.</td>
</tr>
<tr>
<td>GS-10076</td>
<td>File not found.</td>
</tr>
<tr>
<td></td>
<td>A file is missing.</td>
</tr>
<tr>
<td>GS-10077</td>
<td>Cannot load file.</td>
</tr>
<tr>
<td></td>
<td>An unexpected error occurred loading the file.</td>
</tr>
<tr>
<td>GS-10079</td>
<td>Cannot import file, %1 already exists.</td>
</tr>
<tr>
<td></td>
<td>The form file already exists, import is canceled (gsform).</td>
</tr>
<tr>
<td></td>
<td>• Delete the 4fd form file</td>
</tr>
<tr>
<td></td>
<td>• Force overwrite (-i option).</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-10080</td>
<td>Cannot import file, wrong extension.</td>
</tr>
<tr>
<td></td>
<td>Check file extension.</td>
</tr>
<tr>
<td>GS-10081</td>
<td>Container is too small, user defined properties will be ignored.</td>
</tr>
<tr>
<td></td>
<td>The layout tagged container is too small to hold identifier. Any user defined properties will be ignored. This is a warning message. Increase width of container.</td>
</tr>
<tr>
<td>GS-10082</td>
<td>Record has only phantom fields.</td>
</tr>
<tr>
<td></td>
<td>• Change widget type of a record field Formfield from phantom to some other type.</td>
</tr>
<tr>
<td></td>
<td>• Drag-drop some record field that is not a phantom type into this record.</td>
</tr>
<tr>
<td></td>
<td>• Delete record.</td>
</tr>
<tr>
<td>GS-10083</td>
<td>valueMin must be lower than valueMax.</td>
</tr>
<tr>
<td></td>
<td>• Change valueMin so that it is less than valueMax.</td>
</tr>
<tr>
<td></td>
<td>• Change valueMax so that it is greater than valueMin.</td>
</tr>
<tr>
<td>GS-10084</td>
<td>Invalid text.</td>
</tr>
<tr>
<td></td>
<td>The text property value cannot contain newline characters, \n or \r.</td>
</tr>
<tr>
<td></td>
<td>Delete or replace these characters.</td>
</tr>
<tr>
<td>GS-10085</td>
<td>Wrong file extension.</td>
</tr>
<tr>
<td></td>
<td>The file chosen for import has an incorrect extension.</td>
</tr>
<tr>
<td></td>
<td>Select a file with a per extension to import.</td>
</tr>
<tr>
<td>GS-10086</td>
<td>Cannot create compilation task.</td>
</tr>
<tr>
<td></td>
<td>Check the build rule; if correct, contact your FourJ's Support center.</td>
</tr>
<tr>
<td>GS-10087</td>
<td>Invalid widget position.</td>
</tr>
<tr>
<td></td>
<td>Change position of widget.</td>
</tr>
<tr>
<td>GS-10088</td>
<td>StepX needs to be greater than 1.</td>
</tr>
<tr>
<td></td>
<td>The property stepX must have a value greater than 1.</td>
</tr>
<tr>
<td></td>
<td>• Edit stepX property.</td>
</tr>
<tr>
<td></td>
<td>• Uncheck repeat property.</td>
</tr>
<tr>
<td>GS-10089</td>
<td>Invalid nested table position.</td>
</tr>
<tr>
<td></td>
<td>Change table position.</td>
</tr>
<tr>
<td>GS-10090</td>
<td>Invalid file, closing tag does not correspond to the opening one.</td>
</tr>
<tr>
<td></td>
<td>The file is invalid; it contains mismatching tags and XML parsing is not possible.</td>
</tr>
<tr>
<td>GS-10091</td>
<td>Invalid file, orphan closing tag.</td>
</tr>
<tr>
<td>GS-10092</td>
<td>Empty space required on all sides of layout tag.</td>
</tr>
<tr>
<td></td>
<td>Minimum of one unit empty space is required on all sides of layout tag containers.</td>
</tr>
<tr>
<td></td>
<td>Re-size layout tag container or its children.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-10095</td>
<td>componentType property must be defined as a name of valid webComponent. Set a valid value from available WebComponents.</td>
</tr>
<tr>
<td>GS-10097</td>
<td>Invalid WebComponent. The WebComponent is invalid as it does not satisfy XSD file rules. Open wcsettings file in code editor and rectify in accordance with XSD validation.</td>
</tr>
<tr>
<td>GS-10098</td>
<td>Invalid WebComponent schema. The WebComponent XSD file that is used for validating WebComponent XML files is invalid. Take a clean copy from of file. Undo all changes done manually.</td>
</tr>
<tr>
<td>GS-10099</td>
<td>Empty text widget will be ignored. The text property value will be ignored as it is blank. This is a warning message. Solution: Since property value is blank, it plays no role, and is ignored.</td>
</tr>
<tr>
<td>GS-10100</td>
<td>Property '%1' conflicts with 'id' or 'parentid'. The value of property 'expandedColumn' or 'isNodeColumn' is same as either 'idColumn' or 'parentIdColumn' of tree widget. Change the property value of either of the properties so that expandedColumn and isNodeColumn do not have the same value as either idColumn or parentIdColumn.</td>
</tr>
<tr>
<td>GS-10101</td>
<td>Invalid Template directory '%1' The BAM template directory (GSTSETUPDIR environment variable) is incorrect. It should be an accessible directory absolute path, and contain settings (such as setting.agconf and/or creatables, for example). The path MUST be for the Genero Studio client, not the Genero Studio server.</td>
</tr>
<tr>
<td>GS-10102</td>
<td>tabindex '%1' is already used in '%2' %1 refers to the tabIndex value, and %2 is the name of the widget having the tabIndex value that is already used. Revisit the values of the tabIndex for all objects in your form, and change the value of the tabIndex to a value that is not being used.</td>
</tr>
<tr>
<td>GS-10103</td>
<td>Property tabindex of '%1' has been reset because it was duplicated. When you open a file with a tabIndex error in Form Designer, the issue discussed in GS-10102 is fixed automatically. This message is the notification. No additional action is required, however you may wish to validate the tabIndex entries for the objects in your form.</td>
</tr>
<tr>
<td>GS-10104</td>
<td>The FDMAP property type is deprecated, use MAP instead. The FDMAP property editor for Form Designer in BAM settings has been renamed as MAP. It is available outside of Form Designer, therefore the FD prefix was removed. Use the MAP property editor instead. The property editor is defined in the settings.agconf settings file, located in the template directory.</td>
</tr>
<tr>
<td>GS-10105</td>
<td>Unknown option '%1'</td>
</tr>
</tbody>
</table>
### Business Records error messages

A list of Business Records error messages. For messages that are not self-explanatory, additional information is provided.

#### Table 184: Business Records Error Messages

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-10300</td>
<td>Need one and only one relation for program %1.</td>
</tr>
<tr>
<td>GS-10301</td>
<td>Old model format does not support BasicItem.</td>
</tr>
<tr>
<td>GS-10900</td>
<td>Cannot load file, unsupported node %1</td>
</tr>
<tr>
<td>GS-10901</td>
<td>File is saved using Genero Studio version %1, current version is %2. Please upgrade Genero Studio.</td>
</tr>
</tbody>
</table>

#### Table 184: Business Records Error Messages (continued)

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-24001</td>
<td>Error loading file file.</td>
</tr>
<tr>
<td></td>
<td>An error occurred loading the file.</td>
</tr>
<tr>
<td></td>
<td>Check the file name and permissions .</td>
</tr>
<tr>
<td>GS-24002</td>
<td>Malformed XML.</td>
</tr>
<tr>
<td></td>
<td>XML file content is invalid.</td>
</tr>
<tr>
<td></td>
<td>Select the correct file or correct the XML.</td>
</tr>
<tr>
<td>GS-24003</td>
<td>File file not found in Business Application diagram (&lt;BA file name&gt;).</td>
</tr>
<tr>
<td></td>
<td>A referenced element is not found in the Business Application diagram.</td>
</tr>
<tr>
<td></td>
<td>Select the correct file.</td>
</tr>
<tr>
<td>GS-24004</td>
<td>Invalid value for %1 property.</td>
</tr>
<tr>
<td></td>
<td>Check the property syntax in the documentation and correct any errors.</td>
</tr>
<tr>
<td>GS-24005</td>
<td>Conflicted item.</td>
</tr>
<tr>
<td></td>
<td>The Business Application diagram and Business Record unique id do not match.</td>
</tr>
<tr>
<td></td>
<td>Open the 4ba file and resolve the conflict.</td>
</tr>
<tr>
<td>GS-24201</td>
<td>Missing master Record.</td>
</tr>
<tr>
<td></td>
<td>The document must contain at least one record that is master.</td>
</tr>
<tr>
<td></td>
<td>Create a record.</td>
</tr>
<tr>
<td>GS-24202</td>
<td>Unique query key must be set.</td>
</tr>
<tr>
<td></td>
<td>At least one field must be declared the unique key in a record.</td>
</tr>
<tr>
<td></td>
<td>Set a unique key.</td>
</tr>
<tr>
<td>GS-24203</td>
<td>Record used in relation must be active.</td>
</tr>
<tr>
<td></td>
<td>Change the record to active, or remove the relation.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-24204</td>
<td>Empty relation. This relation has no field definition. Add foreign/primary (or source/destination) fields.</td>
</tr>
</tbody>
</table>
| GS-24205 | Database table column referenced more than once in the form. In one form, a database table column can be referred only once. Table aliases should be used to attach more than one field to a database.  
• Select one of the wrong Formfields and change the fieldType attribute from table_column to table_alias.  
• Select one of the wrong Formfields and change fieldType attribute to non_database. |
<p>| GS-24206 | Missing master table for record record. Set the master table property.      |
| GS-24207 | Table %1 unused in query. A database table is used in the record, but there is no join for it in the query. Add a join for the table in the record query. |
| GS-24208 | No schema attached. A database is required for the business records. Set the database name property to an existing database. |
| GS-24209 | Invalid relation, primaryField and foreignField must have same number of fields. Modify the relation fields so that a primary field corresponds to each foreign field. |
| GS-24210 | Invalid query, left and right join must have same number of columns. Check the query fields so that a left field corresponds to each right field. |
| GS-24211 | Relation types don't match exactly. The type of a foreign key doesn't match the corresponding primary key's type. Check the relation, change the field types, or fix the primary / foreign key. |
| GS-24212 | Relation fieldfieldname not found. The relation refers a field that is missing in the record. Modify the relation or add the field to the record. |
| GS-24213 | Non existing schema schema attached to document. Check the schema being referenced. |
| GS-24214 | Nonexistent table table referenced in document. The document uses a database table that is not present in the schema. Update the schema, or change the field using the table. |</p>
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-24215</td>
<td>Nonexistent column <code>table.column</code> referenced in document. The document uses a database column that is not present in the schema. Update the schema or change the field using the <code>table.column</code>.</td>
</tr>
<tr>
<td>GS-24216</td>
<td>Database table defined in &quot;no database&quot; document. Set the database property to an existing schema or remove the field.</td>
</tr>
<tr>
<td>GS-24217</td>
<td>Table alias referenced more than once in the form. The same table alias is associated with two columns in the form. • Change one table alias name. • Change either the table or column name.</td>
</tr>
<tr>
<td>GS-24218</td>
<td>Alias <code>alias</code> referenced for different tables in document. The same alias is used for different tables. Rename the alias so that it refers to the same database table.</td>
</tr>
<tr>
<td>GS-24219</td>
<td>Alias Lookup field is ignored on <code>%1</code> as it is the master table of the record. A lookup field is dedicated to foreign field update in the master table; do not set it on a master table column. Remove the lookup property value or update the field database settings.</td>
</tr>
<tr>
<td>GS-24220</td>
<td>Name value <code>%1</code> is already used. Duplicate name used in the document. Rename the element so that the name is unique.</td>
</tr>
<tr>
<td>GS-24221</td>
<td>Invalid INTERVAL qualifier. The <code>qual1</code> or <code>qual2</code> set for the INTERVAL <code>sqlType</code> is not valid. It should belong to the INTERVAL classes (i.e., YEAR-MONTH or DAY-TIME). Change either <code>qual1</code> or <code>qual2</code> to fit the respective class range or change the <code>sqlType</code> property from INTERVAL to another <code>sqlType</code>.</td>
</tr>
<tr>
<td>GS-24222</td>
<td>Startfield of DATETIME or INTERVAL qualifiers must come earlier in the time-list than its endfield. The <code>qual1</code> value should be greater than the <code>qual2</code> value, when the <code>sqlType</code> is either DATETIME or INTERVAL. Ensure <code>qual1</code> is greater than <code>qual2</code> or change the <code>sqlType</code> property from INTERVAL to another <code>sqlType</code>.</td>
</tr>
<tr>
<td>GS-24223</td>
<td>Query properties set without attached database schema. Some query properties (join, order, additional tables or where) are defined without a database schema attached. Clear the query properties or attach a database schema to document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-24224</td>
<td>Table <em>table</em> referenced by <em>property</em> is not present in the record.</td>
</tr>
<tr>
<td></td>
<td>A table is referenced in a query property (join or order), but is not</td>
</tr>
<tr>
<td></td>
<td>referenced in any record's field or additional tables.</td>
</tr>
<tr>
<td></td>
<td>• Remove the join or order that references this table.</td>
</tr>
<tr>
<td></td>
<td>• Add a field referencing a column from this table to the record.</td>
</tr>
<tr>
<td></td>
<td>• Add the table to the 'additional properties' property.</td>
</tr>
<tr>
<td></td>
<td>• In the join or order, change the table to another one that is present in</td>
</tr>
<tr>
<td></td>
<td>the record.</td>
</tr>
<tr>
<td>GS-24225</td>
<td>Invalid relation, it must have at least one field.</td>
</tr>
<tr>
<td></td>
<td>• Remove the relation.</td>
</tr>
<tr>
<td></td>
<td>• Add one or more source, destination field pair.</td>
</tr>
<tr>
<td>GS-24226</td>
<td>Value not compatible with <em>dataType</em>.</td>
</tr>
<tr>
<td></td>
<td>The value is not compatible with <em>dataType</em> set. This error occurs when</td>
</tr>
<tr>
<td></td>
<td><em>defaultValue</em> and <em>include</em> property value is incompatible with <em>dataType</em></td>
</tr>
<tr>
<td></td>
<td>property.</td>
</tr>
<tr>
<td></td>
<td>Change <em>defaultValue/include</em>, enter a value compatible to <em>dataType</em> format.</td>
</tr>
<tr>
<td>GS-24227</td>
<td>Invalid join between %1 and %2.</td>
</tr>
<tr>
<td></td>
<td>A join between the two tables is not correct (another one exists with a</td>
</tr>
<tr>
<td></td>
<td>different operator).</td>
</tr>
<tr>
<td></td>
<td>• Remove the join</td>
</tr>
<tr>
<td></td>
<td>• Change the operator</td>
</tr>
<tr>
<td></td>
<td>• Change the join table(s)</td>
</tr>
<tr>
<td>GS-24228</td>
<td>There is no join between table %1 and master table.</td>
</tr>
<tr>
<td></td>
<td>The specified table is joined to another one, but not to the master table,</td>
</tr>
<tr>
<td></td>
<td>creating a Cartesian product.</td>
</tr>
<tr>
<td></td>
<td>• Add the missing join for the %1 table</td>
</tr>
<tr>
<td></td>
<td>• Remove the table %1</td>
</tr>
<tr>
<td>GS-24229</td>
<td>Invalid initializer.</td>
</tr>
<tr>
<td></td>
<td>The initializer format is incorrect.</td>
</tr>
<tr>
<td></td>
<td>Change the initializer to respect the format.</td>
</tr>
<tr>
<td>GS-24230</td>
<td>Invalid source %1 for initializer %2.</td>
</tr>
<tr>
<td></td>
<td>The initializer source (left of &quot;.&quot;) is unknown.</td>
</tr>
<tr>
<td></td>
<td>Change the initializer to respect the format.</td>
</tr>
<tr>
<td>GS-24231</td>
<td>Initializer property %1 is missing.</td>
</tr>
<tr>
<td></td>
<td>The property used in the initializer does not exist.</td>
</tr>
<tr>
<td></td>
<td>Change the property name in the initializer.</td>
</tr>
<tr>
<td>GS-24232</td>
<td>Cannot resolve initializer value.</td>
</tr>
<tr>
<td></td>
<td>The initializer cannot be resolved, the database element is not found, or</td>
</tr>
<tr>
<td></td>
<td>the property is missing.</td>
</tr>
<tr>
<td></td>
<td>Change the initializer to point to a valid element.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-24235</td>
<td>File type not defined in Application Generator settings.</td>
</tr>
<tr>
<td></td>
<td>A file used in Application Generator to generate code ($generate) is of an</td>
</tr>
<tr>
<td></td>
<td>undefined file type in settings.agconf.</td>
</tr>
<tr>
<td></td>
<td>Add the item type definition to the settings.agconf.</td>
</tr>
<tr>
<td>GS-24236</td>
<td>Cannot resolve initializer without a valid database schema.</td>
</tr>
<tr>
<td></td>
<td>The property initializer uses the database schema but the file doesn’t have a</td>
</tr>
<tr>
<td></td>
<td>database schema.</td>
</tr>
<tr>
<td></td>
<td>• Set the database.</td>
</tr>
<tr>
<td></td>
<td>• Set the property value so that the initializer is not resolved.</td>
</tr>
<tr>
<td></td>
<td>• Remove the initializer.</td>
</tr>
<tr>
<td>GS-24237</td>
<td>Unique key field %1 is not present in the Record.</td>
</tr>
<tr>
<td></td>
<td>The unique key field value contains one field which is not in the record.</td>
</tr>
<tr>
<td></td>
<td>• Remove the field from the unique key.</td>
</tr>
<tr>
<td></td>
<td>• Add the field to the record.</td>
</tr>
<tr>
<td>GS-24239</td>
<td>Inactive table %1 in the database schema.</td>
</tr>
<tr>
<td></td>
<td>The database table active flag is set to false, the table is inactive and</td>
</tr>
<tr>
<td></td>
<td>cannot be used for code generation.</td>
</tr>
<tr>
<td></td>
<td>Make the table active or do not use it.</td>
</tr>
<tr>
<td>GS-24240</td>
<td>Duplicate uid '%1'.</td>
</tr>
<tr>
<td></td>
<td>Invalid file format. A unique identifier has been used more than once.</td>
</tr>
<tr>
<td></td>
<td>Contact Genero support.</td>
</tr>
</tbody>
</table>

**XML validation error messages**

A list of XML validation error messages. For messages that are not self-explanatory, additional information is provided.

**Table 185: XML Validation Error messages**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-10400</td>
<td>Invalid root element.</td>
</tr>
<tr>
<td></td>
<td>Edit the root to be &lt;ManagedForm&gt;.</td>
</tr>
<tr>
<td>GS-10401</td>
<td>Invalid child %1.</td>
</tr>
<tr>
<td></td>
<td>The element is not a valid child of the parent.</td>
</tr>
<tr>
<td></td>
<td>Edit the child to be an acceptable child of the parent.</td>
</tr>
<tr>
<td>GS-10402</td>
<td>%1 occurs more than once.</td>
</tr>
<tr>
<td></td>
<td>An element is defined more than once, eg: Topmenu, Toolbar can appear just</td>
</tr>
<tr>
<td></td>
<td>once.</td>
</tr>
<tr>
<td></td>
<td>Remove the duplicate occurrences to keep only a single valid definition.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-10403</td>
<td>Property %1 occurs more than once.</td>
</tr>
<tr>
<td></td>
<td>A property for an element is defined more than once.</td>
</tr>
<tr>
<td></td>
<td>Remove the duplicate occurrences to keep only a single valid definition.</td>
</tr>
<tr>
<td>GS-10404</td>
<td>Unexpected property %1 is ignored.</td>
</tr>
<tr>
<td></td>
<td>This warning is displayed when a property that does not belong to the element is defined. This is ignored.</td>
</tr>
<tr>
<td></td>
<td>Remove the property that does not belong.</td>
</tr>
<tr>
<td>GS-10405</td>
<td>Incomplete %1 definition, missing property(s) %2.</td>
</tr>
<tr>
<td></td>
<td>Some mandatory property or properties of an element have not been defined.</td>
</tr>
<tr>
<td></td>
<td>Contact your support center - provide the 4fd file, the Genero Studio version, and how this file was created.</td>
</tr>
<tr>
<td>GS-10407</td>
<td>Invalid recordField, corresponding formfield not found.</td>
</tr>
<tr>
<td></td>
<td>The recordField does not correspond to any formfield, i.e. no formfield is present whose fieldId property matches the fieldIdRef property of recordField.</td>
</tr>
<tr>
<td></td>
<td>• Check if recordField definition is invalid, and make it valid.</td>
</tr>
<tr>
<td></td>
<td>• Add a valid formField.</td>
</tr>
<tr>
<td></td>
<td>• Remove the recordField.</td>
</tr>
<tr>
<td>GS-10408</td>
<td>Incomplete Table definition for %1, missing record.</td>
</tr>
<tr>
<td></td>
<td>The table does not have a corresponding record defined.</td>
</tr>
<tr>
<td></td>
<td>• Check if any existing records are invalid and could be table's record.</td>
</tr>
<tr>
<td></td>
<td>• Add a valid record.</td>
</tr>
<tr>
<td></td>
<td>• Remove table.</td>
</tr>
<tr>
<td>GS-10409</td>
<td>Invalid record, corresponding table not found.</td>
</tr>
<tr>
<td></td>
<td>The record does not correspond to any table.</td>
</tr>
<tr>
<td></td>
<td>• Check if record definition is invalid and make it valid.</td>
</tr>
<tr>
<td></td>
<td>• Add a valid table.</td>
</tr>
<tr>
<td></td>
<td>• Remove the record if not needed.</td>
</tr>
<tr>
<td>GS-10410</td>
<td>Incomplete %1 definition, missing mandatory child.</td>
</tr>
<tr>
<td></td>
<td>The element's mandatory child has not been defined.</td>
</tr>
<tr>
<td></td>
<td>Add the missing child.</td>
</tr>
<tr>
<td>GS-10411</td>
<td>Invalid geometry for %1.</td>
</tr>
<tr>
<td></td>
<td>The geometry of element is invalid.</td>
</tr>
<tr>
<td></td>
<td>Edit the geometry properties $posX, posY, gridWidth, gridHeight$ such that the element lies within the boundaries of its parent.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-10412</td>
<td>Incomplete FormField definition for %1, missing recordField. The formField does not have a corresponding recordField defined. • Check if any existing recordFields are invalid and could be corresponding to formfield. • Add a valid RecordField. • Remove FormField.</td>
</tr>
<tr>
<td>GS-10413</td>
<td>Malformed XML. The xml structure is invalid. Check for missing tags.</td>
</tr>
</tbody>
</table>

**File Browser**

*File Browser* is a tool to navigate, open, delete or rename files, and to create new folders or files on a file system.

File Browser manages files based on the kind of information that they contain (MIME type file management). File Browser will launch the specific action / executable defined as the default for each file type, or you may choose between the actions available for that file type.

For example, *Code Editor* opens for a .4gl file, and *Form Designer* for a .4fd file.

The File Browser displays in a view named *Files*.

**Navigating files in File Browser**

The folders of the file structure in File Browser are displayed as a tree.

Open the File Browser with **Tools > File Browser** or **Window > Views > Files**.

Icons indicate the folders and file types. If the file is under *Source Code Management (SCM)* on page 624, the status is indicated on the icon.

The integrated Toolbar includes options for navigating the tree and refreshing the directory. Navigation history is maintained across sessions of Genero Studio and accessed by the drop down arrows of the previous and next buttons in the tool bar.

To show hidden files in the File Browser, check the *Show Hidden Files* box in **Tools > Preferences > File Browser**.

**Selecting files in File Browser**

Click the file name to select it, double-click to open it.

Files open in the module associated with them.

To select consecutive files or folders, click the first item, press and hold down SHIFT, and then click the last item. To select nonconsecutive files or folders, press and hold down CTRL, and then click each item.
Managing files in File Browser

Manage file actions such as open, cut, copy, paste, rename, and delete.

Use the mouse to drag and drop folders and files from one folder into another. Select a file listing, and right-click to display the actions available for that file, based on the mime type.

The **New file** option allows you to create a new document. See the **The File New menu** on page 98 main menu option for additional information about creating new files.

You can **associate file types** handled by Genero Studio applications with predefined actions, and **create user-defined actions**.

**Locate a file (starting at File Browser)**

From the File Browser, you can locate the file in the System File Browser.

Before you begin, the File Browser (**Files view**) is open. See **Navigating files in File Browser** on page 588.

You have a file visible in the **Files view** that you wish to locate in the System File Browser for your operating system.

1. Right-click on the file.
2. Select **Locate in System File Browser**.

**Debugging an application in Studio**

You have several tools available for debugging your applications in Genero Studio.

Debugging an application involves testing and controlling the execution of a program step by step, in order to find logical and runtime errors.

The **Graphical Debugger** on page 589 provided in Genero Studio has several tools and views for debugging applications.

The Graphical Debugger can be used to debug an application running locally or it can attach to one running remotely on a server or on a mobile device. Procedures and examples show you how to:

- **Debug a local or remote application**
- **Debug a Web services server application** on page 614
- **Attach the Graphical Debugger to a mobile process** on page 617

Topics:

- **Graphical Debugger** on page 589
- **Debug a Web services server application** on page 614
- **Debugging a mobile app** on page 615

**Graphical Debugger**

The **Graphical Debugger** provides a graphical interface to test and control the behavior of a Genero application.

You can navigate through functions, and create and manage breakpoints on functions and code lines. Choose and group together variables to watch and follow a number of variables easily. For testing purposes, you can alter variable values while the application is running.

- **Controlling program execution** on page 590
- **Debugger output** on page 595
Controlling program execution

Information on using the debugger.

- **Start the Debugger** on page 590
- **Stop the Debugger** on page 591
- **Debugger output** on page 595
- **Step through the program** on page 591
- **Breakpoints** on page 592
- **Watchpoints** on page 593

**Start the Debugger**
To launch the Debugger right-click on a program node and select **Debug**.

Graphical Debugger can be launched from within Project Manager:

- Right-click a program node and select **Debug** from the contextual menu; this launches the selected application in debug mode.
- Use the **Debug** Toolbar icon; this launches the default application.

If the Debug option is not enabled, check:

- Is the source file directory physically available (delete / rename) ?
- Are you using the correct configuration to access this project ? (local / remote) ?

Once the debugger is launched, the program begins execution. If you have set a breakpoint in a Genero file (4gl), execution stops at that breakpoint, and the source file containing the breakpoint is opened in the Document view.

**Start the Debugger on a running program**
You can debug a running program.

The procedure you follow to debug a running program will depend on whether the program is local or remote.

- **Debug a running local program** on page 590
- **Debug a running remote program** on page 591

**Debug a running local program**
Complete this procedure to attach the Graphical Debugger to a running local process.

Before you begin:

- You have a Genero program that is running locally.

1. Select **Debug >> Attach to Process...**
   The **Attach to process** dialog opens.
2. Select the **Attach to local process** radio button.
3. In the list of processes, select the process you wish to attach to.
   
   **Tip:** Each process is identified by the process ID, or PID. On Windows™ systems, you can use the Task Manager to map the process ID to the fglrun command that started the Genero program.
4. Click **Attach**.
   The debug session starts. The **Select the file** dialog opens, as the debugger needs to know where the source files are located.
5. Navigate to the directory that contains the source files for your application and select the appropriate source (.4gl) file, then click Open. The source file opens. You are now in a standard debug session.

With the debug session open, follow the procedures for using the Graphical Debugger.

**Debug a running remote program**

Complete this procedure to attach the Graphical Debugger to a running remote process.

Before you begin:

- You have a Genero program that is running remotely.
- If the source files are located on a remote machine, you must mount a network drive.

1. Select Debug >> Attach to Process...
The Attach to process dialog opens.
2. Select the Attach to remote process (SSH) radio button.
3. Complete the required fields.

   - **Host**: The remote host.
   - **Port**: The port number for communicating with the remote host.
   - **User**: The user name needed to connect to the remote host.
   - **Password**: The password needed to connect to the remote host.
   - **FGLDIR**: The FGLDIR directory for the version of Genero containing the fglrun used to launch the program.
   - **Process ID**: The process ID of the Genero program.

4. Click Attach. The debug session starts. The Select the file dialog opens, as the debugger needs to know where the source files are located.
5. Navigate to the directory that contains the source files for your application and select the appropriate source (.4gl) file, then click Open. The source file opens. You are now in a standard debug session.

With the debug session open, follow the procedures for using the Graphical Debugger.

**Stop the Debugger**

What happens when you stop a debug session depends on how the debug session was started.

Regardless of how the debug session was started, the Graphical Debugger will automatically exit when the program being debugged terminates.

To exit the Debugger, select Abort last task from the Debug Menu.

- If the program was started in debug mode, the program terminates.
- If the debugger was attached to a running process, the debugger terminates. The program continues to run.

**Step through the program**

The step command allows you to "step" through your program executing one line of source code at a time. When a function call appears within the line of code, that function is also stepped through.

Once launched, program execution will be halted at the first breakpoint encountered, waiting for your action. Debugger commands can then be selected from the Toolbar, or Debug menu options.

**Related reference**

- Debugger commands on page 603
  The Debugger supports a set of command-line commands.
- Debug context menu on page 598
Commands on the **Debug** menu execute the application in debug mode.

### Breakpoints

A breakpoint is a signal that tells the debugger to pause the program. A breakpoint displays as a red dot beside the corresponding code line.

Breakpoints are processed as private to each user. They are linked to a source code line, not to a line number; altering a source file does not affect the breakpoint’s relative position in functions. Breakpoints are stored when the Debugger is stopped, and set when the Debugger is started. Previously-set breakpoints are available each time the Debugger is run.

- [Set a breakpoint](#) on page 592
- [Conditional breakpoints](#) on page 593

### Related concepts

- [Breakpoints view](#) on page 602

This **Breakpoints** view displays information on all breakpoints set for the current project.

### Set a breakpoint

Set a breakpoint in a line of code if you want to examine that area of code more closely.

To add a breakpoint, click in the gutter at the corresponding code line, or right-click and select **Add/Delete breakpoint**. The breakpoint displays as a red dot in the gutter.

![Figure 293: Set Breakpoint](image)

To delete the breakpoint, click on the red dot, or right-click and select **Add/Delete breakpoint**.

To disable a breakpoint, right-click and select **Enable/Disable breakpoint**. A disabled breakpoint displays as a red dot with an empty center. You can enable the breakpoint later if required.

The **Add/Delete breakpoint** and **Enable/Disable breakpoint** options are disabled when the debugger is active.

### Initial breakpoint

Set initial breakpoints prior to launching the Debugger. Otherwise, the application executes normally within the Debugger framework and you must use the Debugger **Interrupt** option to monitor Debug views or employ Debugger functions. Setting a breakpoint at the first function call is usually a good starting point.

### Additional breakpoints

After the initial Breakpoint is set, you can either set additional breakpoints prior to starting the Debug session or start the session with the single breakpoint. Breakpoints on Function entry points automatically open the function source module during the debug session so you can examine the code more closely and even follow program execution line-by-line in the function. You may find it helpful to set function breakpoints before starting the session, but you can also set additional breakpoints as you use Debugger commands to navigate during the session.

**Tip:** You can use the Code Structure view to quickly identify and open source modules so that you can set breakpoints as desired. This is especially useful for functions in external modules.

### Related concepts

- [Debugger (fgldb) command prompt](#) on page 595
The Graphical Debugger also provides a command line interface. Once you have started a debug session, you can enter `fgldb` commands directly into the `fgldb Command` view.

**break** on page 603  
Sets a breakpoint at the specified line or function.

**Conditional breakpoints**  
You can use a Boolean expression to specify that a breakpoint is conditional.

In the **Condition** column of the **Breakpoints** tab, enter the condition or open the **Edit Expression** dialog. If a condition is defined, it is checked each time the line is to be executed. Execution stops at this breakpoint only if the condition is true.

Example:

```
i > 50
```

**Watchpoints**  
A watchpoint is a signal that tells the debugger to pause the program when the value of an expression changes.

- **Set a watchpoint** on page 593  
- **Conditional watchpoints** on page 594

**Set a watchpoint**  
Watchpoints can be set to stop program execution each time the value of an expression changes. In order to set the watchpoint, the program must be running in the Debugger.

1. **Set a breakpoint** in the function that has the variable expression in scope.  
2. **Start debugging** the program.  
3. When the program stops at the breakpoint, select the **Data view** on page 600 tab to display the variables.  
4. Right-click the variable and select **Add to watch** to add the variable to the Watchpoints view.
5. Continue running the program. The program will stop when the value of a watched variable changes. You can see the values in the Watchpoint view.

**Related concepts**

*Data view* on page 600
In the Data view you can examine or set the values of global, module, and function variables while your application is running.

*Watchpoints view* on page 601
Each time the value of a watched variable changes, the program will stop, and the variable values will be displayed in the Watchpoints view.

*Debugger (fgldb) command prompt* on page 595
The Graphical Debugger also provides a command line interface. Once you have started a debug session, you can enter `fgldb` commands directly into the `fgldb` Command view.

*watch* on page 613
Sets a watch point for an expression which stops execution of your program whenever the value of the expression changes.

**Conditional watchpoints**
You can specify that the watchpoint is conditional, using a Boolean expression to determine whether to stop execution of the program when the variable value has changed.

Each time the value of the watched variable changes (and the BOOLEAN expression, if used, is TRUE), the program will stop, and the variable values will be displayed Watchpoints view.

In the **Condition column** of the Watchpoints view, enter the condition or press the ... button to open the Edit Expression dialog.
Or, you can enter the watch Debugger instruction and condition in the Commands view.

```
watch expression boolean-expression
```

Example:

```
watch i if i > 3
```

**Debugger (fgldb) command prompt**

The Graphical Debugger also provides a command line interface. Once you have started a debug session, you can enter `fgldb` commands directly into the `fgldb` **Command** view.

You can **Start the Debugger** on page 590 in Genero Studio and then use the command window to enter `fgldb` commands, or you can start a debug session at the command line (outside of Genero Studio), with the `-d` option:

```
fglrun -d myprog
```

**Related concepts**

- **Command view** on page 599
  The Command view displays executed debugger commands and allows input of new debugger commands.

**Related reference**

- **Debugger commands** on page 603
  The Debugger supports a set of command-line commands.

**Debugger output**

The Debugger provides multiple views to assist you with program analysis during the Debug session.

- **Command view** on page 599
- **Data view** on page 600
- **Breakpoints view** on page 602
- **Watchpoints view** on page 601
- **Backtrace view** on page 603

**Examining data**

Monitor the values of program variables while the program is running.

**Variables**

The **Data view** on page 600 displays variable values.

- **Global variables**
  Show the variables defined with global scope as well as any Genero predefined variables used in the application, such as `INT_FLAG`.

- **Module variables**
  Show the variables defined with module scope.

- **Local variables**
  Show the variables defined with local scope for the currently executing function. Local variables are separated from argument for better visibility.

- **User variables**
  Allow you to group together variables from any of the other folders, for convenience in testing. To add a variable to the User variables set, or delete an existing User variable, right-click the variable listing to invoke the contextual menu.

User variables which are not in the scope of the current context are greyed.
Examining execution flow

Information on the call stack, stack frames, and navigating the stack.

The call stack and stack frames

Each time your program performs a function call, information about the call is saved in a block of data called a stack frame. Each frame contains the data associated with one call to one function.

The stack frames are allocated in a region of memory called the call stack. When your program is started, the stack has only one frame, that of the function MAIN. This is the initial frame, also known as the outermost frame. As the debugger executes your program, a new frame is made each time a function is called. When the function returns, the frame for that function call is eliminated.

The Debugger assigns numbers to all existing stack frames, starting with zero for the innermost frame, one for the frame that called it, and so on upward. These numbers do not really exist in your program; they are assigned by the Debugger to allow you to designate stack frames in commands.

Each time your program stops, the Debugger automatically selects the currently executing frame and describes it briefly. You can use the frame command to select a different frame from the current call stack.

Navigate the stack

Use the Up/Down options to track function calls. Up displays the calling function. The Up command advances toward the outermost frame each time it is processed. The Data view on page 600 display is based on the new context.

Down is the opposite of Up: it comes back to the inner frame down to the last executed line.

Related reference

Debugger commands on page 603
The Debugger supports a set of command-line commands.

Record/replay a macro

You can record the Debugger commands that you are executing, save them in a file, and to execute the commands again from that file.

Options on the Debug menu allow you to record, stop, and play a macro.

Record macro Start recording Debugger commands.
Stop record Stop command recording and store the commands in a file.
Play macro Execute the recorded commands from the file.

Profiler

The Profiler is a tool built in the runtime system that generates a report about where the program spends time, and which function calls which function. The Profiler can help to identify areas in the program that are slower than expected.

Use the Debug > Execute with Profiler menu option to start the Profiler. After the program finishes executing, the Profiler information is displayed in the Output tab.

The flat profile contains a list of the functions called while the programs was running.

Tip: Runtime system internal function names start with the rts_ prefix. For example, the rts_display() function implements the DISPLAY instruction.
Table 186: Flat profile columns

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Counts the number of calls for this function</td>
</tr>
<tr>
<td>%total</td>
<td>Percentage of time spent in this function. Includes time spent in subroutines called from this function.</td>
</tr>
<tr>
<td>%child</td>
<td>Percentage of time spent in functions called from this function.</td>
</tr>
<tr>
<td>%self</td>
<td>Percentage of time spent in this function excluding the time spent in subroutines called from this function.</td>
</tr>
<tr>
<td>name</td>
<td>Function name</td>
</tr>
</tbody>
</table>

Note: 100% represents the total processing time. The time spent waiting for user interaction is ignored.

The profiler is a tool built into the runtime system. For further information on the profiler and how it works, see the *Genero Business Development Language User Guide*.

Local vs. remote debug

You can configure Genero Studio to debug a local or remote application.

Check your active configuration before beginning a debug session.

- Use a **local** Genero configuration to work on your local application.
- Use a **remote** Genero configuration or your **local** Genero configuration to access and debug your remote application.

Use these steps to debug an application where the binaries are located on a remote server (production server) and the sources are located locally.

1. Set a remote configuration.
2. Confirm that all binaries are on the remote server.
3. Create an empty **application node** with the same name as the 42r binary located on the remote server.
4. Set all required environment variables on the **application node**.

   **FGLLDPATH**
   
   Set to the path of the 42m and 42r files located on the remote server.

   **FGLRESOURCEPATH**
   
   Set to the path of the 42f files (and other resource files).

   **FGLSOURCEPATH**
   
   Set to the path of the source files located locally.

Reference

Reference information for the Debugger.

- Debug context menu on page 598
- Views on page 599
- Debugger commands on page 603
**Debug context menu**
Commands on the **Debug** menu execute the application in debug mode.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug</td>
<td>Begins program execution, debugging up to the first breakpoint or interruption.</td>
</tr>
<tr>
<td>Execute with Profiler</td>
<td>Execute selected application with Profiler. See Profiler on page 596</td>
</tr>
<tr>
<td>Next</td>
<td>Executes the current line, and stops at the next source line in the <em>current</em> function. If this source code line is a call to another function, you can Step in or Step out.</td>
</tr>
<tr>
<td>Step in</td>
<td>Step into the current function, executing the next source line inside the <em>called</em> function.</td>
</tr>
<tr>
<td>Step out</td>
<td>Step out of the called function before it ends, returning to the next source line following the CALL statement.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Some complex instructions such as CONSTRUCT or FOR loops need two steps before going to the next instruction: one step to prepare the statement, one step to execute it.</td>
</tr>
<tr>
<td>Continue</td>
<td>Resumes program execution, until another breakpoint is reached or the program terminates.</td>
</tr>
<tr>
<td>Interrupt</td>
<td>Interrupts the program execution, displaying the current line being executed and updates the Debugger views such as the Data view.</td>
</tr>
<tr>
<td></td>
<td>Stops the execution of the program being debugged, returns control to the debugger, displays the current line being executed and updates the debugger views such as the Data view. This allows you to interrupt a program that is in an endless loop, or that is displaying a form, for example.</td>
</tr>
<tr>
<td>Up</td>
<td>Displays the calling function. The up command advances toward the outermost frame each time it is processed. The Data view on page 600 is based on the new context.</td>
</tr>
<tr>
<td>Down</td>
<td>The opposite of Up: it comes back to the inner frame down to the last executed line.</td>
</tr>
<tr>
<td>Record Macro</td>
<td>Start recording Debugger commands.</td>
</tr>
<tr>
<td>Stop Record</td>
<td>Stop command recording and store the commands in a file.</td>
</tr>
<tr>
<td>Play Macro</td>
<td>Execute the recorded commands from the file.</td>
</tr>
<tr>
<td>Add/Delete Breakpoint</td>
<td>Sets or removes a breakpoint for the current line.</td>
</tr>
<tr>
<td>Enable/Disable Breakpoint</td>
<td>Re-actuates or disables the breakpoint for the current line.</td>
</tr>
<tr>
<td>Abort Last Task</td>
<td>Terminates the debugger session.</td>
</tr>
</tbody>
</table>

**Related concepts**
*Menus and Toolbars* on page 96
Menus and Toolbars are constructed dynamically depending on the context and the currently active module.

**Views**
Information about Debugger views.

- **Command view** on page 599
- **Data view** on page 600
- **Watchpoints view** on page 601
- **Breakpoints view** on page 602
- **Backtrace view** on page 603

**Command view**
The Command view displays executed debugger commands and allows input of new debugger commands.

![Command view](image)

**Figure 295: Command view**
The Command view displays the executed debugger commands. The output of the command, if any, appears immediately below the command.

The Command view allows for input of debugger (fgldb) commands. These commands can be entered by hand, or previously entered commands can be selected from the combobox. Commands can only be entered when the debugger is interrupted. An hourglass in the field indicates that the debugger is active and that commands cannot be entered.
**Data view**

In the Data view you can examine or set the values of global, module, and function variables while your application is running.

![Figure 296: Data View](image)

In the Data view, variables are organized into groups and displayed in a tree. Expand a group to display its variables. For a variable, you can see the variable name, value, and type (record, column, data type). Double-click the value to edit (change) the value for testing purposes.

For an array value, only six rows are initially shown. If more rows are available than are displayed, a message at the end of the array ("Show more …") indicates more items are available. Click on this message to display additional rows.

The integrated toolbar has two sections. The icons on the left-hand side of the toolbar allow you to:

- Display a variable value. When you click this icon, the Edit Value dialog opens, allowing you to modify the value of the variable.
- Add, copy and delete variables in the User variables group.
- Add a variable to the watch list.
- Fetch all variables.

The icons on the right-hand side of the toolbar allow you to show or hide variables from the different groups. From left to right, the icons control:

- Show only modified variables. When this icon is selected, only the modified variables display. When you change the value of a variable, only that variable will be seen as modified, as you can only edit one variable at a time. However, if you perform steps in the execution of the program, or if you continue its execution and the program interrupts, all variables modified since the last time the program was interrupted are displayed.
- Show local variables. This icon is a toggle switch; click it to show or hide the contents of the Local variables group.
- Show module variables. This icon is a toggle switch; click it to show or hide the contents of the Module variables group. Modules from all variables are shown; the current module (if any) is indicated by a green arrow.
- Show global variables. This icon is a toggle switch; click it to show or hide the contents of the Global variables group.

Right-click a variable to display a context menu with some of the same commands.
Watchpoints view
Each time the value of a watched variable changes, the program will stop, and the variable values will be displayed in the Watchpoints view.

The integrated Toolbar includes options to manage watchpoints.

- **Delete watchpoint**
  - Delete the selected watchpoint.

- **Edit watchpoint condition**
  - Edit the condition associated to the watchpoint.

- **Delete watchpoint condition**
  - Removes the condition associated to the watchpoint.

- **Delete all watchpoints**
  - Removes all watchpoints.

- **Disable all watchpoints**
  - Disables all watchpoints.
**Breakpoints view**

This **Breakpoints** view displays information on all breakpoints set for the current project.

Each breakpoint has its own row of information. You can manage breakpoints in the Breakpoints view before starting the debugger. During a debugging session, only those breakpoints that apply to the current project and session are shown.

![Breakpoints view](image)

### Figure 299: Breakpoints view

<table>
<thead>
<tr>
<th>Id</th>
<th>Enable</th>
<th>File</th>
<th>Line</th>
<th>Function</th>
<th>Condition</th>
<th>Hit Count</th>
<th>Skip Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
<td>orders.4gl</td>
<td>54</td>
<td>main</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>orders.4gl</td>
<td>86</td>
<td>setup_actions</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>✓</td>
<td>orders.4gl</td>
<td>134</td>
<td>order_new</td>
<td>store_num = “1”</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

- **Id**: Number assigned to the breakpoint by the Debugger. Note that the number is assigned after the debugger starts. If you examine the **Breakpoints** view outside of a debugging session, the breakpoints do not have a number assigned.

- **File, Line, Function**: Breakpoint's location in the source file.

- **Enable**: Select to enable the breakpoint. Clear to disable the breakpoint.

- **Condition**: Specify conditions for the breakpoint. Create or edit a condition by clicking in the row’s **Condition** column. If a condition is defined, it is checked each time the line is to be executed. Execution stops at this breakpoint only if the condition is true.

- **Hit Count**: Total number of times a breakpoint has been hit

- **Skip Hits**: Number of times the breakpoint should be ignored

To sort the breakpoints, click on a column header.

The integrated Toolbar includes options to manage breakpoints. These options are also available by right-clicking on a line in the Breakpoints view.

- **Go to source code**: Open the file containing the breakpoint. Double-clicking a line will also open the file containing the breakpoint.

- **Delete breakpoint**: Delete the selected breakpoint.

- **Edit breakpoint condition**: Edit the condition associated to the breakpoint.

- **Delete breakpoint condition**: Remove the condition associated to the breakpoint.

- **Delete all breakpoints**: Remove all breakpoints.

- **Disable all breakpoints**: Disable all breakpoints.
Related concepts
Breakpoints on page 592
A breakpoint is a signal that tells the debugger to pause the program. A breakpoint displays as a red dot beside the corresponding code line.

Backtrace view
In the Backtrace view you can view or trace the functions called while running the program.

![Backtrace view](image)

**Figure 300: Backtrace view**

The first line displayed is the current line. Then the calling function is displayed, and so on, up to the main function. The Backtrace view is display-only. Lines are added in the output while the program is running.

An arrow points to the currently executing function.

The integrated Toolbar includes options to navigate a backtrace.

- **Go to selected frame**: Make the selected frame the current frame. The Data view will be updated with the variables available in the selected frame.
- **Go to source code**: Display the source code corresponding to the selected frame.

Debugger commands
The Debugger supports a set of command-line commands.

**break**
Sets a breakpoint at the specified line or function.

**Syntax**

```
break [ line | module:line | function ]
```

1. *line* is a source code line in the current module.
2. *module:line* is a source code line in a specific module.
3. *function* is a function name.

**Usage**

The `break` command sets a breakpoint at a given position in the program.

When the program is running, the debugger stops automatically at breakpoints defined by this command.

**Example**

```
(fgldb) break mymodule:5
Breakpoint 2 at 0x00000000: file mymodule.4gl, line 5.
```
Related concepts
Set a breakpoint on page 592
Set a breakpoint in a line of code if you want to examine that area of code more closely.

backtrace
Prints a summary of how your program reached the current state.

Syntax
```yaml
backtrace
bt
```

Usage
The backtrace command prints a summary of your program's entire stack, one line per frame. Each line in the output shows the frame number and function name.

Example
```yaml
(fglldb) bt
  #1 addcount() at mymodule.4gl:6
  #2 main() at mymodule.4gl:2
(fglldb)
```

clear
Clear breakpoint at the specified line or function.

Syntax
```yaml
clear [function|line]
```
1. `function` - deletes any breakpoints set at entry to the specified function.
2. `line` - deletes any breakpoints set at or within the code of the line specified by number.

Usage
With the clear command you can delete specific breakpoints in your program. Use the clear command with no arguments to delete any breakpoints at the next instruction to be executed in the selected stack frame.

See the delete command to delete individual breakpoints by specifying their breakpoint numbers.

Example
```yaml
(fglldb) clear mymodule:5
Deleted breakpoint 2
(fglldb)
```

continue
Continues the execution of the program after a breakpoint.

Syntax
```yaml
continue [ignore-count]
```
1. `ignore-count` defines the number of times to ignore a breakpoint at this location.
Usage

The `continue` command continues the execution of the program until the program completes normally, another breakpoint is reached, or a signal is received.

Example

```
(fgldb) continue
   <.program output..>
Program exited normally.
```

`define`

Specifies a user-defined sequence of commands.

Syntax

```
define command-name \ command1 command2 ... \ end
```

1. `command-name` is the name assigned to the command sequence.
2. `command` is a valid debugger command.
3. `end` indicates the end of the command sequence.

Usage

The `define` command allows you to create a user-defined command by assigning a command name to a sequence of debugger commands that you specify. You may then execute the command that you defined by entering the command name at the debugger prompt.

User commands may accept up to ten arguments separated by whitespace.

Example

```
(fgldb) define myinfo
   > info breakpoints
   > info program
   > end
(fgldb)
```

`delete`

Removes breakpoints specified in a debugger session.

Syntax

```
delete breakpoint
```

1. `breakpoint` is the number assigned to the breakpoint by the debugger.

Usage

The `delete` command allows you to remove breakpoints when they are no longer needed in your debugger session. If you prefer you may disable the breakpoint instead. See the `disable` command.

Example

```
(fgldb) delete 1
(fgldb) run
```
Program exited normally.
(fgldb)

disable
Disables the specified breakpoint.

Syntax

disable breakpoint

1. breakpoint is the number assigned to the breakpoint by the debugger.

Usage

The disable command instructs the debugger to ignore the specified breakpoint when running the program. Use the enable command to reactivate the breakpoint for the current debugger session.

Example

(fgldb) disable 1
(fgldb) run
Program exited normally.
(fgldb)

display
Displays the specified expression's value each time the program stops.

Syntax

display expression

1. expression is your program's expression that you wish to examine.

Usage

The display command allows you to add an expression to an automatic display list. The values of the expressions in the list are printed each time your program stops. Each expression in the list is assigned a number to identify it.

This is useful in tracking how the values of expressions change during the program's execution.

Example

(fgldb) display a
1: a = 6
(fgldb) display i
2: i = 1
(fgldb) step
2: i = 1
1: a = 6
16 for i = 1 to 10
(fgldb) step
2: i = 2
1: a = 6
17 let a = a+1
(fgldb)
**down**
Selects and prints the function called by the current function, or the function specified by the frame number in the call stack.

**Syntax**

```
down [wn] [num]
```

1. `num` is the number of frames to move down the stack. The default is 1.

**Usage**

This command moves down the call stack, to the specified frame, and prints the function identified with that frame. To print the function called by the current function, use the `down` command without an argument.

```
(fgldb) down
#0 query_cust() at custquery.4gl:22
22      CALL cleanup()
(fgldb)
```

**enable**
Enables breakpoints that have previously been disabled.

**Syntax**

```
enable [breakpoint]
```

1. `breakpoint` is the number assigned to the breakpoint by the debugger.

**Usage**

The `enable` command allows you to reactivate a breakpoint in the current debugger session. The breakpoint must have been disabled using the `disable` command.

**Example**

```
(fgldb) disable 1
(fgldb) run
Program exited normally.
(fgldb) enable 1
(fgldb) run
Breakpoint 1, at mymodule.4gl:5
```

**file**
Specifies the name of the program being debugged.

**Syntax**

```
file filename
```

1. `filename` is the name of the program being debugged.

**Usage**

The `file` command can be used to change to a different file during a debugging session.
**finish**
Instructs the program to continue running until just after the function in the selected stack frame returns, and then stop.

**Syntax**
```plaintext
clean
```

**Usage**
The `finish` command instructs the program to continue running until just after the function in the selected stack frame returns, and then stop. The returned value, if any, is printed.

**Example**
```plaintext
(fgldb) finish
```

**help**
Provides information about debugger commands.

**Syntax**
```plaintext
help [command]
```

1. `command` is the name of the debugger command for which you wish information.

**Usage**
The `help` command displays a short explanation of a specified command.
Enter the `help` command with no arguments to display a list of debugger commands.

**Example**
```plaintext
(fgldb) help delete
```

**info**
Describes the current state of the program.

**Syntax**
```plaintext
info [breakpoints | sources | program | stack | files | line (linespec | module:line | function)]
```

1. `breakpoints` lists the breakpoints that you have set.
2. `sources` prints the names of all the source files in your program.
3. `program` displays the status of your program.
4. `stack` summarizes how your program got where it is.
5. `files` lists the names of the executable file and core dump files currently in use, and the files from which symbols were loaded.
6. `line linespec` maps the specified source code line to program addresses.
7. `line module:line` maps the specified source code line to program addresses.
8. `line function` prints the program addresses for the first line of the function named `function`. 
Usage
The info command describes the state of your program.

The command info line linespec prints the starting and ending addresses of the compiled code for the source line specified. See the list command for all the ways that you can specify the source code line.

Example
(fgldb) info sources
Source files for which symbols have been read in:

mymodule.4gl, fglwinexec.4gl, fglutil.4gl, fgldialog.4gl, fgldummy4js.4gl
(fgldb)

list
Prints source code lines of the program being executed.

Syntax
l[list]

Usage
The list command prints source code lines of your program, including the current line.

Example
(fgldb) run
Breakpoint 1, at mymodule.4gl:5
5 call addlist()
(fgldb) list
5 call addlist()
6 call addname()
.
14 end function
(fgldb)

next
Continues running the program by executing the next source line in the current stack frame, and then stops.

Syntax
n[ext]

Usage
The next command allows you to execute your program one line of source code at a time. The next command is similar to step, but function calls that appear within the line of code are executed without stopping. When the next line of code at the original stack level that was executing when you gave the next command is reached, execution stops.

After reaching a breakpoint, the next command can be used to examine a troublesome section of code more closely.

Example
(fgldb) next
### output
Prints only the value of the specified expression, suppressing any other output.

#### Syntax

```
output expression
```

1. `expression` is your program's expression that you wish to examine.

#### Usage

The `output` command prints the current value of the expression and nothing else, no newline character, no "expr=", etc.

The usual output from the debugger is suppressed, allowing you to print only the value.

#### Example

```
(fgldb) output b
```

### print
Displays the current value of the specified expression.

#### Syntax

```
print expression
```

1. `expression` is your program's expression that you wish to examine.

#### Usage

The `print` command allows you to examine the data in your program.

It evaluates and prints the value of the specified expression from your program, in a format appropriate to its data type.

#### Example

```
(fgldb) print b
$1=5
(fgldb)
```

### quit
Terminates the debugger session.

#### Syntax

```
quit
```

#### Usage

The `quit` command allows you to exit the debugger.
Example

(fgldb) quit
<system prompt>

run
 Starts the program executing until a breakpoint is reached or the program terminates normally.

Syntax

run [arg1 arg2 ...]

1. *arg* is an argument to be passed to your program.

Example

(fgldb) run
Breakpoint 1, at mymodule.4gl:3
3      call addcount()
(fgldb)

set
 Allows you to set the value of specific variables for the duration of the debugger session.

Syntax

set {environment varname=value | verbose {on | off} | annotate}

1. *varname* is the environment variable to be set to *value*.

Usage

The *set* command changes the values of environment variables.

When setting an *environment* variable, *value* may be any string. If the *value* parameter is omitted, the variable is set to a null value. The variable is set for your program, not for the debugger itself.

When *verbose* is set to *on*, the debugger will display additional messages about its operations, allowing you to observe that it is still working during lengthy internal operations.

Example

set verbose on

Important: On UNIX™ systems, if your SHELL variable names a shell that runs an initialization file, any variables you set in that file affect your program. You may wish to move setting of environment variables to files that are only run when you sign on, such as .login or .profile.

source
Executes a file of debugger commands.

Syntax

source commandfile

1. *commandfile* is the name of the file containing the debugger commands.
Usage
The source command allows you to execute a command file of lines that are debugger commands. The lines in the file are executed sequentially.

The commands are not printed as they are executed, and any messages are not displayed. Commands are executed without asking for confirmation.

An error in any command terminates execution of the command file.

Example
(fgldb) source mycommands

signal
Sends an INTERRUPT or QUIT signal to the program.

Syntax

  signal [SIGINT]

  signal [SIGQUIT]

Usage
The signal SIGINT command resumes execution of your program where it has stopped but immediately sends an INTERRUPT signal. The source line that was current when the signal was received is displayed.

Use signal SIGQUIT to send a QUIT signal.

Example
(fgldb) signal SIGINT
Program exited normally.

  for i = 1 to 10
(fgldb)

step
Continues running the program by executing the next line of source code, and then stops.

Syntax

  step [count]

1. count defines the number of lines to execute before stopping.

Usage
The step command allows you to "step" through your program executing one line of source code at a time. When a function call appears within the line of code, that function is also stepped through.

A common technique is to set a breakpoint prior to the section or function that is causing problems, run the program until it reaches the breakpoint, and then step through it line by line.

Example
(fgldb) step
4 call addlist(a)
(fgldb)

**Important:** The *step* command cannot be used to step through a function that was compiled without debugging information. Execution continues until it reaches a function that does have debugging information.

**until**

**Syntax**

```
until \ line \ module:line \ function \
```

1. *line* is a source code line in the current module.
2. *module:line* is a source code line in a specific module.
3. *function* is a function name.

**Usage**

The `until` command continues running your program until either the specified location is reached, or the current stack frame returns. This can be used to avoid stepping through a loop more than once.

**Example**

```
(fgldb) until addcount()
```

**up**

Selects and prints the function that called this one.

**Syntax**

```
up \ num\n```

1. *num* is the number of lines to move up the stack. The default is 1.

**Usage**

The `up` command advances toward the outermost frame, to frames that have existed longer.

**watch**

Sets a watch point for an expression which stops execution of your program whenever the value of the expression changes.

**Syntax**

```
watch expression \[boolean-expression\]
```

1. *expression* is the expression to watch.
2. *boolean-expression* is an optional boolean expression.

**Usage**

The watch point stops the program execution when the value of the expression changes.

If *boolean-expression* is provided, the watch point stops the execution of the program if the expression value has changed and the *boolean-expression* evaluates to TRUE.
Example

```
(fgldb) watch i if i >= 3
```

**where**

Alias for backtrace command.

**Syntax**

```
where
```

**Usage**

The `where` command prints a summary of your program's entire stack, one line per frame. Each line in the output shows the frame number and function name. It is an alias of the `backtrace` command.

Example

```
(fgldb) where
#1 addcount() at mymodule.4gl:6
#2 main() at mymodule.4gl:2
(fgldb)
```

**Related concepts**

backtrace on page 604

Prints a summary of how your program reached the current state.

**Unsupported commands**

Some debugger commands are not supported in Genero Studio.

Debugger commands not supported in Genero Studio are:

- define
- source
- tbreak
- tty

**Debug a Web services server application**

Complete this procedure to debug a Web services server application

Before you begin:

- You have a Genero Studio project that includes a Web services server application and (optionally) a Web services client application.
- Your Genero Studio configuration for Web applications is correctly configured, and references the correct parent application (defaultgst-debug) and parent service (ws.defaultgst-debug). These are defined for you in the default configuration for Web applications; you shouldn't need to define these yourself.

This procedure assumes you wish to debug the server side of a Web services application pair. For this procedure, you must run the client and the server application using the Web configuration.

**Tip:** To debug the client side of a Web services application, simply ensure that the Web services server application is running, then start the client application in debug mode.

1. Select the Web configuration for Genero Studio.
2. Ensure the Web service property is selected for the Web services server application.
3. Add a breakpoint to the Web services server application.
4. With the server application selected in the Projects view, select Debug > Debug to run the server application in debug mode.
   A page opens in a Web browser using the Web service URL. This is how Genero Studio starts the Web service.

5. Start the client application.
   You can either:
   • Run an external client program.
   • Start the client application from the same project as the Web services server. Right-click on the client application and choose Execute. A dialog appears, informing you that applications are running and asking you whether you wish to close the applications. Click No. A new Web page opens for the client application.

6. Use the client application. The Web services server application interrupts execution when the breakpoint is hit.

7. Use Genero Studio to debug the application.

---

### Debug a TUI client

At this time, the "debug" action is not available. To debug a program run in text mode, use the "attach to process" feature.

See [Debug a running local program](#) on page 590.

---

### Debugging a mobile app

You have several tools available for debugging your Genero Mobile application.

To use the debugging tools, you will either have:
- An app running in developer mode, with the Debug service enabled.
- A debug version of a deployed app.

#### Debug version of a deployed app

Debugging methods and tools are available for the debug versions of standalone apps.

- [Create a debug version of a deployed app](#) on page 615
- [Run the debug version of a deployed app (Android)](#) on page 616
- [Run a debug version of a deployed app (iOS)](#) on page 616

### Create a debug version of a deployed app

By default, the mobile app is built in release mode, which cannot be debugged. To build an app in debug mode, you must check the Mobile Debug Package environment set, which sets the DEBUG_PACKAGE environment variable to 1.

Before you begin, your app project must be open in Genero Studio.

1. Select Tools > Genero Configurations.
2. Select a Configuration Name.
   The current configuration is selected by default.
   Within this environment set, the DEBUG_PACKAGE environment variable is set to 1 (TRUE) to build a debug version of the package.
4. Package the app. See [Package a mobile app](#) on page 1175.
   The package for the debug version of the app is created and ready to be deployed.

**Note:** If you install the debug version of the app on an iOS device, you must then enable the debug port in the app settings. For further information, see the [Genero Business Development Language User Guide](#).
Run the debug version of a deployed app (Android™)

After deploying the app to your device, you must start the app in order to use the debug tools.

Before you begin:

• You must deploy the debug version of the app to the device.
• You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

1. On the device, launch the debug version of the app.
2. If the debug service is running on the device, stop it.
   
   If the debug service is running, the red Genero Mobile debug service icon appears in the status bar.
   
   a) Drag down from the top of the screen to display a menu.
   
   b) Below the Genero Mobile debug service entry, tap **Cancel**.
   
   The red Genero Mobile debug service icon no longer appears in the status bar.
3. Set **Debug service** to **ON**.
   
   The red Genero Mobile debug service icon appears in the status bar.
4. Tap **Browse bundled apps**.
5. Tap **InternalGeneroApps**.
6. In the list, find the two entries for your app.
   
   The two entries are file names: *programname.42r* and *packagenodename.xcf*.
   
   **Note:** If the package node name and the program name are identical, only one entry (*programname.xcf*) displays.
7. Tap on either file name to start the app.
   
   The debugger starts and you can debug your application.

Related tasks

*Create a debug version of a deployed app* on page 615

By default, the mobile app is built in release mode, which cannot be debugged. To build an app in debug mode, you must check the **Mobile Debug Package** environment set, which sets the **DEBUG_PACKAGE** environment variable to 1.

Run a debug version of a deployed app (iOS)

After deploying the debug version of the app to your device, you must start the app in order to use the debug tools.

Before you begin:

• You must deploy the debug version of the app to the device.
• You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

With Genero Mobile for iOS, the debug version of a deployed app enables the debug preferences by enabling port 6400 for debugging.

1. On the device, launch the debug version of the app.
2. In Genero Studio, select **Debug > Attach to Mobile Process**.
   
   The debugger starts and you can debug your application.

If you have trouble connecting, these are things you can check in the application’s settings:

• The debug port check box must be selected. You should not need to enable the debug port as it is checked by default.
• The debug port in the application’s settings must be set to 6400, otherwise Genero Studio will not be able to connect to it.

Related tasks

*Create a debug version of a deployed app* on page 615
By default, the mobile app is built in release mode, which cannot be debugged. To build an app in debug mode, you must check the Mobile Debug Package environment set, which sets the DEBUG_PACKAGE environment variable to 1.

### Debug tools for apps in developer mode

Some debug tools are only available when running your app in developer mode.

These debug tools are only available for apps running in developer mode.

- Preview a form on a mobile device on page 617
- Run an app with the graphical debugger on page 617
- Run an app with the Profiler on page 618

These debug tools can be used for apps running in developer mode.

- View the AUI tree (Android) on page 619
- View program logs (Android) on page 621
- View device logs (Android) on page 622

### Preview a form on a mobile device

When working in developer mode, you can preview your form on a mobile device.

**Before you begin:**

Select the appropriate Genero Studio configuration for displaying to your mobile device or emulator, and have your app project open in Genero Studio.

1. Open a form specification file (4fd, 4fdm, or per).
2. Select Build > Preview.

The form displays on your device.

### Run an app with the graphical debugger

When working in developer mode, you can use the graphical debugger to step through your code as your app executes on your mobile device.

**Before you begin:**

Select the appropriate Genero Studio configuration for displaying to your mobile device or emulator, and have your app project open in Genero Studio.

1. Open an app source (4gl) file.
2. In the source (4gl) file, use Debug > Add/Delete Breakpoint to set your breakpoints.
   The app starts on your device.
4. Use the graphical debugger icons to step through your code.
   For more information on using the graphical debugger, see the Genero Studio User Guide.

### Attach the Graphical Debugger to a mobile process

Complete this procedure to attach the Graphical Debugger to a running mobile process.

**Before you begin:**

- Genero Studio is set to use a mobile configuration.
- The mobile application was packaged and deployed with the Mobile Debug Package environment set selected.
1. Run the debug version of your deployed app.
   - For Android, see Run the debug version of a deployed app (Android) on page 616.
   - For iOS, see Run a debug version of a deployed app (iOS) on page 616
2. Select **Debug > Attach to Mobile Process**.
   This option uses your current configuration to connect to the mobile device.
3. Navigate to the directory that contains the source files for your application and select the appropriate source (.4gl) file, then click **Open**.
   The source file opens. You are now in a standard debug session.

With the debug session open, follow the procedures for using the Graphical Debugger.

**Note:** For details of attaching to and debugging a mobile app on a physical mobile device or emulator using the command line tool, **fgldb**, see the **Debugging on a mobile device** topic in *Genero Business Development Language User Guide*.

**Related concepts**
**Debugging a mobile app** on page 615
You have several tools available for debugging your Genero Mobile application.

**Run an app with the Profiler**
The Profiler is a tool built in the runtime system that generates a report about where the program spends time, and which function calls which function. The Profiler can help to identify areas in the program that are slower than expected.

**Before you begin:**
Select the appropriate Genero Studio configuration for displaying to your mobile device or emulator, and have your app project open in Genero Studio.

For more information on using the Profiler, see the *Genero Studio User Guide*.

**Note:** When in developer mode, the DVM is running on the desktop. The Profiler is therefore providing a profile of the app as it runs on the desktop. It may not be a true representation of how it will perform when deployed to the device.

1. Set the desired app as the default app.
2. Select **Debug > Execute with Profiler**.
   The app starts on your device.
3. Use the app.
4. End the app.
   The Profiler report appears in the Output panel.

**Viewing the AUI tree**

You can view the AUI tree in a browser. The AUI tree is the tree of the program's objects and user interface elements, and can be used to see what is present in the AUI tree for the program and the values of each element's properties.

The app must be running in developer mode or be the debug version of a deployed app.

- View the AUI tree (Android) on page 619
- View the AUI tree (iOS) on page 620
- View the AUI tree (Development Client) on page 621

**View the AUI tree (Android™)**

You can view the AUI tree for your Android™ mobile app in a browser.

Before you begin:

- Your app must be able to run in development mode or be the deployed debug version of the app.
- You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

1. If you are going to run your app in developer mode, complete these steps.
   a) From Genero Studio, launch your application.
   b) Select **Tools > Android Tools > Show AUI Tree**.
      The AUI Tree displays in a browser.
   c) Click on a node in the tree to highlight the corresponding item on the device or emulator.

2. To view your app in a browser while running the deployed debug version of your app using Genero Studio, complete these steps.
   a) To run the app from your device, launch your app in debug mode. See Run the debug version of a deployed app (Android) on page 616.
      The application launches and the initial form displays.
   b) If you are using Genero Studio, select **Tools > Android Tools > Show AUI Tree**.
      The AUI Tree displays in a browser.
   c) Click on a node in the tree to highlight the corresponding item on the device or emulator.

3. To view your app in a browser while running the deployed debug version of your app, complete these steps.
   a) To run the app from your device, launch your app in debug mode. See Run the debug version of a deployed app (Android) on page 616.
      The application launches and the initial form displays.
   b) Swipe down on the icon bar and locate the entry for the Debug service. Make a note of the URI address provided.
      The URI takes the form of `http://<device_ip_address>:<port_number>`.
   c) Open a browser and enter in the URI from the previous step.
d) Click **AUI tree**.
The AUI Tree displays in a browser.
e) Click on a node in the tree to highlight the corresponding item on the device or emulator.

**Important:** The AUI tree does not automatically update in your browser as you move around the app. As you navigate the app on your device, you must reload your browser to view the changes in the AUI tree.

**View the AUI tree (iOS)**
You can view the AUI tree for your iOS mobile app in a browser.

Before you begin:
- Your app must be able to run in development mode or be the deployed debug version of the app.
- You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

1. If you are going to run your app in developer mode, complete these steps.
   a) From Genero Studio, launch your application.
   b) Select **Tools > iOS Tools > Show AUI Tree**.
      The AUI Tree displays in a browser.
   c) Click on a node in the tree to highlight the corresponding item on the device or simulator.

2. If you are going to run the deployed debug version of your app, complete these steps.
   a) On the iOS device, open Genero Mobile.
   b) In the **INFORMATION** section, note the **HTTP URL** provided.
      For example, **HTTP URL=http://192.168.0.160:6400**
   c) Launch your application in debug mode. See **Run a debug version of a deployed app (iOS)** on page 616.
   d) Swipe down to view details about the icons in the status bar.
      The description for the Genero Mobile debug service displays the URL you must use to access debug details. It takes the form **http://device-ip-address:port-number**.
   e) Enter the URL in a browser.
   f) Click **AUI tree**.
      The AUI Tree displays in a browser.
   g) Click on a node in the tree to highlight the corresponding item on the device or simulator.

3. If you are going to run the deployed debug version of your app, and your machine (running Genero Mobile) and device (running the app) are on the same wireless network, you can use Genero Studio to view the AUI tree.
   a) On the iOS device, open Genero Mobile.
   b) Select **Tools > Genero Configurations**.
   c) Select your iOS configuration from the list.
   d) Click the Edit icon next to the **Use Display Client** field.
      The Display Client management window opens.
   e) In the **Host** field, enter the ip address of your device. Click **OK** until all configuration management windows close.
   f) Launch your application in debug mode. See **Run a debug version of a deployed app (iOS)** on page 616.
   g) From Genero Studio, select **Tools > iOS Tools > Show AUI Tree**.
      The AUI Tree displays in a browser.
   h) Click on a node in the tree to highlight the corresponding item on the device or simulator.

**Important:** The AUI tree does not automatically update in your browser as you move around the app. As you navigate the app on your device, you must reload your browser to view the changes in the AUI tree.
**View the AUI tree (Development Client)**

While viewing your app in the Genero Development Client, you can view the AUI tree for your mobile app in a browser.

Before you begin, you should have verified your configuration within Genero Studio to display to the Genero Development Client.

The Genero Development Client allows the developer to view the AUI tree on the default port 6400.

1. On your iOS device, launch the Genero Development Client.
2. Find the **HTTP VIEWING URL**. Make a note of the URL.
   
   **Note:** The **HTTP VIEWING URL** displays in the Development window. If you do not see the development window, press the **Develop** button.
3. In Genero Studio, set your configuration for the **iOS Dev Client** and start your app.
4. In the Genero Development Client interface on your iOS device, click **Connect**.
   
   The app displays on the device.
5. Open a browser and enter in the HTTP VIEWING URL.
   
   The GMI Information page displays.
6. Click **AUI Tree**.
   
   The AUI Tree displays.

As you navigate the app on your device, you must reload your browser to view the changes in the AUI tree.

**Viewing the program logs**

You can view the program logs in a browser.

- **View program logs (Android)** on page 621
- **View program logs (iOS)** on page 622
- **View program logs (Development Client)** on page 622

**View program logs (Android™)**

You can view the program logs for your Android™ mobile app in a browser. The output displays VM messages (standard output and standard error).

Before you begin:

- Your app must be the deployed debug version of the app. You cannot view the standard output and error logs when running an app in developer mode.
- You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

1. To view the standard output and error logs in a web browser:
   
   a) To run the app from your device, launch your app in debug mode. See **Run the debug version of a deployed app (Android)** on page 616.
   
      The application launches and the initial form displays.
   b) Swipe down on the icon bar and locate the entry for the Debug service. Make a note of the URI address provided.
      
      The URI takes the form of `http://<device_ip_address>:<port_number>`.
   c) Open a browser and enter in the URI from the previous step.
   d) Click **VM Output**.
      
      The standard output and standard error messages display.

2. To view using the **Display Standard output and error** menu option:
   
   a) To run the app from your device, launch your app in debug mode. See **Run the debug version of a deployed app (Android)** on page 616.
      
      The application launches and the initial form displays.
b) Select Tools > Android Tools > Display Standard output and error. The program logs (standard output and standard error messages) are written to the Output view.

c) To stop the program logs from appearing in the Output view, select Tools > Android Tools > Stop display Standard output and error.

View program logs (iOS)
You can view the program logs for your iOS mobile app in a browser.

Before you begin:

• Your app must be able to run in development mode or be the deployed debug version of the app.
• You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

Important: Sensitive and personal data may be written to the output. Make sure that the log output is written to files that can only be read by application administrators.

1. On the iOS device, select Settings > Genero Mobile.
2. Enable GUI LOGGING.
3. Open Genero Mobile (GMI).
4. In the INFORMATION section, note the HTTP URL provided.
   For example, HTTP URL=http://192.168.0.160:6400
5. On your development machine, open your project in Genero Studio.
6. Start your app.
7. Open a browser and enter the HTTP URL identified in a previous step.
8. Click Logs.
   The program logs display in the browser.

View program logs (Development Client)
While viewing your app in the Genero Development Client, you cannot view the program logs in a browser.

The Genero Development Client acts as a remote GUI client, and therefore does not have logging options for the client. In order to get logs for an app displayed in the Genero Development Client, you need to enable the virtual machine's --start-guilog option when starting the app.

Important: Sensitive and personal data may be written to the output. Make sure that the log output is written to files that can only be read by application administrators.

See Front-end protocol logging in the Genero Business Development Language User Guide for details regarding the --start-guilog option.

Viewing the device logs
You can view the device logs in a browser.

The app must be running in developer mode or be the debug version of a deployed app.

Note: At this time, only Genero Mobile for Android™ offers this feature.

• View device logs (Android) on page 622

View device logs (Android)
You can view the Android device logs in a browser on your development machine.

Before you begin:

• Your app must be able to run in development mode or be the deployed debug version of the app.
• You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

1. If you are going to run your app in developer mode, complete these steps.
a) From Genero Studio, launch your app.
b) Select Tools > Android Tools > Show Device Logs.
   The device logs display in the Output view. New log entries will be appended in real time.

2. If you are going to run the deployed debug version of your app, complete these steps.
a) To run the app from your device, launch your app in debug mode. See Run the debug version of a deployed app (Android) on page 616.
   The application launches and the initial form displays.
b) Swipe down on the icon bar and locate the entry for the Debug service. Make a note of the URI address provided.
   The URI takes the form of http://<device_ip_address>:<port_number>.
c) Open a browser and enter in the URI from the previous step.
d) Click Android Logcat.
   The device logs display.

Debugging a Web Component
You can debug a web component on a Genero Mobile application.

• Debug a web component on an Android device on page 623
• Debug a web component on an iOS device on page 623

Debug a web component on an Android device
You can enable the debugging of a Web component through Chrome.

This procedure requires:
• GMA 1.1 or greater.
• Android 4.4 or greater.

You need to enable the GMA debug mode and follow the instructions provided at http://developer.chrome.com/devtools/docs/remote-debugging#configure-webview

1. Connect your device to your desktop with a USB cable.
2. If GMA is running on your device, stop GMA on your device.
3. Enable USB debugging on the device.
4. Run your application with a web component.
   Tip: Please be patient as the application may take longer to start.
   GMA starts with debug service enabled.
5. Open Chrome on your desktop.
6. In the Chrome address bar, enter chrome://inspect/#devices.
7. Check Discover USB devices.
8. Accept the USB debugging on your device.

A list of URLs appear. You can inspect the individual pages.

Debug a web component on an iOS device
This configuration allows you to debug a web component on an iOS device.

For GMI, you can only debug with the iOS simulator. You cannot debug on the physical device.

1. Start the app.
2. Navigate to the webview you want to debug.
3. Open Safari.
4. Make sure the Show Develop menu in toolbar is enabled in Preferences > Advanced.
5. Go to Develop > iOS Simulator and choose the html to debug.
Source Code Management (SCM)

The Genero Source Code Management (SCM) module enables collaborative sharing and maintaining of the files in Genero projects.

- What is the SCM module? on page 624
- Using the SCM module on page 625
- SCM module reference on page 636

There are many software version and revision control systems that you can use to manage your source code. Genero Studio for Windows® includes Apache's Subversion client. Genero Studio for GNU/Linux® relies on Subversion 1.6.2 or later, which must have been installed on the system. For more information about Apache's Subversion, see: http://subversion.apache.org/

Although Genero Studio provides Subversion (SVN), you can use the version control system of your choice. The Locate in System File Browser contextual menu option allows for easy access to any third-party revision control system which is integrated in the operating system file browser.

What is the SCM module?

The Source Code Management module (SCM) options allow you to manage files in a Subversion (SVN) repository. The Subversion repository stores current and historical versions of the files, allowing you to recover older versions of your data when necessary. Your files (called working copies) are stored in an ordinary directory on your local file system. When you make changes to these working copies, you commit the changed files to the SVN repository. If projects or project files have already been committed to a repository, you can load copies into a working directory on your local system.

A SVN repository often holds the files (or source code) for several projects; usually, each project is a subdirectory in the repository's file system. Your working copy will usually correspond to a particular subtree of the repository.

If a Project is under Source Code Management, the icons for the nodes will have a status icon superimposed when you open the Project.

Figure 302: Project view with status icons
Table 188: Status icons and their meanings

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>Item is unchanged since it was last committed to the repository.</td>
</tr>
<tr>
<td>⚠</td>
<td>Item has been changed and has not been committed yet. The application and the project will also have change indicators.</td>
</tr>
<tr>
<td>+</td>
<td>Item is scheduled for addition to the repository.</td>
</tr>
<tr>
<td>⚠</td>
<td>Item is in conflict with updates received from the repository.</td>
</tr>
</tbody>
</table>

Related tasks
Specify a Subversion client on page 645
To specify the location of your SVN client, select Tools > Preferences > Source Code Management > Subversion from the main menu.

Using the SCM module
How to use Subversion (SVN) from the Source Control Management module.
- Checkout files on page 625
- Add files on page 626
- Commit / Review changes on page 626
- Locking on page 627
- Revert changes / Un-add files on page 629
- Delete files on page 630
- Update / Update All on page 630
- Cleanup on page 631
- Copy working files and directories on page 631
- Revert from a single revision on page 631
- Merge and revert on page 631
- Move a working copy (Switch) on page 632
- Create patch on page 632
- Apply patch on page 632
- Browse repository on page 633
- View log information on page 633
- Specify the revision range for logs on page 633
- Blame on page 634
- Diff with revised file on page 635

Checkout files
You can checkout a project from the SCM menu or directly from the location of your new working directory.
Before you begin, you must have access to an SVN Repository that has been set up to store your versioned files. Contact your system administrator or Subversion vendor for information and documentation on the set up of Subversion and repositories.

1. Identify the directory or directories on your local system that will contain the files you checkout from the repository.
2. Navigate to the directory that will serve as your new working directory. If the directory does not exist, create it.
3. Check out the files from the repository to your checkout directory. Right-click on the directory and select SCM > Checkout.
   An SVN Checkout dialog will guide you through the checkout steps and the checkout directory path will be filled in for you.
   
   **Tip:** You can also access the SVN Checkout dialog from the main menu by selecting SCM > Checkout, however you will then have to provide additional directory information.
4. Once the files have been added to your checkout directory, you are prompted to select a project file (4pw). If one does not exist, you can create a new project and save it in the checkout directory.
   The files to be versioned must be stored in the checkout directory.
5. Use File > New to create files and store them in the checkout directory and the project structure as needed.
6. Any files from the checkout directory that have been added to the Project can be committed to the repository.

**Note:** Although the 4pw project file is used internally by Genero Studio, we recommend that you include it in the checkout directory and commit it to the repository to share it with other developers. Although the 4pw and .deps files are in the checkout directory, they do not have to be added to the Project Structure.

**Add files**
Complete this procedure to add unversioned files found in the checkout directory.

1. Select Window > Views > SVN Status.
2. From the SVN Status view, navigate to your checkout directory if it is not already displayed.
3. Click Add mode.
   All unversioned files found are listed. The files/folders with 'not a working copy' text status or none property status within an unversioned folder will also be shown.
4. Check those files you wish to add to the repository.
5. Click the Add button.
   The files are added, their text status changes to 'added'. They are not yet committed.
6. To commit these newly added files, select Add Mode again to change views and select Commit.
7. To revert this action (for files that have been added but not committed), see Revert changes.

**Commit / Review changes**
To commit your updated version of a file to the repository, right-click on the file in your Project and choose SCM > Review Changes.

From the SVN Status view, check the box in the Select column for the files you wish to commit, and press the Commit button at the top of the view. This will invoke the SVN Commit dialog to commit any checked file.
Figure 303: SVN Commit dialog

Commit Comments
Enter a comment in the text box, or select a comment from the list of recent comments.

Files to unlock on commit

Unlock committed files (default)
Unlock only those locked files that are committed. Any locked file that is not committed will not be unlocked on commit.

Unlock all files
Unlock all locked files after commit. If some of the locked files are modified but not committed, you are warned that some of the modified files will be unlocked and asked whether to proceed with commit.

Keep all locks
No file is unlocked after commit.

Related concepts
SVN Status view on page 639
The SVN Status view displays information about changes to your working copy.

Locking
Locking a file provides exclusive rights to a user for changing that file in the repository. As long as the file in the repository is locked by a user, no other user can change (commit) that file in the repository. This helps to avoid conflicting commits.

Subversion provides a property svn:needs-lock. This property can be set on files only. The svn:needs-lock property sets the file to read-only when checked out. A user who has checked out the file will have to lock the file to modify
it in the working copy. If another user wants to modify the same file, they will have to first acquire the lock on the same file and is informed that the file is already locked by another user. In this way a locked file cannot be modified simultaneously which avoids possible conflicts.

**Related concepts**

**SVN Locks view** on page 638
The SVN Locks view shows the lock and lock information for locked files in the given checkout directory.

**SVN Lock strategies**
There are three user strategies for working with locks in Genero Studio; use svn:needs-lock on some files, on all files, or not at all.

**Use svn:needs-lock on some files**
There are some file types like binary files, which cannot be merged. Setting svn:needs-lock on those files specifies that only one user can modify them at one time. You must lock the file before modifying or check for existing locks. If the file is not locked, you may have to merge the changes on commit. Locked files must be unlocked after commit.

**Use svn:needs-lock on all files**
With this strategy, no commit conflicts can occur because no two users can modify the same file at the same time. You will have to lock the file before modifying it. Locked files must be unlocked after commit.

**Not using the svn:needs-lock property**
This strategy can be used when all the files can be merged manually in case of conflicts. In case of binary files it is not possible to merge the changes, so modifying the files without locking must be avoided. If the file is not locked and modified, you may have to manually merge the changes in case of commit conflicts.

**Set svn:needs-lock property**
Setting svn:needs-lock on a file specifies that only one user can modify it at one time.

1. Use **SCM > Properties** from the right-click context menu on a file to make changes to SVN properties.
2. Use the plus sign to add a new property. Select svn:needs-lock from the **Property name** list.
3. Set the **Property value** to *.
Figure 304: Set svn:needs-lock

Related concepts
SVN Lock dialog on page 644
The SVN Lock dialog is used to lock a file.

Lock a file
If svn:needs-lock has been set on a file, the current user can lock the file.
1. From the project or SVN Status view, right-click on the file and select Lock.
2. Enter a comment and press the Lock button.

Related concepts
SVN Lock dialog on page 644
The SVN Lock dialog is used to lock a file.

Revert changes / Un-add files
You can discard changes to files not yet committed, and you can unadd files which have been added but not yet committed.
1. Open the SVN Status view.
   If you are in Commit mode, the Revert button displays. If you are in Add mode, the Unadd button displays.
2. Check those files you wish to revert or unadd.
3. Click Revert (Commit mode) or Unadd (Add mode).
4. A dialog asks you to confirm your selection.
   Click Yes.

For modified files, the change is reverted and the previously committed version of the files is restored to your local (working) copy. For added files, the file is changed from 'added' to 'unversioned'.
Delete files

The **Delete** command deletes a file from your checkout directory and from the repository.

1. Right-click the file.
2. Select **Delete** from the context menu.

Your local copy is deleted. The next time you commit, the Projects view will request the source code management system delete the file from its repository.

Update / Update All

**Update / Update All** updates any outdated files in your project with the latest version stored in the repository.

The Update options are presented as buttons in the Status view Toolbar, or can be accessed right-clicking on a specific file selected from the project tree in the Projects view.

![SVN Update dialog](image)

**Figure 305: SVN Update dialog**

Specify what files to update in the SVN Update dialog.

In the **Update** combobox, specify which files to update. Choices are:

- **All** (a full checkout)
- **Selection** (selected files)

In the **Directory content to update** combobox, specify the depth for the update.

- With an All update, the depth is automatically set to Recursive.
- With a Selection update, the choices are Recursive, None, Immediate files only, or Immediate files and folders only. For updates from the Projects view using the context menu, the depth will always be None.

Select the **Omit externals** checkbox to exclude external items. External items can be of two types:

- If a working copy contains items from some other location of same repository, these items are called external for this working copy.
- If there are two repositories and one of the repositories (R1 for example) contains items from other repository (R2 for example), these items are called "externals" for repository R1.

In the **Revision** combobox, select the revision to be updated. The **Select Revision** button opens the **SVN Log Select revision** dialog.

**Related concepts**

- [SVN Log view](page 637)
The **SVN Log** view displays information about the revisions in a repository.

### Cleanup

The **Cleanup** command cleans up the working copy, removing stale locks.

1. In the Projects or Files view, right-click on a file or directory.
2. Select **SCM > Cleanup**.

### Copy working files and directories

A Subversion working copy is an ordinary directory tree on your local system, containing a collection of files. You can create another copy of your project, project files, or directories, to allow parallel development, for example.

Before you begin, you must have access to an SVN repository that has been set up to store your versioned files. Contact your system administrator or Subversion vendor for information and documentation on the set up of Subversion and repositories.

1. Identify the working directory you wish to copy.
2. In the Files tab, navigate to the working directory you wish to copy.
3. Right-click on the directory and select **SCM > Copy**.

   The SVN copy dialog displays.

   **Tip:** You can also access the SVN copy dialog from the main menu by selecting **SCM > Copy**, however you will then have to provide additional directory information.

4. Complete the **SVN copy dialog**. The From section populated for you.
5. Copy the files.

### Revert from a single revision

Follow this procedure to revert changes from a single revision.

1. Select **SCM > Merge/Revert...** from the main menu to open the **Merge/Revert dialog** on page 641.
2. Enter the repository URL and version, leaving the Range ends with a different URL check box not selected.
3. Specify the working copy directory to be updated in the working copy path.
4. Select the desired merge options
5. Click **Merge/Revert**.

### Merge and revert

Follow this procedure to merge and revert changes from a range of revisions.

1. Select **SCM > Merge/Revert...** from the main menu to open the **Merge/Revert dialog** on page 641.
2. Enter the repository URL and Revision.
3. Check the Range ends with a different URL check box and specify the revision URL and version to close the range.
4. Specify the working copy directory to be updated in the working copy path.
5. Select the desired merge options
6. Click **Merge/Revert**.
Move a working copy (Switch)

The Switch subcommand allows you to move a working copy to a new branch of the repository.

![Figure 306: SVN Switch dialog]

Create patch

You can create a patch file containing the differences between your working copies and the corresponding files in the repository. The created file is in Unidiff format.

If one of the checked files is unversioned or missing, the Create Patch command is disabled.

To create the patch file:

1. In the SVN Status view on page 639, check the file(s) that contain the modifications to be included in the patch.
2. Click Create Patch.
3. In the Save file dialog, select a directory and enter a name for the patch file.

The patch file is created and opened in Code Editor.

Apply patch

The Apply patch command applies a patch file to files in the repository.

This feature is available for Windows™ and Linux™ users only.

1. Identify the patch file to be used.
2. Select SCM > Apply Patch...
3. Complete the Apply Patch dialog on page 640.
4. Press Apply.

A page containing the output of the command is displayed. The output will indicate Successful when all the files from the patch file are successfully patched. If only some of the files are patched successfully, the output will indicate Aborted.

Related concepts

Apply Patch dialog on page 640
The Apply Patch dialog is used to apply a patch file. Only text modifications are applied.

Related tasks

Create patch on page 632
You can create a patch file containing the differences between your working copies and the corresponding files in the repository. The created file is in Unidiff format.

**Browse repository**

The SVN Repository lists the files in your selected repository and branch.

1. Select **Window > Views > SVN Repository** to open the SVN Repository Browser.
2. Enter the URL of the repository or select it from the drop down list.
3. Select the desired version. To select a specific revision within the version, click the **Select Revision icon** and modify the selection range.
4. Click the **Refresh icon**.

**View log information**

Complete this procedure to view information about the revisions for a specified repository.

1. Select **SCM > Show Log**.
2. Enter the URL of the repository. Click the Browse repository icon to display the Repository browser.
3. Select the desired version.
4. Click the Fetch the revisions in the revision range icon (single blue arrow).
   The SVN Log view displays. The Revisions, Actions, Author, Date, and Messages are displayed in the log. The icons for Actions indicate the status.
5. To limit the list of revisions displayed, open and complete the Revision log dialog.
6. Select a specific revision to display information about that revision.
7. The comments and actions/files associated with that revision are displayed.

![Figure 307: SVN Log view](image)

**Specify the revision range for logs**

You can select the range of revision logs to display using the Revision log options dialog.

1. Click the **Open revision log options icon**.
2. Provide the range in the from: and to: fields, respectively.
3. Click **OK**.
   The view is updated to display only those revisions in the range specified.
4. Use the **Fetch** actions to retrieve more log entries.

**Related concepts**

*SVN Log view* on page 637
The **SVN Log** view displays information about the revisions in a repository.

**Blame**

The **Blame** command shows author and revision information inline for specified files or URLs.

The Blame view displays the file as read-only in the Code Editor. Each line of text is annotated in the left margin with the author (username) and the revision number for the last change to that line. On clicking the left margin, the log comment for the revision displays.

**Operations available in Blame view**

Blame view operations can be accessed from the Toolbar.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show log</td>
<td>Shows the revision log view and the log entries starting from the selected revision to the first revision.</td>
</tr>
<tr>
<td>Blame in New Tab</td>
<td>Shows a new Blame view, where the maximum blame revision is the selected revision and the minimum blame revision is the minimum revision for the current Blame view.</td>
</tr>
<tr>
<td>Blame</td>
<td>Updates the current Blame view, where the maximum blame revision is the selected revision and the minimum blame revision is the minimum revision for the current blame view.</td>
</tr>
<tr>
<td>Diff with previous revision</td>
<td>Shows the difference between the selected blame revision and the previous revision.</td>
</tr>
<tr>
<td>Next / Previous</td>
<td>If you have viewed blame multiple times in the Blame view, you can scroll through the views using the next and previous options.</td>
</tr>
</tbody>
</table>

**Access the Blame view**

As a prerequisite:

The file must be a versioned file. The blame option is not available for Added files.

Access the Blame view for author and revision information.

1. Open one of these views:
   - File Browser
   - Projects
   - SVN Status
   - SVN Log (If the viewed log is about a file, then the blame option is available.)
   - SVN Repository Browser
2. Select one or more files.
   If multiple files are selected, separate blame views are opened for each file.
3. Right-click on the file and select a Blame option.

<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCM &gt; Blame</td>
<td>Blame information fetches with the revision range from 1 to HEAD.</td>
</tr>
<tr>
<td>Menu option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SCM &gt; Blame...</td>
<td>SVN Blame configuration dialog displays. If multiple files are selected, the dialog displays once and the selected settings are used for all files.</td>
</tr>
<tr>
<td></td>
<td>• Specify the range.</td>
</tr>
<tr>
<td></td>
<td>• Click OK.</td>
</tr>
</tbody>
</table>

If you are in the SVN Log view:

- If one or two revisions are selected, both **Blame** and **Blame...** are available.
- If one revision is selected, the **from** is 1 and the **to** is the revision number selected.
- If two revisions are selected, the **from** is the smaller revision number and the **to** is the greater revision number.
- If no revision is selected, the **from** is 1 and the **to** is HEAD.

If you are in the SVN Repository Browser view, the **to** is currently selected revision.

**Related concepts**

Blame on page 634

The **Blame** command shows author and revision information inline for specified files or URLs.

**Diff with revised file**

Ctrl-Click to select two revisions of a file in the log. Select **Diff** from the right-click Menu to call the Using the Diff tool on page 451 utility to compare the committed file with the file that you have revised.

**Integrate bug tracking**

Changes made in your project may also be related to a specific bug or issue ID. If you use a bug tracking system (such as Bugzilla) you can associate the changes you make in your Source Code Management system (such as Subversion) to its bug tracking ID.

![Figure 308: Source Code Management capturing bug ID](image-url)
Integrate bug tracking
In Subversion, add specific 'bugtraq' properties on the repository. See SVN documentation for more information about usage for each of these properties.

- **bugtraq:logregex**: Activates the bug tracking system.
- **bugtraq:message**: Activates an additional field on the Commit dialog to prompt the user for an issue number.
- **bugtraq:label**: Label for the additional field on the Commit dialog.
- **bugtraq:warnifnoissue**: Display a Warning message if the added field is empty when doing a commit.
- **bugtraq:url**: URL of the bug in the tracking system.

Figure 309: Source Code Management displaying bug ID

Figure 310: Add properties on SVN repository

SCM module reference
Reference information for Source Code Management.

- Views on page 637
- Dialogs on page 640
- Specify a Subversion client on page 645
• SVN error messages on page 645

Views
To display views that are not visible, use the Window > Views menu.
The SVN Repository view is covered in the topic Browse repository on page 633.
• SVN Log view on page 637
• SVN Locks view on page 638
• SVN Status view on page 639

SVN Log view
The SVN Log view displays information about the revisions in a repository.

Figure 311: SVN Log view

| Repository URL | Location of repository of which to view logs. |
| Repository Browser | Display Repository Browser. See Browse repository on page 633. |
| Fetch entries in the revision range | Fetch entries in current revision range. |
| Fetch next 100 entries | Fetch next 100 entries in current revision range. |
| Fetch next 5000 entries | Fetch next 5000 entries in current revision range. |
| Open revision log | Specify revision range. See Specify the revision range for logs on page 633 |
| Open diff view | See Diff with revised file on page 635. |
| Open property diff view | See diff information for SVN properties. |
| Properties | See properties set on file. SVN Properties dialog on page 644. |
| Show repository browser | Display Repository Browser. See Browse repository on page 633. |
| Edit log comment | Edit log comment of selected entry. |
Revert changes from selected revisions
See Merge and revert on page 631.

Revert to this revision
See Merge and revert on page 631.

Show Blame information
See Blame on page 634.

SVN Locks view
The SVN Locks view shows the lock and lock information for locked files in the given checkout directory.

From the SVN Locks view you can unlock your locked copies and steal locks from other users if forced acquisition of a lock is necessary.

Figure 312: SVN Locks view

Checkout directory path
Enter a working copy directory path to view locked files in that directory.

Browse
Navigate to a checkout directory to populate the checkout directory path.

Refresh
Refresh the data in view.

Unlock
Unlock the selected file.

Steal Lock
Forcefully acquire the lock if locked by another user.

My locks
Display locks owned by the current user.

File list
- File: Displays file path relative to checkout directory path.
- Text Status: File text status in current working copy.
- Property Status: File property status in current working copy.
- Lock Type: Information about whether the file is locked in current working copy or not.
- Working copy: The file is locked in current working copy.
• Repository only: The file is locked in repository not in current working copy.
• **Lock Owner**: User who has locked the file.
• **Lock Comment**: Lock comment given by the lock owner when the file was locked.
• **Lock Date**: Date when the file was locked.

**SVN Status view**
The SVN Status view displays information about changes to your working copy.

![SVN Status view](image)

**Figure 313: SVN Status view**
Check the file or files to affect. The spacebar can be used to check/uncheck the current item. Toolbar icons allow you to execute commands on the selected files. The files displayed, and the icons enabled, depend on the mode and the status of the files selected.

The text status of a file can be:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>added</strong></td>
<td>The file has been added to the repository and needs only to be committed.</td>
</tr>
<tr>
<td><strong>modified</strong></td>
<td>The file has been modified in the checkout directory and the update has not yet been committed.</td>
</tr>
<tr>
<td><strong>normal</strong></td>
<td>A committed file in the repository.</td>
</tr>
<tr>
<td><strong>unversioned</strong></td>
<td>The file has not yet been added to the repository. After being added, it should be committed.</td>
</tr>
<tr>
<td><strong>conflicted</strong></td>
<td>The file was modified in both the repository and checkout directory, and Subversion cannot resolve the conflict. Use the menu option <strong>Merge</strong>.</td>
</tr>
<tr>
<td><strong>not working copy</strong></td>
<td>All the files within an unversioned folder will have this status.</td>
</tr>
<tr>
<td><strong>missing</strong></td>
<td>The file has been deleted from the checkout directory. If you commit a file marked as missing, it will be skipped during commit and will remain as it is.</td>
</tr>
</tbody>
</table>
incomplete

If the checkout has stopped without completion, the directory will have this status.

SVN Status modes

Commit mode

Files that can be committed are displayed. Files that cannot be committed but are not supported in Genero Studio are displayed (with the exception of files with 'ignored' and 'external' status). Unversioned files are not displayed. This is the default mode.

Add mode

Files with a status of added, copied/moved, unversioned, or 'not a working copy' are displayed.

To switch between modes, click the Add mode button.

Dialogs

Information about SCM dialogs.

- **Apply Patch dialog** on page 640
- **Merge/Revert dialog** on page 641
- **The SVN Copy dialog** on page 642
- **SVN Checkout dialog** on page 643
- **SVN Lock dialog** on page 644
- **SVN Properties dialog** on page 644
- **SVN Repository view** on page 645

**Apply Patch dialog**
The Apply Patch dialog is used to apply a patch file. Only text modifications are applied.

![Apply Patch dialog](image)

**Figure 314: Apply Patch dialog**

**Patch file**
The path of the selected patch file.
Source Code Management (SCM)

**Root dir**  
The path of the directory where the patch is to be applied.

**Skip directories**  
Integer value defining the directory level to be search for the file to be patched.

**Create file if not existing**  
If the file to be patched is not present in the current directory path, the corresponding file will be created as an unversioned file. If the file is unchecked in the SVN Status view on page 639, the corresponding file will not be created.

**Test apply patch**  
Displays the output of the Apply Patch command without actually applying it to the files; files will not be modified.

**Show files**  
Opens a dialog displaying a list of all the files that will be patched.

**Merge/Revert dialog**  
The SVN Merge/Revert dialog lets you merge or revert revisions.

![SVN Merge/Revert](Figure 315: SVN Merge dialog)

- **URL** - Enter the URL of the repository and select the Revision number, or HEAD for the most recent revision
- **To URL** - Enter the URL of the repository and Revision number at which you want to end, or click the Use the start URL button to continue to the end.

To access the dialog:
• Select **SCM >> Merge/Revert** from the Genero Studio menu.
• Right-click on a file and select **SCM >> Merge/Revert** from the context menu.
• Right-click on a revision entry from the SVN Log view and select either Revert changes from selected revisions
  OR Revert to this revision

You can revert the changes from a single revision, or you can revert the changes across a range of revisions.

**The SVN Copy dialog**
The SVN Merge/Revert dialog lets you copy project files.

![SVN Copy dialog](image)

**Figure 316: SVN Copy dialog**

Select the corresponding radio buttons (Working Copy/URL ) to indicate that you will:

• Copy the working copy of the project file from which you invoked the SVN copy dialog to a different repository
  having the URL that you have entered. Select the revision (HEAD is the default).
• Copy the selected file from a repository having the URL that you have entered, to a working copy having the
  location that you have entered. Select the revision (HEAD is the default).
• Copy the selected file from one repository to another. Select the revision (HEAD is the default).

The Copy result is shown.
**SVN Checkout dialog**

The SVN Checkout dialog guides you through the checkout process, allowing you to load copies of repository files into a checkout directory on your local system.

![SVN Checkout dialog](image)

**Figure 317: SVN Checkout dialog**

The **Repository** section:

- **URL of repository** - enter the URL of the SVN repository that you wish to access.
- **Checkout directory path** - enter the path of the directory for the checked out files; the Browse button allows you to locate the path.
- **Checkout depth** - choose the level you desire; for example, "only file children" will checkout the files but not the directories.
- **Omit externals** - check to exclude "external items".

The **Revision** section:

Specify the particular version from which to checkout the files. The Select Revisions button opens the SVN Log Select revision dialog.

**After the Checkout button is clicked**

Messages related to the checkout operation display in the **Output View**.

The results of the Checkout and other SCM commands are displayed in **Results** pages. A **Run in background** button, at the bottom of the SCM dialogs, allows you to close the dialog but continue running the process in the background, as indicated in the Task Manager.
**SVN Lock dialog**

The SVN Lock dialog is used to lock a file.

![SVN Lock dialog](image)

**Figure 318: SVN Lock dialog**

- **Comment**: Enter a comment to lock the file.
- **Steal locks**: Forcefully acquire the lock if locked by another user.

**SVN Properties dialog**

Use SCM > Properties from the right-click context menu on a file to make changes to SVN properties.

Use the integrated Toolbar to add, modify, or delete properties. See your Subversion manual for additional information about properties.

![SVN Properties dialog](image)

**Figure 319: SVN Properties dialog**
SVN Repository view
The SVN Repository displays the contents of the selected repository and version.

Figure 320: SVN Repository Browser
When a user does not have the rights to access a directory in the repository, the directory is marked with a forbidden icon ( ).

Specify a Subversion client
To specify the location of your SVN client, select Tools > Preferences > Source Code Management > Subversion from the main menu.

1. Select Tools > Preferences > Source Code Management > Subversion from the main menu.
2. In the SVN client path, enter or browse to the path of your SVN client; include the bin directory in this path.
3. Press Check Version to validate the SVN client and to show SVN version details.

Related concepts
Setting Preferences on page 128
Customize Genero Studio to meet your needs.

SVN error messages
A list of SVN error messages. For messages that are not self-explanatory, additional information is provided.

Important: It is likely that the error is an SVN error. When Genero Studio executes an SVN command, it asks the SVN executable to perform the command. Genero Studio displays the result, success or error. You will likely have to refer to the official SVN documentation to identify the problem and find the solution.

Table 189: SVN Error Messages

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-22001</td>
<td>SVN Tool error.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-22003</td>
<td>Cannot start SVN Tool.</td>
</tr>
<tr>
<td></td>
<td>The SVN Tool is not properly set in preferences.</td>
</tr>
<tr>
<td></td>
<td><strong>In Tools &gt; Preferences &gt; Source Code Management &gt; Subversion</strong>, set and check the SVN client path.</td>
</tr>
<tr>
<td>GS-22004</td>
<td>SVN Add command failed.</td>
</tr>
<tr>
<td>GS-22005</td>
<td>SVN Commit command failed.</td>
</tr>
<tr>
<td>GS-22006</td>
<td>SVN Export command failed.</td>
</tr>
<tr>
<td>GS-22007</td>
<td>SVN import command failed.</td>
</tr>
<tr>
<td>GS-22008</td>
<td>SVN Info command failed.</td>
</tr>
<tr>
<td>GS-22009</td>
<td>SVN mkdir command failed.</td>
</tr>
<tr>
<td>GS-22010</td>
<td>SVN remove command failed.</td>
</tr>
<tr>
<td>GS-22011</td>
<td>SVN rename command failed.</td>
</tr>
<tr>
<td>GS-22012</td>
<td>SVN rename URL command failed.</td>
</tr>
<tr>
<td>GS-22013</td>
<td>SVN resolved command failed.</td>
</tr>
<tr>
<td>GS-22014</td>
<td>SVN revert command failed.</td>
</tr>
<tr>
<td>GS-22015</td>
<td>SVN log command failed.</td>
</tr>
<tr>
<td>GS-22016</td>
<td>SVN status command failed.</td>
</tr>
<tr>
<td>GS-22017</td>
<td>SVN update command failed.</td>
</tr>
<tr>
<td>GS-22018</td>
<td>Apply patch failed.</td>
</tr>
<tr>
<td>GS-22019</td>
<td>Server certification error.</td>
</tr>
<tr>
<td></td>
<td>In the displayed dialog, selection an option:</td>
</tr>
<tr>
<td></td>
<td><strong>Accept permanently</strong></td>
</tr>
<tr>
<td></td>
<td>You will not be asked for the server certificate for the same repository again.</td>
</tr>
<tr>
<td></td>
<td><strong>Accept Once</strong></td>
</tr>
<tr>
<td></td>
<td>Accepts the certificate only for the current operation. For the next operation, this dialog will appear again.</td>
</tr>
<tr>
<td></td>
<td><strong>Reject</strong></td>
</tr>
<tr>
<td></td>
<td>Cancels the operation.</td>
</tr>
<tr>
<td>GS-22020</td>
<td>SVN Authentication failed error.</td>
</tr>
<tr>
<td></td>
<td>The current user does not have access to the SVN repository.</td>
</tr>
<tr>
<td></td>
<td>Provide a valid user name and password in the authentication dialog box.</td>
</tr>
<tr>
<td>GS-22021</td>
<td>SVN proplist command failed.</td>
</tr>
<tr>
<td>GS-22022</td>
<td>SVN Copy command failed.</td>
</tr>
<tr>
<td>GS-22023</td>
<td>SVN Lock process failed.</td>
</tr>
<tr>
<td>GS-22024</td>
<td>SVN Unlock process failed.</td>
</tr>
<tr>
<td>GS-22025</td>
<td>SVN Cat command failed.</td>
</tr>
<tr>
<td>GS-22051</td>
<td>Item found in checkout. Double click to open.</td>
</tr>
</tbody>
</table>

**Important:** This feature is not supported on mobile platforms.

- What is Genero Report Writer (GRW)? on page 647
- gst-topics/grw_section_getting_started.ditamap
- Create a report program on page 671
- Create the data schema on page 774
- Create a report design document on page 777
- Create and manage report templates on page 986

### What is Genero Report Writer (GRW)?

Genero Report Writer provides a means for creating high-quality reports.

- GRW overview on page 647
- GRW runtime architecture on page 649

### GRW overview

With Genero Report Writer, you create a report application that serves as a data model. The data is retrieved from a data source (typically a database) and streamed to a layouter, which lays out the data according to the specifications of a report design document.

The data may come from a database, a data file, an XML file, or a Web service. The separation of data from layout enhances security, facilitates reuse, and protects the report from changes to the data storage.

Genero Report Writer separates the data source from the report design. Both data and design are used to produce the report output.

### Data source

The data source (a.k.a. the data model) is defined by the report application, which is created using BDL. There are two ways to create the report application: generate the application or write the application by hand.

If you are retrieving the data from a database, you can use the Business Application Modeler (BAM) to generate the SQL data source. The BAM provides easy-to-use wizards that generate the source to use as a library for inclusion in your applications or, if needed, as a stand-alone reporting application. The BAM is part of Genero Studio.

If you are not using SQL to retrieve data from a database, or if you simply choose not to have your data model generated, you manually write your application source using BDL.

### Report design

The Genero Report Designer (GRD) is the tool used to create a report design document. It sets the layout of the data. Genero Report Designer is included in Genero Report Writer; it can also be installed as a stand-alone application.

Genero Studio includes a number of inbuilt report templates, which you can use to create your own reports and modify as required. If you are writing multiple reports that use the same design, a report template makes this process easier and ensures consistency.

To use the Genero Report Designer, the only input needed is a data schema. The data schema displays the data as objects, organized in a tree view, for inclusion on the report. The data schema also tells how the data is grouped and sorted, allowing the report designer to add report summaries (such as subtotals) and breaks. The data schema is extracted from the BDL report program.
Report output

A Genero report is based on XML standards (XML reports, XML data, SVG output). Since the data is in XML format, you can have multiple views of the same data. The report can be output to various formats, or directly to a printer or other output devices.

You can use Genero Report Writer to create reports for a variety of purposes. For example, you can produce:

- Lists
- Reports with a customized format, such as invoices, corporate documents, or accounting reports
- Reports that include images
- Reports on pre-printed forms
- Labels
- Business Graphs (such as MapCharts, Category Charts, and XY Charts)
- XML documents
- HTML documents
- Excel™ spreadsheets
- Documents in Microsoft™ RTF format
- ASCII-based reports, to print legacy Genero reports "as is"

The rapid delivery of pages, with no need for temporary files or temporary tables, consumes less memory and speeds up the printing of very large reports.

The report design document sets the initial output options. You can override the output options using the BDL program.

Related concepts

Business Application Modeling (BAM) on page 205
Genero Studio Business Application Modeling (BAM) is a graphical design tool to generate the code that extracts data and creates data models in BDL. It automatically generates the logic and source code for a database application to query, add, update and delete rows in database tables. BAM can generate desktop, web, and mobile applications.

Create and manage report templates on page 986
A report template defines the layout of a professionally-designed report that you can use to quickly create an initial report design.

Create a report design document on page 777
The Report Design document (4rp) defines the report page, including the report data and the report elements for organizing and displaying this data.

**GRW runtime architecture**

At runtime, various software components work together to produce a Genero report.

---

**Figure 321: The Genero Report Writer runtime architecture**

The purpose of this diagram is to describe the general workflow of a report, as it is processed by Genero Report Writer. At a high level, it identifies the workflow using a generic architecture diagram.

The data may come from a database, a data file, an XML file, or a Web service. The reporting application, written in Genero Business Development Language (BDL), defines the data model using input from the data model designer.

The Genero DVM (Dynamic Virtual Machine) executes the BDL code to retrieve the data and streams data from the database to the Genero Report Engine (GRE).

The Report Designer graphically lays out the report, creating a report design document which references the data schema.

The Genero Report Engine (GRE) is the layouter. Using the report design document, it lays out paged streams to the Genero Report Viewer. Alternatively, the report can be written to a file (PDF, image, HTML, XML, Microsoft® Excel, or RTF format).

---

**Get Started with Reports**

These topics give you the information you need to initially work with Genero Report Writer.

- What is Genero Report Writer (GRW)? on page 647
- The Reports demo on page 651
- Configure fonts on page 657
Steps to a Report
To use Genero Report Writer to create a report, you need to properly prepare your environment, create the report application and report, and run and test the report.

- Set the Report Writer environment on page 650
- Create a Report on page 650
- Run the Report on page 651

Related concepts
What is Genero Report Writer (GRW)? on page 647
Genero Report Writer provides a means for creating high-quality reports.

Writing the Genero BDL report program on page 672
A Genero BDL report program retrieves the data and outputs it to a report.

Designing a Report on page 781
Design a simple or complex report using the views, menus, and tools in the Report Designer.

Set the Report Writer environment
Complete these steps to set up the Report Writer environment

1. From the Genero Studio main menu, select Tools > Genero Configurations to open the Genero Configuration Management window.
2. In the Environment Sets list, ensure that the Report Writer environment set is active (has a check mark).
3. Select the Report Writer environment set to view its environment variables. Ensure that these environment variables are correct.

   The default environment sets are listed in the Compiler/Runtime section.

   Note: At the application level, you must include the library libgre.42x. See GRW reference for Genero BDL applications on page 719.

Related concepts
Environment sets on page 164
Define sets of environment variables to add to or overwrite the default environment. These environment settings are also usable by each Genero Studio client that adds a remote Genero Studio Server to its list.

Create a Report
Creating a report involves setting up a project to store the report files, developing a report application, designing the report, and testing the report.

Before you begin, ensure that you have set the report environment correctly.

This procedure is provided to give you a quick start, or overview, for creating a report application. For each step, there are many options that are not discussed here, but are covered in other topics. Use this procedure as a guide for using Genero Report Writer to create reports.

1. Create the project.
   A project contains the nodes for storing and managing your report application files.
   a) Select File > New > Genero > Simple Project (.4pw) and click OK.
   b) Select File > Save as and complete the dialog to save your project.

   The project tree should have a Group node, and the Group node should contain an Application node, a Databases node, and a Library node. You can rename these nodes to more meaningful names.

2. Set up the project's Application node.
   a) Right-click on your Application node to create the files and folders. For example, create two virtual folders named src and reports.
   b) Select the Application node. In the Properties view, in the External dependencies list, add libgre.42x.

   For more information about the libgre.42x library file, see the GRW reference for Genero BDL applications on page 719 section.
3. Create your report application. For example, right-click on the src node, select **New File > Genero > Source (.4gl)**, and create a file named *myreport.4gl*.

   **Note:** The Genero report application (4gl) is responsible for reading data from a database (or alternate data source) and streaming it to the Genero Report Engine. For more information, see *Writing the Genero BDL report program* on page 672.

4. Generate the data schema by selecting the 4gl source file and setting its **Compiler options** property to **--build-rdd**.

   **Note:** The data schema (rdd) file identifies the fields and the grouping of data streamed by the Genero reporting application to the Genero Report Engine. It is used by the report design document, giving the designer the ability to select fields for placement on a report and setting report triggers. For more information, see *Generate a data schema from a Genero BDL report program* on page 680.

5. Create a report design document (.4rp).
   
a) Select **File > New > Report from Template**, select one of the available templates, and click **OK**.

   **Note:** You can also start with an empty report design document using **File > New > Reports > Empty Report (.4rp)** or **> List Report (.4rp)**, and then select the data schema (rdd) on the Data View tab.
   
b) Follow the steps in the New Report from Template wizard to select a data schema and add fields.

   You can now see the report design document in the main work area and the data on the Data View tab. For further information, see *Create a report from an existing template* on page 986.
   
c) Report output can be output in various formats. Select **File > Report properties > Output configuration** to change the default output for the report.
   
d) Save the report design document and add it to the reports virtual folder in your application node.

   For more information on working with the report design document, see *Create a report design document* on page 777.

6. Open the source (.4gl) file and add the necessary code to your report application.

   **Tip:** You can use the code from *Creating a simple report* on page 674 as a basis.

   Ensure that your code:
   
   • References the data schema.
   • References the report design document.

7. Build the report application.

   Right-click the report application node or the report group node in the Genero Studio Projects tab and select **Build** to compile and link the application.

**Run the Report**

Run the report from the Project Manager.

Right-click the report application node in the Project Manager and select **Execute** to run the program.

**Note:** If the report data comes directly from a database, your BDL program must have access at execution to that database.

By default a new report is in SVG format, and the output is set to Preview using the **Genero Report Viewer**. You can change these defaults in your Report Design document, or in your report program code.

**The Reports demo**

The OrderReport demo application provides a sample reporting application, along with various reports design documents.

These topics provide an overview of the general functionality of the demo application, as well as a description of the provided sample reports.

• **The Reports demo overview** on page 652
• **Run the Reports demo** on page 652
• **Genero Report Viewer for HTML5** on page 655

**The Reports demo overview**
When Genero Studio launches, the Welcome Page lists the demo projects. Select **Reports** to open the OrderReport demo project in Genero Studio.

The Reports demo project consists of:

- Application files, generated or written using Genero Business Development Language.
- A set of report definition files, created with the Genero Report Designer.
- The data schema for the database used with the demo reports.

The demo project files can be found in `Documents and Settings\<username>\Documents\My Genero Files\samples\Reports`. Each project is organized by an appropriately-named directory.

To run the demo:

- Expand the **Reports** project in the Project Manager, and right click on the **Order Report** application node.
- Select **Execute** from the context menu.

**Run the Reports demo**
When the report begins to run, the application displays a dialog box to allow you to select the report you wish to run, and the output format.

![Figure 322: Reports demo form](image)

1. Select a report from the combobox. See **Reports demo reports** on page 652 for the list of reports.
2. Select the output format. See **Reports demo outputs** on page 653 for the output options.
3. Select the desired action, to **Save On Disk** or **Preview**. See **Reports demo outputs** on page 653 for more details.

**Reports demo reports**
When you run the Reports demo, you select the desired output.

With the Reports demo, you select a report from the combobox. The reports include:

- **OrderReport**

  A report detailing a customer order. This report illustrates the use of the section property, as well as the use of different page headers and footers on each page.

- **OrderLabels**

  A report consisting of pages of mailing labels for customers. This report illustrates the labels functionality, enabling you to print multiple labels on a single report page.
| Report Writer | 653 |

**OrderList**
A report showing a text-based list, based on the list template.

**OrderStock**
A report displaying stock information, including a bar code. This report illustrates the use of attachment and positioning rules, and shows you how to design a report onto pre-printed paper using a background image.

**ListDemo**
a report in list format that includes images, totals, and conditional text coloring. This report provides two examples of RTL expressions that control the font color and the "thumbs-up/thumbs-down" image at runtime, based on a field value.

**Generic List**
A report that provides an interface that allows you to select which fields are included in the list report.

**TableDemo**
A report in table format. This report shows a simple usage of a table.

**GroupTableDemo**
A report in table format. This report shows a table with several header and body rows.

**CategoryChart**
Several examples showing a category chart, or a chart that is grouped by two fields. These examples show how different charts can be produced from the same data

**MapChart**
Several examples showing a map chart, or a chart that is grouped by one field. These examples show how different charts can be produced from the same data

**StaticPivotTable**
A table of customer data, grouped by customers and orders.

**DynamicPivotTable**
Allows you to select which fields to use as dimensions (for grouping) and which fields to include as measures. Generates a table of user-selected measures, grouped by user-selected dimensions.

**MasterReport**
A report showing the use of sub-reports.

**Reports demo outputs**
When you run the Reports demo, you determine the format of the report output, and whether it is saved to disk.

**Output formats**
The Output As section of the Reports user interface allows you to select the output format for your report.

**Image**
An image file is created. By default, with the demo app, it is created in .jpg format.
- If preview is selected, the image displays in the default image viewer for the client.
- If save on disk is selected, the image is created in the GRW demo project directory.

**PDF**
A .pdf file will be created in the GRW demo project directory. The file can be viewed with an Acrobat PDF reader.

*Note:* Static pivot tables are not yet supported in Java.

*Note:* Sub-reports are not yet supported in Java.
SVG
An SVG file will be created in the GRW demo project directory.

Browser
An SVG file will be created for viewing and is displayed using HTML5, with the Genero Report Viewer for HTML5 on page 655.

Excel XLS Spreadsheet
The report data is output in Excel format.

RTF (MS-Word)
The file is save in Rich Text Format (RTF), which provides a format for text and graphics interchange that can be used with different output devices, operating environments, and operating systems; more that just Microsoft Word.

HTML
The report document is output in HTML format

Output actions
You can either view the report immediately, or you can save the report to disk.

Preview
Displays the report according to the selected output option. A copy is not saved to disk.

Save On Disk
Saves a copy of the report to disk. The report will be saved in the demo project directory. The name of the report will be "report", with an extension of .pdf or .svg, or image000#.jpg where # is incremented sequentially from 1 for each successive image that is saved to disk.

Genero Report Viewer
If you choose SVG as the output format and Preview as the action, the report displays in the Genero Report Viewer, which is automatically installed as part of the Genero Desktop Client.

The Report Viewer is a component of the Genero Desktop Client that is automatically installed by the installation program of the GDC; it can be used to display reports that are in the SVG format only.

Important: SVG format for reports is currently not supported for previewing when running with the Web Client.
Use the **File** >> **Print** menu option in Report Viewer to send the report to a printer.

**Genero Report Viewer for HTML5**

If you choose **Browser** as the output format and **Preview** as the output action, the report displays in the Genero Report Viewer for HTML5.

Genero Report Viewer for HTML5 is a lightweight report viewer based on pure HTML5 and JavaScript technologies. It is optimized for low bandwidth and slow networks. The rendering is pixel-exact. Fonts are optimized, including good performance for Asian fonts. In terms of performance, it allows streaming and can display pages immediately as they become available. It fetches only what is necessary. The reports are static files and, as such, can be viewed at any time by sharing the relevant URLs.
In addition to displaying the report, the report viewer provides navigation options. It supports direct navigation to a specific page. The Print icon prepares the report in a single HTML page that you can then print using your browser's print functionality.

**Related concepts**

- [Genero Report Viewer for HTML5 customization](#) on page 1014

When you select Browser as the output option, files are created for the report. These files are viewed using the Genero Report Viewer for HTML5.

**Configure printers**

To output well-formatted reports, verify that the required printer is available to your server and supported by your system.

Execute the `$GREDIR/bin/printerinfo` scripts to list the available printers.

However not all available printers are supported. The supported printers must be graphical printers.

- On Windows®, a Windows® printer driver for the printer in question must be installed (unless you are using the driverless option, see below).
- On UNIX™, the printer must support Postscript, or a converter from Postscript to the printer's native language (typically a ghostscript backend) must be installed and configured. We recommend use of the Postscript Printer Description (PPD) file for your brand of printer.
- On any platform, you can use driverless, server-side silent printing if the printer is reachable over the network at TCP port 9100, and supports both the PJL and the Postscript language. For further information, see [Printing without a driver using PJL](#) on page 704.

Plain text printers or bar code printers with proprietary control languages are not supported.

Search the internet for Windows® drivers for some brands of thermal transfer bar code printers when the vendor does not provide a driver, as others have specialized in providing such drivers.

Do not use dot matrix printers in graphical mode, even if the drivers exist. The speed is typically too slow, and the print results are poor.
UNIX™ and CUPS

- CUPS should be used if available. Direct the browser to http://host:631. Always make use of the "print the test page" option during the installation.
- If CUPS is not available and the printer is listed in printerinfo but does not work, test if any other graphical application (such as OpenOffice or FireFox) can print to that printer.

Configure fonts

To output well-formatted reports, verify that the necessary fonts are available.

- Font troubleshooting on page 657
- Fonts in PDF output on page 658
- Tips for installing fonts on page 658
- Using monospace fonts in compatibility reports on page 659

Font troubleshooting

There are several reasons why a font might not display correctly in your report.

Use this checklist to troubleshoot the problem:

- Is the font available on your system? Use the $GREDIR/bin/fontinfo script to list the fonts available on the server:

  fontinfo [-V] [-h]

  where -V displays the version and -h displays the help.

  If the font is not available, you can install it with the procedure specified by the vendor.

  Note: On Linux® systems, installing fonts can be simplified by tools such as Font Manager.

  - The debugger also shows which font is used by the report. Switch debugging on by setting GREDEBUG to 9, and run the report. Debugging information appears in the standard output. For further information, see Debugging your Report Design Document on page 858.
  
  - Are you outputting your report to PDF format? Use the $GREDIR/bin/fontinfopdf script to list all fonts that can be used in PDF documents. For more information, see Fonts in PDF output on page 658.
  
  - Are you using distributed mode? Are the GRE and GRD installed on different operating systems? If so, the fonts might display incorrectly. For more information, see Tips for installing fonts on page 658.
  
  - Is Genero Studio configured to the correct locale? Check the LANG environment variable. For further information, see Configure LANG on a Genero Studio Server on page 195.
  
  - Are you embedding fonts and outputting the report to SVG? Is the specified font missing from the server side? Switch font embedding off in the SVG output by calling fgl_report_configureSVGDevice().
  
  - Does your report include Asian characters, and are you outputting to SVG or PDF format?

    Make sure the specified fonts contain the required characters. In your report design, for any text box using Asian characters, you may need to set the fidelity property to true. For compatibility reports, set the fidelity parameter to "true" in the fgl_report_configureCompatibilityOutput function.
  
  - Does your font lack certain glyphs?

    In this case, GRE attempts to take the missing characters from a different font that contains the glyphs. For example, the monospaced (fixed-width) font "Courier" does not contain graphics characters. If a report contains a grid that is drawn using graphics characters, GRE might substitute characters from a set that is not fixed-width, causing the layout to break. Avoid this problem by using a font like "Lucida Sans Typewriter" (see http://java.sun.com/j2se/1.5.0/docs/guide/intl/font.html#lucida.), which is both fixed-width and contains the required characters.
  
  - Are you using compatibility mode, and does your report use monospace fonts? For more information, see Using monospace fonts in compatibility reports on page 659.
  
  - Have you updated your version of Java™, or moved to a different server? If so, the default sans-serif font might have changed. To avoid issues with the default font, specify a font at the root of the document. In the Report
Designer, set the Font properties for the Page Root node (for example, the **Font Name**, **Font Size**, and **Text Fidelity** properties).

### Fonts in PDF output

If you output your reports to PDF format, some fonts may not display as expected.

You can set the properties for PDF reports, including the font directory, by using the `fgl_report_configurePDFDevice` function.

Not all fonts can be used for embedding in PDF, because the license flag might prevent a font from being used. The utility `$GREDIR/bin/fontinfopdf` lists all fonts that can be used in PDF documents:

```
fontinfopdf [-V] [-h] folder
```

Where `-V` displays the version, `-h` displays the help, and `folder` displays the directory for the fonts.

If a font is found by `fontinfo` but not `fontinfopdf`, it is likely that:

- In the case of a Type 1 font, the text (.pfa) is present in the font directories, but the binary font file (.pfb) is not there. Type 1 fonts need a .pfb file to generate the reports. You can compile the .pfb file from the .pfa file by using font compilation tools such as `pfa2pfb` or `t1binary` from the `t1binaries` package.
- The True Type font is not installed on your operating system. Install the font using the OS guidelines.
- The font is found by the JVM but not by the PDF library. Specify the font directory using the `fgl_report_configurePDFDevice` function.

Asian fonts can also cause problems in PDF format. Make sure the specified fonts contain the required characters. In your report design, for any text box using Asian characters, you may need to set the **fidelity** property to `true`.

### Tips for installing fonts

The best results are achieved if the fonts on the server where Genero Report Engine (GRE) is installed and those used in the Genero Report Designer (GRD) are the same. If these are on different systems, you may need to install fonts.

You don't need to install fonts if:

- GRE and GRD are installed on the same machine, or
- GRE and GRD are installed on different machines that have the same operating system and the same installed fonts.

**Tip:** To get information about the available fonts, use the `fontinfo` and `fontinfopdf` scripts in the `$GREDIR/bin` directory.

Even if the fonts used in the GRD and the GRE server differ, the report might still display correctly especially if it has only a few positioned items. However:

- If the server cannot find a specified font, it does not raise an error but uses a fallback font. A warning has been added to the runtime system in debugging mode (`$GREDEBUG > 0`) when a fallback font is used because a font specified in a template file cannot be found on the system.
- If no font is specified, a default font is used.

If the font is not available, you can install it with the procedure specified by the vendor.

**Tip:** On Linux® and macOS™, you can check the available fonts using the `fc-list` tool.

### Fonts available across systems

You can avoid some of the problems by using fonts that work across operating systems:

**Lucida family of Fonts**

SUN-Java contains a basic set of fonts (Serif, Sans Serif and Monospaced) in the "Lucida" family. Some distributions of the Java™ Runtime Environment do not contain all fonts, but it is legal to copy the fonts from one
distribution to another. These fonts contain a large part of the Unicode characters.

The available codes are listed in http://java.sun.com/j2se/1.5.0/docs/guide/intl/font.html#lucida.

**Liberation fonts**

Another option for free fonts is the "Liberation" fonts originally provided by Red-Hat. These fonts have the same metrics (character width) as the Microsoft™ fonts Arial, Times New Roman, and Courier, and a similar look. These fonts can be installed on both Windows® and Linux™. A description of the fonts can be found at http://en.wikipedia.org/wiki/Liberation_fonts. The fonts can be downloaded at http://fedorahosted.org/liberation-fonts/.

**Related concepts**

Distributed Mode on page 1015

The report engine can be started as a daemon to which report applications can connect to process reports. One or more engines can be started on the same machine or on different machines, hence *distributed mode*.

**Using monospace fonts in compatibility reports**

If you use monospace fonts such as Courier when in compatibility mode, you might find that the font size is too small or has other unexpected behavior.

If you use a monospaced font, a printable area, and an ASCII character layout, the font size is determined by the number of characters that are displayed in the horizontal. The more characters per line, the smaller the font size. For example, 132 characters per line results in a font size of 10, while 80 characters per line results in a font size of 12. This font size might be smaller than necessary and produce unwanted margins.

There are a few tricks you can use to alleviate this problem.

**Specify the printable area**

You can specify the printable area (the space within the margins) using the API functions:

- `fgl_report_configurePageSize(pageWidth,pageHeight)`
- `fgl_report_setPageMargins(topMargin, bottomMargin, leftMargin, rightMargin)`

**Specify the right margin**

In a 4GL report, the characters per line is specified by the optional RIGHT MARGIN parameter in the OUTPUT section. If RIGHT MARGIN is not specified, the graphical output assumes 80 characters per line.

You can override the specified RIGHT MARGIN for multiple reports using the `pageWidthInCharacters` parameter in the `fgl_report_configureCompatibilityOutput` function:

```
fgl_report_configureCompatibilityOutput(pageWidthInCharacters, fontName, fidelity, reportName, reportCategory, systemId)
```

**Place text within the printable area**

The text output is printed starting at the upper left corner of the printable area. The text can be shifted to the right and downward using the function `fgl_report_setPageMargins()`.

This solution gives finer control on the placement of text and allows margin values to be changed at runtime. However, for existing reports where the exact text placement is required, such as the placement of an address within an envelope window, you will need to adjust the placement.
Why unwanted margins appear

Depending on the dimensions of the printable area and the number of lines and columns to be printed, you might see additional margins at the right or bottom of the page. This is caused by a mismatch of the aspect ratio. For example, the "Nimbus Mono" characters have a ratio of 3/5, while "Courier" has an aspect ratio of 0.425. Fonts with lower values appear taller and more condensed than fonts with higher values.

For example, you want to print a report with 69 lines by 80 columns using the font "Nimbus Mono". The report will be printed onto a page in the format "letter" (11" x 8.5") with 0.5" margins on each side. The printable area has the dimensions 10" x 7.5". The maximum height of a line is 10"/69=10.43 point and the maximum width of a column is 7.5"/80=6.75 point. However, if you select a font size that yields a line height of 10.43 point, then the lines will only be 80*10.43*3/5=6.95" wide, leaving a gap of 0.55" on the side.

Run the "MonospaceTest" program

The MonospaceTest program can list the ratio of the different monospaced fonts installed on a system and compute which font produces the smallest margin for a particular report.

The program operates in two modes depending on the command line arguments:

**List Mode**
Invoke the MonospaceTest program without command line arguments. The program lists all monospaced fonts on the system, grouped by their specific aspect ratios. For each font group, it suggests layouts for common page formats and margin values.

**Search mode**
Invoke the MonospaceTest program with four arguments: `heightOfPrintableAreaInMM`, `widthOfPrintableAreaInMM`, `lines`, and `cols`. The program computes which font produces the smallest margins.

1. Compile the program:

```java
javac MonospaceTest.java
```

This creates the following files in the current directory:

```
$ls -l *.class -rw-r--r-- 1 alex alex 341 2010-08-17 12:24
MonospaceTest$Value.class -rw-r--r-- 1 alex alex 654 2010-08-17 12:24
MonospaceTest$MMValue.class -rw-r--r-- 1 alex alex 686 2010-08-17 12:24
MonospaceTest$InchValue.class -rw-r--r-- 1 alex alex 726 2010-08-17 12:24
MonospaceTest$Entry.class -rw-r--r-- 1 alex alex 6944 2010-08-17 12:24
MonospaceTest.class
```

2. Run the program by typing the following in the directory containing the compiled "class" files:

```java
java MonospaceTest [args]
```

MonospaceTest.java
The MonospaceTest.java program can query for the aspect ratio values of the different fonts installed on a system and perform some computations based on these values.

Run the following program:

```java
import java.awt.GraphicsEnvironment;
import java.awt.Font;
import java.awt.font.FontRenderContext;
import java.awt.font.GlyphVector;
import java.awt.geom.Rectangle2D;
import java.util.Vector;
```
import java.util.Hashtable;
import java.util.Enumeration;

/*
   Tips for using monospaced fonts for compatibility reports

   1 Specifying the printable area

   The printable area is specified using the API functions
   fgl_report_configurePageSize(pageWidth,pageHeight)
   and
   fgl_report_setPageMargins(topMargin,bottomMargin,leftMargin,rightMargin).
   The printable area is the space within the margins.

   2 Description of the general mechanism used to render compatibility reports

   2.2 Font size computation

   Given a monospaced font (e.g. Courier), a printable area (e.g.
   210mm-20mm x 297mm-20mm) and a ASCII character layout (e.g. 69 lines x 80
   characters),
   the layouter will select the largest possible font size so that no
   character will be printed outside the printable area. The font size
   is an integer value so that the font might be slightly smaller than
   actually necessary. This may produce unwanted margins as explained below.

   2.3 The need to specifying "RIGHT MARGIN" or call the API function
   fgl_report_configureCompatibilityOutput() instead

   As explained in section 2.2, one of the factors that influence the font
   size computation is the number of characters that will be displayed in the
   horizontal. A higher number of characters (e.g. 132) per line will
   cause a smaller font to be chosen (e.g. 10 point) while smaller numbers
   (e.g. 80) may yield a larger font size (e.g. 12 point).
   In a 4GL REPORT this is specified by the "RIGHT MARGIN" parameter in
   the OUTPUT section. Unfortunately this parameter is optional so
   that many existing 4GL reports are coded without specifying this
   parameter. In this case the graphical output will assume 80 characters
   per line so that the font may be too small for reports that require
   less width. For reports that print beyond the 80th column the layouter
   will render the page using the font computed for 80 characters and then
   scale the result to prevent clipping.
   Since both effects are undesirable, care should be taken to specify the
   value correctly.
   In cases where the "RIGHT MARGIN" specification would have to be
   added to a large number of REPORT objects, it is possible to call the API
   function
   fgl_report_configureCompatibilityOutput(pageWidthInCharacters,fontName,fidelity,reportName,
   reportCategory,systemId) instead.
   The parameter "pageWidthInCharacters" allows to overrides the value of
   "RIGHT MARGIN" specified in the REPORT.

   2.3 Text placement within the printable area

   The text output is printed at the upper left corner of the printable
   area so that the character at line 1, column 1 will appear in the upper
   left corner. The text can be shifted to the right and downward using the
   function fgl_report_setPageMargins() as mentioned in section 1.

   2.3.1 Handling of character margin definitions in the ASCII REPORT.
Left character margin definition "LEFT MARGIN cols" and top margin definitions "TOP MARGIN lines" in the "OUTPUT" section of the report are ignored in the graphical output. Margin values should be specified using the API function fgl_report_setPageMargins() as explained in section 1. This solution gives finer control on the placement of the text on the paper and in addition it allows changing the margin values at runtime. The downside is that for existing reports where the exact text placement is required (e.g. placement of an address within an envelope window), adjustments will be necessary. As discussed below, there are a number of other issues that make it unlikely that printing with the default settings will produce output that matches the original to the extent that individual character are placed at exactly the same locations.

Ideally, the most remote character (e.g. the character at line 69, column 80) is printed in the bottom right corner.

3 Unwanted margins

Depending on the dimensions of the printable area and the number of lines and columns to be printed one may experience additional margins at the right and/or at the bottom of the page.

3.1 Source of the problem

3.1.1 Mismatch of aspect ratio

Looking at several different monospaced fonts displaying at the same height one will find that more often than not, the character width differs between the fonts. As an example consider the font "Nimbus Mono" whose characters have a ratio of 3/5 while "Courier" has an aspect ratio of 0.425, "Courier New" a ratio of 0.53 and "Courier 10 Pitch" a ratio of 0.602.

Fonts with lower values will visually appear taller and more condensed than fonts with higher values.

Besides aesthetic aspects, the ratio has consequences for the margins that appear when the ratio of the font and the ratio of the printable area do not match well.

In order to illustrate the problem consider the hypothetical case that we wanted to print a report that has 10 lines by 10 columns using the font "Nimbus Mono" onto a page in the format "letter" (11" x 8.5") with 0.5" margins on each side. In this case the printable area has the dimensions 10" x 7.5". Choosing a font size that produces exactly 10 lines (about 72 point) we will find that there is a 1.5" margin between the rightmost column and the right edge of the printable area. The reason for this gap is, that the font "Nimbus Mono" has an aspect ratio of 3/5 so that a rectangle of 10 by 10 that is 10" high will appear 6" wide leaving a 1.5" gap to the horizontally available 7.5".

Had we chosen 8 lines by 10 columns then there would have been no gap at all. For this particular page layout and font we can find other optimal layouts for all integer solutions of the equation cols=lines*5/4.

All optimal character layouts between 1 and 100 lines are:

<table>
<thead>
<tr>
<th>Lines</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>24</td>
<td>30</td>
</tr>
<tr>
<td>28</td>
<td>35</td>
</tr>
<tr>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>36</td>
<td>45</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>44</td>
<td>55</td>
</tr>
<tr>
<td>48</td>
<td>60</td>
</tr>
</tbody>
</table>
The formula can be derived as follows:
1) \( \text{characterWidth} = \frac{7.5}{\text{cols}} \)
2) \( \text{characterHeight} = \frac{10}{\text{lines}} \)
3) \( \text{characterHeight} = \text{characterWidth} \times \frac{5}{3} \)
4) \( 3 \) in \( 2 \):
   \( \text{characterWidth} \times \frac{5}{3} = \frac{10}{\text{lines}} \)
   \( \text{characterWidth} = \frac{(10 \times 3)}{(\text{lines} \times 5)} \)
5) \( 1 \) in \( 4 \):
   \( \frac{7.5}{\text{cols}} = \frac{(10 \times 3)}{(\text{lines} \times 5)} \)
   \( \text{cols} = \frac{\text{lines} \times 5}{4} \)

Given a printable area of 11" by 8.5" without margins and a common character layout of 69 lines by 80 columns a font needs to have an aspect ratio of 0.666 to not produce any margins as the following calculation shows:
1) \( \text{characterWidth} = \frac{8.5}{80} \)
2) \( \text{characterHeight} = \frac{11}{69} \)
3) \( \text{characterHeight} = \frac{\text{characterWidth}}{\text{ratio}} \)
4) \( 3 \) in \( 2 \):
   \( \frac{\text{characterWidth}}{\text{ratio}} = \frac{11}{69} \)
5) \( 1 \) in \( 4 \):
   \( \frac{8.5}{80} \times \frac{1}{\text{ratio}} = \frac{11}{69} \)
   \( \text{ratio} = 0.666 \)

For the paper format ISO A4 the aspect ratio would have to be 0.61 as computed as follows:
1) \( \text{pageHeight} = \sqrt{\sqrt{2}} / 4 \) // A4 page height in meter
2) \( \text{pageWidth} = \frac{\text{pageHeight} \times \sqrt{2}}{4} = 1 / \left( \sqrt{\sqrt{2}} \times 4 \right) \) // width=height/\sqrt{2} for all ISO formats
3) \( \text{characterWidth} = \frac{\text{pageWidth}}{80} = 1 / \left( \sqrt{\sqrt{2}} \times 320 \right) \)
4) \( \text{characterHeight} = \frac{\text{pageHeight}}{69} = \sqrt{\sqrt{2}} / 276 \)
5) \( \text{characterHeight} = \frac{\text{characterWidth}}{\text{ratio}} \)
6) \( 5 \) in \( 4 \):
   \( \frac{\text{characterWidth}}{\text{ratio}} = \frac{\sqrt{\sqrt{2}}}{276} \)
5) \( 3 \) in \( 6 \):
   \( \frac{1}{\left( \sqrt{\sqrt{2}} \times 320 \times \text{ratio} \right)} = \frac{\sqrt{\sqrt{2}}}{276} \)
   \( \text{ratio} = \frac{1}{\left( \sqrt{\sqrt{2}} \times 276 \times 320 \right)} \)
   \( \text{ratio} = 0.61 \)

With a font that has an aspect ratio of 1/2 (0.5) such as "MS Gothic" one can produce the layout of a line printer in 8 LPI mode.
The paper format is "letter" (8.5" x 11") with 0.5" margins on all sides and the character layout is 88 lines by 132 columns.

Computing the optimal ratio for a printable area of 7.5" x 10" for a character layout of 88 lines by 132 columns:

1) \( \text{characterWidth} = \frac{8.5"-1"}{132} = \frac{7.5"}{132} \)
2) \( \text{characterHeight} = \frac{11"-1"}{88} = \frac{10"}{88} \)
3) \( \text{characterHeight} = \text{characterWidth} / \text{ratio} \)

4) 3 in 2:
   \( \frac{\text{characterWidth}}{\text{ratio}} = \frac{10"}{88} \)

5) 1 in 4:
   \( \frac{7.5"}{132}/\text{ratio} = \frac{10"}{88} \)
   \( \text{ratio} = \frac{660}{1320} \)
   \( \text{ratio} = 0.5 \)

As another example consider a font that has an aspect ratio of 3/5 (0.6) such as "Nimbus Mono". For this font, one can produce an optimal output for 88 lines by 110 columns on the same printable area as shown in the table above.

3.1.2.1 Measurements of historic line printer formats

Line printers typically printed at the following formats:

<table>
<thead>
<tr>
<th>Character layout</th>
<th>Page size</th>
<th>Font size</th>
<th>Pitch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) 66 lines by 80 characters</td>
<td>11&quot; by 8.5&quot;</td>
<td>6 LPI (12 point)</td>
<td>9.41</td>
</tr>
<tr>
<td>CPI 0.6375</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2) 88 lines by 80 characters</td>
<td>11&quot; by 8.5&quot;</td>
<td>8 LPI (9 point)</td>
<td>9.41</td>
</tr>
<tr>
<td>CPI 0.85</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3) 66 lines by 132 characters</td>
<td>11&quot; by 14&quot;</td>
<td>6 LPI (12 point)</td>
<td>9.43</td>
</tr>
<tr>
<td>CPI 0.636</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4) 88 lines by 132 characters</td>
<td>11&quot; by 14&quot;</td>
<td>8 LPI (9 point)</td>
<td>9.43</td>
</tr>
<tr>
<td>CPI 0.85</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The CPI values in the table are computed from the page measurements. It is not clear if these values were used or if the rounded value "10 pitch" was used. If we assume that the rounded values were used, then 80 characters require only 8" leaving a right margin of 0.5". The table of measurements is then:

<table>
<thead>
<tr>
<th>Character layout</th>
<th>Page size</th>
<th>Font size</th>
<th>Pitch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) 66 lines by 80 characters</td>
<td>11&quot; by 8.5&quot;</td>
<td>6 LPI (12 point)</td>
<td>10 CPI</td>
</tr>
<tr>
<td>CPI 0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2) 88 lines by 80 characters</td>
<td>11&quot; by 8.5&quot;</td>
<td>8 LPI (9 point)</td>
<td>10 CPI</td>
</tr>
<tr>
<td>CPI 0.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3) 66 lines by 132 characters</td>
<td>11&quot; by 14&quot;</td>
<td>6 LPI (12 point)</td>
<td>10 CPI</td>
</tr>
<tr>
<td>CPI 0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4) 88 lines by 132 characters</td>
<td>11&quot; by 14&quot;</td>
<td>8 LPI (9 point)</td>
<td>10 CPI</td>
</tr>
<tr>
<td>CPI 0.8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It is interesting to observe that option 1 from the table above also fitted well on A4 portrait which has the format 11.7" by 8.27" leaving a margin of 0.27" on the right and 0.7" at the bottom.

3.1.2 Margins due to integer rounding
Choosing a font with a matching aspect ratio does not guarantee small margins because the layouter only accepts integer values as a point size for font selection so that the "perfect" font size has to be rounded down to the nearest integer value.

As an example consider using the font "MS Gothic" with an aspect ratio of 0.5 which fits perfectly on the format 8.5" x 11" with 0.5" margins on all sides and a character layout 88 x 132. As shown in 3.1.1 this font produces no margins at all for this layout. However, the required font size is 8.2125 point which has to be rounded down to 8 point. Using this font produces a gap on the right side of 4.9mm and a gap at the bottom of 6.6mm.

Using the font "L M Typewriter12 Regular" which has a less optimal aspect ratio of 0.514 produces a gap at the bottom of 6.9mm at its best point size of 8.1 point. However, since the rounding difference between 8.1 and 8 is smaller than with the optimal font, the total margins after rounding are actually smaller than those using the optimal font. For this font the margins at 8 point are 1.3mm at the bottom and 8.6mm on the right.

4 The program "MonospaceTest"
The program can be used to query for the aspect ratio values of the different fonts installed on a system and it performs some computations based on the values. The program operates in two modes depending on the command line arguments as explained below:

4.1 List Mode
By invoking the program without command line arguments, it examines all monospaced fonts on the system and lists them grouped by their specific aspect ratio. For each font group it lists ideal layouts for various common page formats and margin values.

4.2 Search mode.
By invoking the program with four arguments "heightOfPrintableAreaInMM", "widthOfPrintableAreaInMM", "lines" and "cols" it will compute which font produces the smallest margins.

4.3 Compiling the program
Type "javac MonospaceTest.java" to compile the program. This will create the following files in the current directory:

```bash
$ls -l *.class
-rw-r--r-- 1 alex alex       341 2010-08-17 12:24 MonospaceTest
$Value.class
-rw-r--r-- 1 alex alex       654 2010-08-17 12:24 MonospaceTest
$MMValue.class
-rw-r--r-- 1 alex alex       686 2010-08-17 12:24 MonospaceTest
$InchValue.class
-rw-r--r-- 1 alex alex       726 2010-08-17 12:24 MonospaceTest
$Entry.class
-rw-r--r-- 1 alex alex      6944 2010-08-17 12:24 MonospaceTest.class
```

4.4 Running the program
Type "java MonospaceTest [args]" in the directory containing the compiled "class" files to run the program.

*/

class MonospaceTest
{
    final static double A4HeightInMM=Math.sqrt(Math.sqrt(2))/4*1000;
    final static double A4WidthInMM=A4HeightInMM/Math.sqrt(2);
    final static double LetterHeightInMM=11*2.54*10;
    final static double LetterWidthInMM=8.5*2.54*10;
    final static double FourteenInchInMM=14*2.54*10;
    static void usage()
    {
        }
System.err.println("Usage: java MonospaceTest|java MonospaceTest
heightOfPrintableAreaInMM widthOfPrintableAreaInMM lines cols");
System.exit(1);

public static void main(String[] args) throws Exception
{
    if(args.length==4)
    {
        double printableHeightInMM;
        double printableWidthInMM;
        int lines;
        int cols;
        printableHeightInMM=Double.parseDouble(args[0]);
        printableWidthInMM=Double.parseDouble(args[1]);
        lines=Integer.parseInt(args[2]);
        cols=Integer.parseInt(args[3]);
        findBestFont(printableWidthInMM,printableHeightInMM,cols,lines);
    }
    else
    if(args.length==0)
    {
        listAllOptions();
    }
    else
    {
        usage();
    }
}

static void findBestFont(double printableWidthInMM,double
printableHeightInMM,int cols,int lines)
{
    System.out.println("Fonts suitable for "+lines+" x "+cols+" characters on a
printable area of "+(printableHeightInMM/10)+"cm by "+(printableWidthInMM/10)+"cm (aspect ratio="+(printableWidthInMM/
printableHeightInMM)+")\n");
    GraphicsEnvironment
    ge=GraphicsEnvironment.getLocalGraphicsEnvironment();
    FontRenderContext frc=new FontRenderContext(null,false,true);
    Font[] fonts=ge.getAllFonts();
    double pageAspectRatio=printableWidthInMM/printableHeightInMM;
    boolean[] isMonospaced=new boolean[fonts.length];
    double[] gapOnTheRight=new double[fonts.length];
    double[] gapAtTheBottom=new double[fonts.length];
    double[] gapOnTheRightI=new double[fonts.length];
    double[] gapAtTheBottomI=new double[fonts.length];
    double[] fontAspectRatios=new double[fonts.length];
    double[] fractionalPointSize=new double[fonts.length];
    for(int i=0;i<fonts.length;i++)
    {
        Font f=fonts[i];
        if(isMonospaced(f,frc))
        {
            isMonospaced[i]=true;
            Rectangle2D characterRect=getCharacterDimensions(f,frc);
            fontAspectRatios[i]=getAsRatio(characterRect);
            double textHeight=lines;
            double textWidth=cols*fontAspectRatios[i];
            double textAspectRatio=textWidth/textHeight;
            if(textAspectRatio<pageAspectRatio) // Page is wider than
            text -> gap on the right
            {
                gapOnTheRight[i]=printableWidthInMM-
printableHeightInMM*textAspectRatio;
                gapAtTheBottom[i]=0;
            } else
            { The text is wider than the page. 
                gapOnTheRight[i]=0;
                gapAtTheBottom[i]=printableHeightInMM-
printableWidthInMM*textAspectRatio;
            }
        }
    }
}
double numberOfLines = printableHeightInMM / 10 / 2.54 * 72.0 / characterRect.getHeight();
    double fontSize = f.getSize2D();
    fractionalPointSize[i] = fontSize * numberOfLines / lines;
}  
else // Page is higher than text -> gap at the bottom
{
    gapOnTheRight[i] = 0;
    gapAtTheBottom[i] = printableHeightInMM - printableWidthInMM / textAspectRatio;
    double numberOfCols = printableWidthInMM / 10 / 2.54 * 72.0 / characterRect.getWidth();
    double fontSize = f.getSize2D();
    fractionalPointSize[i] = fontSize * numberOfCols / cols;
    }
    double integerPointSizeFactor = Math.floor(fractionalPointSize[i]) / fractionalPointSize[i];
    gapOnTheRightI[i] = (printableWidthInMM - gapOnTheRight[i]) * (1 - integerPointSizeFactor) + gapOnTheRight[i];
    gapAtTheBottomI[i] = (printableHeightInMM - gapAtTheBottom[i]) * (1 - integerPointSizeFactor) + gapAtTheBottom[i];
    }
    }  
for(int i = 0; i < fonts.length; i++)
{
    double smallestGapArea = Double.MAX_VALUE;
    int smallestIndex = -1;
    for(int j = 0; j < fonts.length; j++)
    {
        if(fonts[j] != null && isMonospaced[j])
        {
            double gapArea = gapOnTheRightI[j] * printableHeightInMM + gapAtTheBottomI[j] * printableWidthInMM;
            if(gapArea < smallestGapArea)
            {
                smallestIndex = j;
                smallestGapArea = gapArea;
            }
        }
    }
    if(smallestIndex >= 0)
    {
        System.out.println("Font " + fonts[smallestIndex].getFontName() + ", with a character aspect ratio of "+ fontAspectRatios[smallestIndex] + " at size "+ fractionalPointSize[smallestIndex] + " point, produces a gap "+ (gapOnTheRight[smallestIndex] > 0 ? "on the right") : "at the bottom") + "+ of "+ (gapOnTheRight[smallestIndex] + gapAtTheBottom[smallestIndex]) + "mm\n");
        System.out.println("Using an integer point size of " + (int)Math.floor(fractionalPointSize[smallestIndex]) + " will produce a gap on the right of "+ gapOnTheRightI[smallestIndex] + "mm and a gap at the bottom of "+ gapAtTheBottomI[smallestIndex] + "mm\n");
        fonts[smallestIndex] = null;
    }
}
static void listAllOptions()
{
    GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
    FontRenderContext frc = new FontRenderContext(null, false, true);
    Font[] fonts = ge.getAllFonts();
    Hashtable<Double, Entry> entries = new Hashtable<Double, Entry>();
for(int i=0;i<fonts.length;i++)
{
    Font f=fonts[i];
    if(isMonospaced(f,frc))
    {
        Double fontAspectRatio=new Double(getAspectRatio(f,frc));
        Entry e=entries.get(fontAspectRatio);
        if(e==null)
            entries.put(fontAspectRatio,new
        Entry(fontAspectRatio.doubleValue(),f));
        else
            e.getFonts().add(f);
    }
}
for (Enumeration<Entry> e = entries.elements();
    e.hasMoreElements();)
{
    Entry entry=e.nextElement();
    double fontAspectRatio=entry.getAspectRatio();
    System.out.println("Aspect ratio="+fontAspectRatio+" found in
fonts:");
    Vector<Font> fontList=entry.getFonts();
    for(int i=0;i<fontList.size();i++)
        System.out.println(fontList.get(i).getFontName());
    System.out.println("Fitting options:");
    String[] paperNames=new String[]{"letter","11\" by 14\"","A4"};
    double[] paperHeights=new double[]
    {LetterHeightInMM,FourteenInchInMM,A4HeightInMM};
    double[] paperWidths=new double[]
    {LetterWidthInMM,LetterHeightInMM,A4WidthInMM};
    Value[] margins=new Value[] {new InchValue(0),
        new InchValue(0.5),
        new InchValue(1),
        new InchValue(1.5),
        new InchValue(2),
        new MMValue(10),
        new MMValue(20),
        new MMValue(30),
        new MMValue(40),
        new MMValue(50)};
    for(int portrait=0;portrait<2;portrait++)
    {
        boolean isPortrait=portrait==0;
        for(int type=0;type<paperNames.length;type++)
        {
            for(int m=0;m<margins.length;m++)
            {
                double marginValue=margins[m].getMMValue();
                double pageWidth=(isPortrait?
                paperWidths[type]:paperHeights[type])-marginValue;
                double pageHeight=(isPortrait?
                paperHeights[type]:paperWidths[type])-marginValue;
                double pageAspectRatio=pageWidth/pageHeight;
                for(int lines=1;lines<=132;lines++)
                {
                    for(int cols=1;cols<=132;cols++)
                    {
                        double width=cols*fontAspectRatio;
                        double height=lines;
                        if(Math.abs(width/height-
pageAspectRatio)<0.0001)
                        {
System.out.println("Solution for "+paperNames[type]+(isPortrait?" portrait":" landscape")+" with margins of "+margins[m]+": lines="+lines+", cols="+cols);
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
Options for data operations in reports

Genero Studio contains several options for performing data operations such as duplication, grouping, and aggregation. Your choice is determined by your requirements, including the type of data source, the flexibility of access, and importance of efficient performance.

- Write a custom data source using GROUP BY/ORDER BY in the SQL if:
  - The report needs only aggregated data, not details.
  - Performance is important.
- Use a report with ORDER EXTERNAL BY if:
  - The report needs details.
  - You want to group by different dimensions in different reports.
  - Performance is important.
- Use sub reports if:
  - The sub report design is used stand alone or from another report.
  - You want to reuse part of a report across other reports.
  - Data can change between calling one sub report and another.
- Use a business graph in your report design if:
  - You want to display a simple table or chart.
  - The data has only one or two dimensions.
- Use a pivot table in your report design if:
  - You want to display a simple table or chart.
  - The data has many dimensions.
- Use a report schema transformation if:
  - You want multiple reports, some using detailed data from the data source, and others using only simple data.
  - You want multiple reports that group and aggregate the detailed data in different ways.
  - The data source is a fixed third-party source, but you need to group and aggregate the data in a different way than is supplied by the source.
• The data is needed more than once, but issuing two SQLs to retrieve it is not an option because the data cannot change between queries.
• Performance lags are acceptable to gain greater flexibility.

Create a report program

With Genero Report Writer, you create the code to fetch and stream the data to the Genero Report Engine and to select the report design document. The code also influences the report output, overriding default output settings or output settings specified in the report design document.

The data source defines the data to be streamed to the report engine. If your data is in a relational database that can be queried using SQL, it is recommended that you generate the data source in Genero Report Writer using the Business Application Modeler (BAM). In other cases, you may need to use existing data sources or to write your own data sources from scratch. For example, your data might exist on the Web or in memory. You can use Genero Report Writer to handle these manually-created data sources.

Once defined, a data source can be integrated into the code of your existing applications, or it can be included in a new custom reporting application. The use of the Genero Report Writer APIs allow further control over the processing of the reports.

• The runtime architecture (GRW for Genero BDL) on page 672
• BDL programs on page 672
• Modifying the output on page 700
• GRW reference for Genero BDL applications on page 719

Related concepts

Steps to a Report on page 650
To use Genero Report Writer to create a report, you need to properly prepare your environment, create the report application and report, and run and test the report.

Writing the Genero BDL report program on page 672
A Genero BDL report program retrieves the data and outputs it to a report.

Designing a Report on page 781
Design a simple or complex report using the views, menus, and tools in the Report Designer.

### The runtime architecture (GRW for Genero BDL)

The Genero BDL application is one part of the runtime architecture for Genero Report Writer for Genero Business Development Language.

![Genero Report Writer workflow diagram](image)

**Figure 325: Genero Report Writer workflow**

- The Genero Business Development Language (BDL) defines the data needed from the database
- The Genero DVM (Dynamic Virtual Machine) executes the BDL code to retrieve the data
- The Genero DVM streams data from the database to the Genero Report Engine (GRE)
- The Genero Report Designer (GRD) graphically lays out the report
- The Genero Report Engine (GRE) lays out paged streams to Genero Report Viewer (GRV)
- The Genero Report Viewer (GRV) (part of the Genero Desktop Client) displays the report on the client; alternatively, the report can be displayed as a PDF or Image, as an HTML or XML file, or output to an Excel spreadsheet.

### BDL programs

You can code a reporting application using Genero Business Development Language.

- Writing the Genero BDL report program on page 672
- Create labels: the report program (Genero BDL) on page 692
- Report Data Wizard on page 693

### Writing the Genero BDL report program

A Genero BDL report program retrieves the data and outputs it to a report.

- Overview on page 673
- Tips on page 673
- Creating a simple report on page 674
- Fetching report data on page 677
- Output options on page 680
• Generate a data schema from a Genero BDL report program on page 680
• Allowing the user to select output options on page 681
• Running a Genero ASCII report using GRW (Compatibility Report) on page 683
• Sub reports on page 684

Related concepts
Change report output options on page 701
Output options are initially set by the report design document. You can override the report settings in the report application.

Steps to a Report on page 650
To use Genero Report Writer to create a report, you need to properly prepare your environment, create the report application and report, and run and test the report.

Create labels: the report program (Genero BDL) on page 692
A common use of reports is to create a page of labels, such as address labels. Creating labels requires you to create a report design document representing a single label and a report application that prints multiple labels on a page.

Overview
Genero BDL is used to write the code for a 4gl file that retrieves the data, often from a database, and outputs it to a report.

This BDL program contains:
• The Report Driver, which specifies the data the report will include. If you are retrieving data from database tables, for example, a database cursor is defined to retrieve specific rows. The Report Driver stores the database values in program variables, and sends these - one record at a time - to the REPORT program block.
• The REPORT program block, which specifies the control blocks for a report, and sends the data to the Genero Report Engine. Variables can be defined to allow calculations on the data values, which are output along with the data values.

The Genero Report Engine uses the data and the Report Design document (4rp - created using Genero Report Designer) to process and output the report, in accordance with the reporting API functions that have been called by the BDL program.

See Steps to a Report on page 650 for a complete outline of the reporting process.

Some Tips for Legacy Report Designers provides information about the correlation between Report Designer and traditional 4GL commands in reports.

The OrderReport.4gl example program included in the GRW demo Reports:
• Retrieves the data and outputs it to the Report Engine.
• Allows the user to select the desired report and the output settings, by displaying a form (Configuration.42f) for input.
• Uses the Reporting API function fgl_report_loadCurrentSettings() to load the original settings from the 4rp file. (Additional optional functions can specify the changes to be made to those settings, based on the user input.)
• Uses the Reporting API function fgl_report_commitCurrentSettings() to commit the changes this program has made.
• The Form Design file for this form, (Configuration.4fd) is one of the files in the GRW demo project, Reports.

See the Reporting API for the specific functions that you can use to control the output of a report; some examples are in Change Report Output Options.

Tips
Tips for writing a BDL report program.

1. The Reporting API functions fgl_report_loadCurrentSettings and fgl_report_CommitCurrentSettings are required and must be called in sequence. (Calls to these functions bracket any calls to other functions that change the default output options.) These functions must be called in the program prior to executing the START REPORT
command in the Report Driver. If your program does not change the default output options, you can simply call the single function fgl_report_loadAndCommit instead.

2. Always use ORDER EXTERNAL in the REPORT program block to tell the report that the input records have already been sorted by the SELECT statement, and there is no need for the program to resort the fields. This will improve performance.

3. Restrict your use of control blocks in the REPORT program block to FIRST PAGE HEADER, BEFORE GROUP OF, AFTER GROUP OF, ON EVERY ROW, or ON LAST ROW. See The Report Program Block.

4. Prefer the use of fields over expressions to hold report values. Since expressions don't have names, it will be difficult to reference them in the Report Design Document (they could only be referenced by index).

5. Prefer calculating values in the REPORT program block over calculating in the Report Design Document, to use the power of the DVM.

6. Don't be stingy when outputting fields; this will enhance the ability to use the BDL source for multiple report formats. Define a record, orderline for example, that contains all the data fields retrieved from the database by the database cursor; use that record definition in the OUTPUT to REPORT statement:

   OUTPUT TO REPORT <reportname> (orderline.*)

Creating a simple report
This example BDL program (SimpleReport.4gl) uses data from the officestore database to create a report. There are more complex examples in the Reports project provided as a demo with Genero Report Writer.

Type Definition
This section gives the database schema used in variable definitions, and defines a User Type that consists of a single record containing all the fields from all the referenced tables.

Creating a TYPE allows the record definition to be specified only once in the program; thereafter, the name of the TYPE is used in the program wherever that record definition would be required.

```
SCHEMA officestore

TYPE ORDERTYPE RECORD
  orders    RECORD LIKE orders.*,
  account   RECORD LIKE account.*,
  country   RECORD LIKE country.*,
  lineitem  RECORD LIKE lineitem.*,
  product   RECORD LIKE product.*,
  category  RECORD LIKE category.*,
  item      RECORD LIKE item.*
END RECORD
```

MAIN program block
The MAIN program block contains the program logic that allows a user to run a report.

- defines a report handler object.
- call the mandatory BDL functions that configure the report:
  - fgl_report_loadCurrentSettings accepts the name of the Report Design document (4rp) as a parameter. The return value is a boolean indicating whether the load of the settings from the 4rp file was successful.
  - fgl_report_commitCurrentSettings returns a SaxDocumentHandler object, defined as handler.
- calls the Report Driver function, runReportFromDatabase, to run the report, passing handler.

**Note:** The example runReportFromDatabase function contains the START REPORT statement. Do not place any code between the call to fgl_report_commitCurrentSettings and the START REPORT statement that would allow the user to cancel the report.
DEFINE handler om.SaxDocumentHandler-- report handler

-- call the mandatory functions that configure the report
IF fgl_report_loadCurrentSettings("myreport.4rp") THEN -- if the file
    LET handler = fgl_report_commitCurrentSettings() -- commit the
    file -- settings
ELSE
    EXIT PROGRAM
END IF

-- run the report by calling the report driver contained
-- in your function runReportFromDatabase
IF handler IS NOT NULL THEN
    CALL runReportFromDatabase(handler)
END IF

END MAIN

Important: The libgre.42x library contains these mandatory functions and other BDL helper functions for Reports. This library must be included in any report application. List this library in the external dependencies property of any Genero Studio Project Manager application node that uses Genero Report Writer.

The Report Driver

The runReportFromDatabase function uses SQL to extract the data from the database officestore:

- handler previously created by the fgl_report_commitCurrentSettings function is passed to this function
- defines a record variable orderline using the User Type defined in MAIN
- defines handler passed as a parameter to this function
- makes a connection to the database in unique-session mode, since the program will not need to connect to other databases.
- declare a cursor for the SQL statement.
- uses the ORDER BY clause of the SQL statement defines the sort order of the data.
- These next statements are only used when the data is being retrieved from a database.
  - the BDL statement START REPORT <reportname> TO XML HANDLER must be used to instantiate the report driver, using the handler that was passed to this function; this specifies that the report data should be output in XML format.
  - the FOREACH loop opens the cursor and fetches the data from the database into the record variable, one row at a time.
  - the OUTPUT TO REPORT statement outputs each data row to the REPORT program block.
  - the FINISH REPORT terminates the BDL report process.
  - closes the SQL cursor.

FUNCTION runReportFromDatabase(handler)
DEFINE orderline ORDERTYPE, -- User Type defines record
    handler om.SaxDocumentHandler -- definition for parameter
    -- passed to this function
DATABASE "officestore" -- database connection
DECLARE c_order CURSOR FOR -- cursor declaration
SELECT orders.*, account.*, country.*, lineitem.*, product.*, category.*,
item.*
FROM orders, account, lineitem, product, category, item, country
WHERE
  orders.orderid = lineitem.orderid
AND orders.userid = account.userid
AND lineitem.itemid = item.itemid
AND item.productid = product.productid
AND product.catid = category.catid
AND country.code = orders.billcountry
ORDER BY orders.userid, orders.orderid, lineitem.linenum

START REPORT report_all_orders TO XML HANDLER handler
function
  FOREACH c_order INTO orderline.*
  OUTPUT TO REPORT report_all_orders(orderline.*)
END FOREACH
FINISH REPORT report_all_orders
CLOSE c_order
END FUNCTION

See the *Genero Business Development Language User Guide* for additional information about the use of cursors, connections, and BDL report statements.

**The REPORT program block**

This program block accepts the data from the driver, specifies the order in which the data was sorted, and outputs the data to be formatted as specified in the report design page (4rp).

The FORMAT section specifies the control blocks for a report. The use of each control break is optional, depending on the requirements of your report document. We recommend that you restrict your usage to these control blocks:

- **FIRST PAGE HEADER** - specifies the action that the runtime system takes before it begins processing the first input record.
- **BEFORE GROUP OF/AFTER GROUP OF** - specifies the action the runtime system takes before or after it processes a group of input records.
- **ON EVERY ROW** - specifies the action the runtime system takes for every input record that is passed to the report definition.
- **ON LAST ROW** - specifies the action the runtime system takes after it processes the last input record that was passed to the report definition and encounters the FINISH REPORT statement.

See BDL Reports in the *Genero Business Development Language User Guide* for a complete discussion of the BDL statements associated with a REPORT block.

**Note:** Since PAGE HEADER and PAGE TRAILER are triggered based on the line count of the BDL report, which does not correspond with the actual page breaks, their usage should be avoided. Create page headers and footers in the report design document instead.

Example REPORT block from SimpleReport.4gl:

- define variables to allow calculations on the data values; the calculations are output along with the data.
- the **ORDER EXTERNAL BY** statement informs the report that the data was retrieved, and will be output, in the specified sort order
- the FORMAT section of the report uses control blocks to set the values of the report variables used to store calculations and to send the report data and calculations to the report. Unlike a Genero report, the PRINT statement does not contain any formatting of the data or the report line (SPACES, LINE_NO, and COLUMN operators aren't used, for example.) The report format is specified in the Report Design document (4rp).
• set the variable overalltotal to zero at the control break at the beginning of the report (FIRST PAGE HEADER).
• re-set the variable ordertotal to zero each time the value of orderid changes in the data received from the report driver (BEFORE GROUP OF).
• For each row of data received from the report driver (ON EVERY ROW), these lines store some calculations in variables. The PRINT statement outputs the database data and variables to the report, in XML format.

```plaintext
REPORT report_all_orders( orderline )
DEFINE
  orderline     ORDERTYPE,
  lineitemprice LIKE lineitem.unitprice,   -- total price for item
  overalltotal  LIKE orders.totalprice,    -- accumulator for total price
  ordertotal    LIKE orders.totalprice     -- accumulator for total price
-- for report
-- for order

ORDER EXTERNAL BY orderline.orders.userid, orderline.orders.orderid,
orderline.lineitem.linenum

FORMAT
FIRST PAGE HEADER
LET overalltotal=0                      -- initialize report total
BEFORE GROUP OF orderline.orders.orderid
LET ordertotal=0                        -- initialize ordertotal for
-- each new order

-- after calculations for each data row, output the data and
-- the calculations to the Report Engine
ON EVERY ROW
LET lineitemprice = orderline.lineitem.unitprice *
  orderline.lineitem.quantity
LET overalltotal=overalltotal + lineitemprice
LET ordertotal=ordertotal + lineitemprice
PRINT orderline.*, lineitemprice, overalltotal, ordertotal

END REPORT
```

**Fetching report data**

You can obtain the data for the report in several ways.

*From a database using SQL*

You can use SQL statements to retrieve the data for the report from database tables, requiring a connection to the database and the use of an SQL cursor.

The SimpleReport.4gl program uses this technique, shown in the runReportFromDatabase function.

**Tip:** Where possible, use a single data set, gathering multiple database tables into this data set. The database can then use a single SELECT statement, which allows for better performance than running multiple queries. However, multiple data sets are necessary in some cases; for example, you need multiple independent lists of data in your reports, or your query is too complex for the database engine.

*From a data file*

You can get the report data from a data file - one created by the BDL UNLOAD statement, for example.

This example uses the OrderReport.unl file in the GRW demo Reports.

The only change to the MAIN program block would be the call to the runReportFromFile function in line 26.
--call the mandatory functions that configure the report
IF fgl_report_loadCurrentSettings("myreport.4rp") THEN -- if the file
  LET handler = fgl_report_commitCurrentSettings() -- commit the
  file -- settings
ELSE
  EXIT PROGRAM
END IF
-- run the report by calling the report driver contained in your
-- function runReportFromFile
IF handler IS NOT NULL THEN
  CALL runReportFromFile(handler)
END IF
END MAIN

The function runReportFromFile replaces the runReportFromDatabase function as the Report Driver. It uses the
unload file OrderReport.unl to provide the data for the report.

• Defines ch as a variable of type base.channel. The BDL Channel class provides read/write access to files.
• Defines the variable dataFile to specify the unload file containing the data.
• Assigns the report name to the dataFile variable.
• Creates the ch channel object.
• opens the channel to the data file OrderReport.unl in read (r) mode.
• The WHILE statement reads a line from the data file, providing the variable list enclosed in brackets. The line is
  output to the REPORT program block. The statement terminates when all the lines have been read.
• Closes the channel.

FUNCTION runReportFromFile(handler)
DEFINE
  orderline OrderType,
  handler om.SaxDocumentHandler,
  ch base.channel, -- definition of channel object
  dataFile String       -- file containing report data

LET dataFile = "./OrderReport.unl"
LET ch = base.Channel.create()
CALL ch.openFile(dataFile,"r")

START REPORT report_all_orders TO XML HANDLER handler
WHILE ch.read([orderline.*])
  OUTPUT TO REPORT report_all_orders(orderline.*)
END WHILE
FINISH REPORT report_all_orders

CALL ch.close()
END FUNCTION

From an XML file
You can get the report data from an XML file instead of getting it directly from a database.

The XML file can be a ProcessLevelData file created by a Genero report (Output to an XML File) which has a
special format, or it may be an arbitrary XML file.

From A ProcessLevelData file
This XML file has a special format, and is created using the Report Writer API function
fgl_report_createProcessLevelData File().
The MAIN block of SimpleReport.4gl would change as follows:

- Provides the name of the data file to be used for the report
- Use the BDL function `fgl_report_runReportFromProcessLevelDataFile` to run a report using a data file. The parameters are:
  - The report handler object
  - A String containing the name of the XML file that contains the data
  - This function will replay the report from the file thereby replacing the running of the report (START REPORT, OUTPUT TO REPORT, FINISH REPORT statements).

```plaintext
MAIN

  DEFINE handler om.SaxDocumentHandler, -- report handler
data_file String,              -- XML file containing data
  report_ok boolean              -- return value from function

  --call the mandatory functions that configure the report
  IF fgl_report_loadCurrentSettings("myreport.4rp") THEN -- if the file
    LET handler = fgl_report_commitCurrentSettings()     -- loaded OK
    LET data_file = "/OrderReportData.xml"
    LET report_ok =
      fgl_report_RunReportFromProcessLevelDataFile(handler, data_file)
  ELSE
    EXIT PROGRAM
  END IF

END MAIN
```

### From Other XML files

XML files that were not created by the Reporting API function `fgl_report_createProcessLevelDataFile()` can be used as the data source for a Genero report. The file `OrderData.xml` in the demo sample files of the Reports project is an example of this type of file.

The function `fgl_report_runFromXML` is used to specify the data source:

- This function sets up the report engine based on the current settings that have previously been loaded by a call to `fgl_report_loadCurrentSettings()`, and may have been modified by calls to `fgl_report_selectDevice()` or `fgl_report_selectPreview()`. The function automatically calls `fgl_report_commitCurrentSettings()`.
- This function will replay the report from the file thereby replacing the running of the report (START REPORT, OUTPUT TO REPORT, FINISH REPORT statements).

This example is from the `OrderReportXML.4gl` demo program in the Reports project:

```plaintext
MAIN

  IF NOT fgl_report_loadCurrentSettings("Table.4rp") THEN
    EXIT PROGRAM
  END IF

  IF NOT fgl_report_runFromXML("OrderData.xml") THEN
    DISPLAY "RUN FAILED"
    EXIT PROGRAM
  END IF
```
From a Web Service
Web Services can be used to fetch data for a report. BDL functions allow you to access a web service and retrieve data, storing it in BDL program variables.

See Web Services on page 1075 and the Genero Business Development Language User Guide for additional information and examples.

Output options
Various output formats are available for a report, to include SVG, PDF, and image files. The output is sent to the destination specified by the output options.

Default Output
If you don't override the default options, on report execution the user:

• Creates an SVG report
• Previews the Report in Genero Report Viewer (for desktop reporting applications) or Genero Report Viewer for HTML5 (for web-based reporting applications).

From the Genero Report Viewer, a user can then direct the output to a printer.

The Reporting API provides BDL functions to change the output in your BDL program by overriding the default options.

Output Options in the 4rp (Report Design Document)
You can define paper settings and output options for a report within the report design document itself. To do this, open a report in the Genero Report Designer, and select File > Report properties > Paper settings or File > Report properties > Output options. These values can be changed at runtime, using the reporting APIs.

Related concepts
Change report output options on page 701
Output options are initially set by the report design document. You can override the report settings in the report application.

Allowing the user to select output options on page 681
Users can choose the format to see the report output.

Configure fonts on page 657
To output well-formatted reports, verify that the necessary fonts are available.

Configure printers on page 656
To output well-formatted reports, verify that the required printer is available to your server and supported by your system.

Generate a data schema from a Genero BDL report program
After you write or modify a Genero report program, you must generate the data schema (rdd) file. This file is used by the Genero Report Designer to provide a list of data objects for use in the report design.

The data schema (rdd) file is based on the REPORT statement in your Genero report application source file (4gl). This rdd file is used in the report design document (4rd) to populate the Data View, providing details about the fields that will be streamed by the application. The schema contains the list of database columns that make up your data record, as well as grouping details.

Although the data for the report originally may have come from several different data tables, the PRINT statement in your BDL REPORT program block outputs the data as part of a single record. See the Genero Studio >> Report Writer documentation topic "Writing the BDL Program" for more information.

From the command line
Use the --buildrdd command-line option of the fglcomp tool to create a data schema (rdd file). For example:
fglcomp --build-rdd SimpleReport.4gl

The output of this command will be SimpleReport.rdd. The rdd file will be stored in the same location as the 4gl file.

**Note:** If your Genero program contains multiple 4gl files, run fglcomp against the file containing your REPORT program block.

**From Genero Studio**

Add --build-rdd to the Compiler options property for your Genero source (4gl) file to generate the rdd file automatically each time the 4gl file is compiled. Select the 4gl file listing in the application node of the Projects view to display its properties in the Properties view. The rdd file will be stored in the directory specified in the Target Directory property of the application node that contains the 4gl file.

**Related concepts**

- Compiler / Runtime configuration (Genero Installations) on page 200
  A Genero Installation contains information about the compiler and runtime version of Genero that will be used by Genero Studio.

- Adding report data (Data view) on page 796
  The Data View specifies the structure of the data record for the report.

**Allowing the user to select output options**

Users can choose the format to see the report output.

In the GRW demo program Order Report, this form allows the user to make choices regarding the output:

![Figure 326: Report demo Form](image)

**Figure 326: Report demo Form**

This code implements this functionality. The form definition file, Configuration.4fd, is included in the demo files for Reports, the GRW Demo.

The MAIN block of SimpleReport.4gl would change as follows:

- Call a function named select_output that displays the form and returns the user's choices
- Configure the report using the mandatory Reporting API functions plus these additional functions:
  - **fgl_report_selectDevice** - sets the report output device; the parameter is one of: "Printer", "PDF", "Image", "SVG".
  - **fgl_report_selectPreview** - sets the preview option; the parameter is TRUE (preview) or FALSE (no preview).

```plaintext
MAIN
DEFINE handler om.SaxDocumentHandler, -- report handler
```
data_file String, -- XML file containing data report_ok boolean, -- return value from function r_filename STRING, -- filename of desired report r_output STRING, -- output format preview INTEGER -- preview indicator

-- call the mandatory functions that configure the report using the
-- choices made by the user
CALL select_output() RETURNING r_filename, r_output, preview
IF fgl_report_loadCurrentSettings(r_filename) THEN  -- if the file loaded
OK
    CALL fgl_report_selectDevice(r_output) -- changing default
    CALL fgl_report_selectPreview(preview) -- changing default
    LET handler = fgl_report_commitCurrentSettings() -- commit changes
ELSE
    EXIT PROGRAM
END IF

-- run the report by calling the report driver contained in
-- your function runReportFromDatabase
IF handler IS NOT NULL THEN
    CALL runReportFromDatabase()
END IF

END MAIN

select_output - this new function displays a form allowing the user to specify the name of the report to be run (the name of the 4rp file), the output format, and whether a preview is preferred.

FUNCTION select_output()
DEFINE
    r_output   STRING,   -- output format option
    r_action   STRING,   -- used to set preview option
    r_filename STRING,   -- filename of Report Design document (4rp filename)
    preview    INTEGER   -- preview option TRUE/FALSE, to display
                           -- report in Report Viewer

CLOSE WINDOW SCREEN
OPTIONS INPUT WRAP
-- display form allowing user to select the 4rp filename and output options
OPEN WINDOW f_configuration WITH FORM "Configuration"
LET INT_FLAG=FALSE
LET preview = FALSE
INPUT BY NAME r_filename,r_output,r_action
ON ACTION CANCEL
    EXIT PROGRAM
ON ACTION CLOSE
    EXIT PROGRAM
END INPUT
CLOSE WINDOW f_configuration
CALL ui.interface.refresh()

-- set preview variable to match output option r_action
IF r_action IS NOT NULL THEN
    IF r_action == "preview" THEN
        LET preview = TRUE
    END IF
END IF

-- return report name (filename of Report Design Document) and output
-- option variables to be used to configure the report engine
RETURN r_filename||"4rp", r_output, preview
END FUNCTION

Creating a report with dummy data
If your report does not contain data, you receive an error message. You can use a workaround to ensure your report
always has at least one line of data and therefore avoid this error.

To add dummy data to SimpleReport.4gl, change the code as follows:

• Replace all calls to OUTPUT TO REPORT report_all_orders(orderline.*) with OUTPUT TO
  REPORT report_all_orders(1,orderline.*). In other words, pass the INTEGER constant "1" in all
  calls.

• Modify the REPORT function report_all_orders to include an additional INTEGER parameter called
  "root", and place this parameter at the first position in the ORDER EXTERNAL BY function.

The code will now look like this:

... REPORT report_all_orders(root,orderline )
  DEFINE
    root INTEGER,
    orderline OrderType,
    lineitemprice LIKE lineitem.unitprice,
    overalltotal LIKE orders.totalprice,
    usertotal LIKE orders.totalprice,
    ordertotal LIKE orders.totalprice
  ORDER EXTERNAL BY root,orderline.orders.userid,
    orderline.orders.orderid, orderline.lineitem.linenum
...

In the Data view, the root trigger is displayed as Group root. If the layout nodes for the page root are placed inside
this trigger, it is safe to run the report even if OUTPUT TO REPORT is never called.

Running a Genero ASCII report using GRW (Compatibility Report)
This type of report is referred to as a compatibility report.

If you have an existing Genero BDL program (4gl) that creates an ASCII report, you can execute the program using
the Genero Report Writer without changing the existing Report Driver or the existing Report program block, and
without creating a Report Design document (4rp).

These additional lines would be made in the MAIN program block of the 4gl file, to call the mandatory API
functions that configure the report:

DEFINE handler om.SaxDocumentHandler
IF fgl_report_loadCurrentSettings(NULL) THEN -- switch on
  Compatibility mode
  -- (run without 4rp file)
  LET handler = fgl_report_commitCurrentSettings() -- commit settings
END IF

Switching on Compatibility mode overrides the "TO ..." part of the START REPORT BDL statement.

The formatting of the report will be that specified in the REPORT program block of your BDL program.
(See Auto-formatting for an alternative to the Compatibility format.) The report engine will use the default
output settings, displaying a preview of the report in SVG format in the Report Viewer. Additional SVG
preview options are available, using the API function fgl_report_configureSVGPreview. The API function
fgl_report_configureCompatibilityOutput can optionally be used to change the default output settings.

The demo report OrderReportASCII illustrates this feature.
Auto-formatting Reports that have no 4rp (Report Design Document)
Beginning with version 2.40, auto-formatting of these reports that have no report design document (4rp) is available, as an alternative to the Compatibility format. New API calls have been added to provide generic report formatting. Currently only one design is provided, a simple list design that is compatible with the List Report template that is shipped with Genero Report Writer. This type of auto-formatting is particularly well suited to produce Excel™ output from arbitrary reports. See fgl_report_setAutoformatType and fgl_report_configureAutoformatOutput.

Report Metadata in Compatibility Reports
New API functions allow you to add report metadata to Compatibility Reports. These functions are intended to be used for compatibility reports only. See Report Metadata functions. For regular reports, the metadata can be set in corresponding properties in the Properties View of the Report Designer.

Mixing Genero ASCII Reports, GRW Reports, and Compatibility Reports
Genero ASCII (text-based) reports and GRW (graphical) reports can be run from the same Genero program without requiring any additional calls.

However, running Compatibility Reports requires that Compatibility mode be switched on. Once Compatibility mode is invoked, it stays active throughout the program, running any subsequent Genero ASCII reports using the Genero Report Writer also. If your program needs to restore text-based output for a Genero ASCII report, you must call the API function fgl_report_stopGraphicalCompatibilityMode.

Sub reports
You can include one or more reports inside another report.

With sub reports, you can maximize the reuse of reports in the Genero BDL application, reuse report designs, and break apart complex designs into smaller reusable parts.

Note: Alternatively, you can use report transformations to duplicate data. See Report schema transformations on page 775 and Options for data operations in reports on page 670.

Developing sub reports
Implementing a sub report requires you to work with both the Genero application source code and the Genero report design documents.

In the BDL code, a sub report call is detected when a START REPORT instruction is found within another report's REPORT block. The report containing the START REPORT instruction is the master report. The called report is the sub report.

In the report design document, the selected data schema includes sub report triggers.

Runtime behavior
At runtime, the reports are combined. The sub report is inserted into the master report.

The behavior at the insertion point follows the same rules that generally apply to nested containers. The master report needs to be designed with awareness of the configuration of the root container of the sub report. For example, if the sub report occupies all the space it can get (the height and width of the MiniPage both set to max), the master report must be prepared to handle that. In particular, the use of ancestors other than MiniPage should be avoided, since they will become overfull if the sub report requires more than one page of space.
**When to use sub reports**

Sub reports offer a solution for specific reporting scenarios. Before implementing a sub report solution, ensure it is the best fit for your reporting needs.

**Create a report composed of other reports**

You have two reports that you want to output as a single report. For example, you have a report that is a graph showing revenue by region, and you have a list report providing the details on the revenue by region. You can create a master report that outputs both of these reports (the graph report and the line detail report) as one report.

**Reuse a report part across multiple reports**

Consider an order confirmation report and an invoice report. The two reports could share a table containing the sales items and their descriptions (the shared part). You can put that table in a sub report. Changing the sub report then changes the two reports using the sub report.

**Create a report where a sublist has more than one sort key item**

Imagine you have a report with two loops, where each loop defines a sublist:

```blade
REPORT report(r)
...
FORMAT
  ON EVERY ROW
  PRINT r.*
#loop1
  DECLARE c1 CURSOR FOR
  SELECT * FROM t1 WHERE t1.key=r.t1key ORDER BY a,b,c
  FOREACH c1 INTO t1.*
  PRINT t1.*
END FOREACH
#loop2
  DECLARE c12 CURSOR FOR
  SELECT * FROM t2 WHERE t2.key=r.t2key ORDER BY id
  FOREACH c2 INTO t2.*
  PRINT t2.*
END FOREACH
END REPORT
```

Should the loops loop1 and loop2 be replaced by sub reports?

Loop1 specifies three sort key items in the `ORDER BY` clause. Replacing loop1 with a sub report using `ORDER EXTERNAL a, b, c` gives you the flexibility to "trigger" on a, b, and c in the design.

Replacing loop2 by a sub report would not provide any gain (assuming there is no interest to group on `id`). For this loop, you might prefer to see the whole design.

**Do NOT use sub reports to process nested sub lists**

Genero reports support arbitrarily complex models using `PRINT` statements inside iterator statements (`WHILE, FOR, FOR EACH`) and conditional statements (`IF, CASE`). It is not necessary to use sub reports for this purpose. When sub reports are used for this purpose to produce complex reports (such as an invoice report), the report becomes scattered across numerous BDL REPORTs and report design (4rp) files, obscuring the overall structure. The impossibility to see the design as a whole must be taken into consideration when using this strategy.

**Coding for sub reports**

A sub report is a report that is started from within another report using a `START REPORT` statement.

The `START REPORT` statement for the sub report takes no arguments. Do not add `TO XML HANDLER` for the sub report call.
Sub reports can be called from any control block wherever printing in loops is allowed: ON EVERY ROW, BEFORE GROUP, AFTER GROUP, or ON LAST ROW.

The START REPORT must be in the calling report block. The START REPORT statement cannot be placed inside a function called from the report.

The START REPORT, OUTPUT TO REPORT, and FINISH REPORT statements must be in the same control block. The OUTPUT TO REPORT and FINISH REPORT statements can be placed inside a function or functions called from the report, however those functions must be in the same control block as the START REPORT statement.

You can have multiple calls to sub reports from the same control block. However, the calls cannot be nested.

Valid:

```plaintext
REPORT master_report(...) 
...
FORMAT 
  BEFORE GROUP ... 
  ...
START REPORT detail_report_1 
FOREACH ...
  OUTPUT TO REPORT detail_report_1(...) 
END FOREACH 
FINISH REPORT detail_report_1 
START REPORT detail_report_2 
FOREACH ...
  OUTPUT TO REPORT detail_report_2(...) 
END FOREACH 
FINISH REPORT detail_report_2 
...
END REPORT
```

Not valid:

```plaintext
REPORT master_report(...) 
...
FORMAT 
  BEFORE GROUP ... 
  ...
START REPORT detail_report_1 
START REPORT detail_report_2 -- Not allowed until after 
  -- FINISH REPORT detail_report_1 
...
END REPORT
```

Sub reports can be nested to arbitrary depth, but recursion is not allowed.

```plaintext
REPORT master_report(...) 
...
FORMAT 
  ON EVERY ROW 
  ...
START REPORT detail_report 
FOREACH ...
  OUTPUT TO REPORT detail_report(...) 
END FOREACH 
FINISH REPORT detail_report 
END REPORT
REPORT detail_report(...) 
...
FORMAT 
  ON EVERY ROW
```
START REPORT other_detail_report -- Since recursion is not allowed, we are not allowed to call "master-report" or "detail_report" from here.

FOREACH ...
    OUTPUT TO REPORT other_detail_report(...
END FOREACH
FINISH REPORT other_detail_report
END REPORT

Detected sub report calls are shown in the report designer. They appear as nodes in the data view, triggers the structure view, and images in the document view.

Example 1

In this example, the master report (master_report) calls two distinct sub reports (acct_order_subreport_1 and acct_order_subreport_2). The invocation of a sub report requires the START REPORT, OUTPUT TO REPORT and FINISH REPORT from within the master report.

FUNCTION run_master_report(handler)
DEFINE handler om.SaxDocumentHandler
START REPORT master_report TO XML HANDLER HANDLER
OUTPUT TO REPORT master_report()
FINISH REPORT master_report
END FUNCTION

REPORT master_report()

-- Report definition for master report
DEFINE data acct_order_data

FORMAT
ON EVERY ROW
DISPLAY "SUBREPORT 1"
DECLARE cur_srl1 CURSOR FOR
SELECT
    account.*,
    country.*,
    item.*,
    lineitem.*,
    orders.*,
    product.*,
    category.*
FROM
    account,
    country,
    item,
    lineitem,
    orders,
    product,
    category
WHERE
    country.code = account.country AND
    product.productid = item.productid AND
    item.itemid = lineitem.itemid AND
    orders.orderid = lineitem.orderid AND
    account.userid = orders.userid AND

FUNCTION run_subreport_1(handler)

FUNCTION run_subreport_2(handler)

FUNCTION run_subreport_3(handler)
country.code = orders.billcountry AND
  country.code = orders.shipcountry AND
  product.catid = category.catid
ORDER BY
  account.userid,
  orders.orderid
START REPORT acct_order_subreport_1
FOREACH cur_sr1 INTO data.*
  OUTPUT TO REPORT acct_order_subreport_1(data.*)
END FOREACH
FINISH REPORT acct_order_subreport_1
CLOSE cur_sr1

DISPLAY "NOW FOR SUBREPORT 2"
DECLARE cur_sr2 CURSOR FOR
SELECT
  account.*,
  country.*,
  item.*,
  lineitem.*,
  orders.*,
  product.*,
  category.*
FROM
  account,
  country,
  item,
  lineitem,
  orders,
  product,
  category
WHERE
  country.code = account.country AND
  product.productid = item.productid AND
  item.itemid = lineitem.itemid AND
  orders.orderid = lineitem.orderid AND
  account.userid = orders.userid AND
  country.code = orders.billcountry AND
  country.code = orders.shipcountry AND
  product.catid = category.catid
ORDER BY
  account.userid,
  orders.orderid
START REPORT acct_order_subreport_2
FOREACH cur_sr2 INTO data.*
  OUTPUT TO REPORT acct_order_subreport_2(data.*)
END FOREACH
FINISH REPORT acct_order_subreport_2
CLOSE cur_sr2

END REPORT

REPORT acct_order_subreport_1(data)
DEFINE
data acct_order_data,
  item_total DECIMAL(10,2),
  order_total DECIMAL(10,2),
  account_total DECIMAL(10,2),
  report_total DECIMAL(10,2)
ORDER EXTERNAL BY
  data.account.userid,
  data.orders.orderid

FORMAT
FIRST PAGE HEADER
  LET report_total = 0
BEFORE GROUP OF data.account.userid
  LET account_total = 0
BEFORE GROUP OF data.orders.orderid
  LET order_total = 0
ON EVERY ROW
  LET item_total = data.lineitem.quantity*data.lineitem.unitprice
  LET order_total = order_total + item_total
  LET account_total = account_total + item_total
  LET report_total = report_total + item_total
PRINTX data.*, item_total, order_total, account_total,
  report_total
END REPORT

REPORT acct_order_subreport_2(data)
  DEFINE
    data acct_order_data,
    item_total DECIMAL(10,2),
    order_total DECIMAL(10,2),
    account_total DECIMAL(10,2),
    report_total DECIMAL(10,2)
ORDER EXTERNAL BY
    data.account.userid,
    data.orders.orderid

FORMAT
  FIRST PAGE HEADER
    LET report_total = 0
BEFORE GROUP OF data.account.userid
    LET account_total = 0
BEFORE GROUP OF data.orders.orderid
    LET order_total = 0
ON EVERY ROW
    LET item_total = data.lineitem.quantity*data.lineitem.unitprice
    LET order_total = order_total + item_total
    LET account_total = account_total + item_total
    LET report_total = report_total + item_total
PRINTX data.*, item_total, order_total, account_total,
  report_total
END REPORT

Example 2
In this example, the master report (master_report) calls the same sub report block (sub_report) for both sub reports.

SCHEMA grwdemo

TYPE storeorder_data RECORD
  orders RECORD LIKE orders.*,
  items RECORD LIKE items.*,
  customer RECORD LIKE customer.*
END RECORD

FUNCTION run_report(hand)
  DEFINE hand om.SaxDocumentHandler
  START REPORT master_report TO XML HANDLER hand
  OUTPUT TO REPORT master_report()
  FINISH REPORT master_report
END FUNCTION
REPORT master_report()
-- This is the master report.
-- PREPARE and DECLARE the cursor to define data to be read in.
-- For each of the sub-reports, include a START REPORT|FOREACH|FINISH REPORT

DEFINE data storeorder_data

FORMAT
ON EVERY ROW
PREPARE s1 FROM
    "SELECT orders.*, items.*, customer.*
    FROM orders, items, customer
WHERE
    orders.order_num = items.order_num AND
    customer.store_num = orders.store_num
ORDER BY
    customer.store_num, orders.order_num"

DECLARE cur CURSOR FOR s1

-- Notice that both reports are using the sub_report() function.
-- They don't have to, but they can;
-- It means that the data being streamed by the sub_report() function
-- is valid for both reports.

# First sub-report
START REPORT sub_report
FOREACH cur INTO data.*
    OUTPUT TO REPORT sub_report(data.*)
END FOREACH
FINISH REPORT sub_report

# Second sub-report
START REPORT sub_report
FOREACH cur INTO data.*
    OUTPUT TO REPORT sub_report(data.*)
END FOREACH
FINISH REPORT sub_report

END REPORT

REPORT sub_report(data)
-- This function is taking in each row, streaming to report, and managing
-- the store_total number needed by the reports.

DEFINE
    data storeorder_data,
    store_total DECIMAL

ORDER EXTERNAL BY
    data.customer.store_num,
    data.orders.order_num

FORMAT

BEFORE GROUP OF data.customer.store_num
    LET store_total = 0

ON EVERY ROW
    LET store_total = store_total + (data.items.price * 
data.items.quantity)
    PRINT data.*, store_total
Creating a master report design
Create a new empty report design document (4 rp). Select the master report data source as the data schema. Detected sub report calls are shown in the report designer. They appear as nodes in the data view, triggers the structure view, and images in the document view.

The Data View
In the data view, the invocation of the sub report is displayed at its position in the application source. While the execution of the sub report requires three invocations (START REPORT, OUTPUT TO REPORT and FINISH REPORT), only the location of the START REPORT item is shown in the data view.

The Report Structure view
A sub report trigger is created in the document report structure. Set the URL property of the sub report trigger to a report design that matches the sub report schema.

Document view (Central Work Area)
Sub reports display as images in the document view in the Central Work Area. Double-click the image to open the design file for editing.
Create labels: the report program (Genero BDL)
A common use of reports is to create a page of labels, such as address labels. Creating labels requires you to create a report design document representing a single label and a report application that prints multiple labels on a page.

Create the report design document
The report design document (.4rp) contains the design for a single label. You can put anything you wish on the label, as long as it fits onto the page.

See Design labels on page 805 for details on creating the single-label report design document (.4rp).

Update your report application
Once the label is defined in a report design document, you update your report application so that it outputs the data needed for a label within **ON EVERY ROW** in the **REPORT** block, and it tells the report engine to output multiple labels onto a single physical page.

- In the **FORMAT** section of the **REPORT** program block, output the data needed for the label **ON EVERY ROW**.
- Configure the report engine for labels using Reporting API functions.
  - *fgl_report_selectLogicalPageMapping* allows you to specify that this report will create a page of labels.
  - *fgl_report_setPaperMargins* sets the margins (top, bottom, left, right).
  - *fgl_report_configureLabelOutput* configures the physical layout of the label page using these parameters:
    - paper Width
    - paper Height
    - label Width (set to null)
    - label Height (set to null)
    - number of labels per row
    - number of labels per column

Example
This example uses the OrderLabels.4rp report design document from the Reports GRW demo.

In the **SimpleReport BDL program**, the MAIN program block that configures the report engine is changed as follows:

- Loads OrderLabels.4rp as the **Report Design Document**
- The function *fgl_report_selectLogicalPageMapping* allows you to specify the format of the page as "labels"
- The function *fgl_report_setPaperMargins* sets margins using these parameters:
  1. top margin
  2. bottom margin
  3. left margin
  4. right margin
- Function *fgl_report_configureLabelOutput* - configures the physical layout of a label page using these parameters:
  1. paper Width
  2. paper Height
  3. label Width
  4. label Height
  5. number of labels per row
  6. number of labels per column

```plaintext
-- configure report engine; the functions prefixed fgl that are called here
-- are part of the Reporting API
```
IF fgl_report_loadCurrentSettings("OrderLabels.4rp") THEN -- load the
labels
-- design document (4rp
file)
-- special settings for labels
CALL fgl_report_selectLogicalPageMapping("labels")
CALL fgl_report_setPaperMargins("5mm","5mm","4mm","4mm")
CALL fgl_report_configureLabelOutput("a4width","a4length",null,null,2,6)
END IF
LET handler = fgl_report_commitCurrentSettings() -- commit changes

The rest of the SimpleReport program would remain unchanged. This example would print two labels across and six labels down per page.

Figure 329: Output of labels on a physical page

Report Data Wizard
The Report Data Wizard provides an easy-to-use interface for creating the data source.

- Using the Report Data Wizard on page 693
- Example Wizard code on page 698
- Use the example code on page 699

Related concepts
Writing the Genero BDL report program on page 672
A Genero BDL report program retrieves the data and outputs it to a report.

Steps to a Report on page 650
To use Genero Report Writer to create a report, you need to properly prepare your environment, create the report application and report, and run and test the report.

Using the Report Data Wizard
In the Report Data Wizard, you can specify the details of your SQL query, including the database and tables, the joins between the tables, the table fields, and the sort order.

A Wizard is now available to generate BDL code for your Genero report program (.4gl). Select File>>New,
Reports, Report from Database (.4gl) to display the dialogs that guide you in creating the SQL query statement that will extract the report data from your database.

Table Selection Page
Select the database and tables for the query.
Define Joins Page

The New Query dialog is used to add the joins between the tables to be used in the SQL query. The information is pre-filled from the database meta-data in the database meta-schema file (4db). You can select different columns in the tables, or select a different join method.

Figure 330: New Query wizard, Table Section page

Press Next to continue.
Figure 331: New Query wizard, Define joins page

- **Left column**: The table on the left side of the join operator.
- **Right column**: The table on the right side of the join operator.
- **Operator**: the method to be used to join the tables for the result set.

Press **Next** to Continue.

This table illustrates the effect of the operators on the query result set:
# Table 190: Effect of the join operators on the query result set

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example WHERE clause generated for BDL code</th>
</tr>
</thead>
<tbody>
<tr>
<td>INNER JOIN</td>
<td>Result set will contain columns from only the rows in the two tables where the specified column value matches.</td>
<td>WHERE orders.userid = account.userid</td>
</tr>
<tr>
<td>LEFT OUTER JOIN</td>
<td>Result set will contain columns from all the rows in the left table, joined with columns from the rows in the right table where the specified column value matches. Where there is no matching value, the columns from the right table in the result set will contain nulls.</td>
<td>WHERE (orders.userid = account.userid OR account.userid IS NULL)</td>
</tr>
<tr>
<td>RIGHT OUTER JOIN</td>
<td>Result set will contain columns from all the rows in the right table, joined with columns from the rows in the left table in which the specified column value matches. Where there is no matching value, the columns from the left table in the result set will contain nulls.</td>
<td>WHERE (orders.userid = account.userid OR orders.userid IS NULL)</td>
</tr>
<tr>
<td>FULL OUTER JOIN</td>
<td>Result set will contain columns from all the rows in the left table, and columns from all the rows in the right table, joined on the specified column value. Where there is no match in one of the tables, the columns from that table will contain nulls.</td>
<td>WHERE (orders.userid = account.userid OR account.userid IS NULL OR orders.userid IS NULL)</td>
</tr>
</tbody>
</table>

## Result Set Page

Select the database table fields for the result set of the SQL query. The data in the result set is passed by the BDL REPORT function to the Genero Report Engine and the Report Definition file (.4rp).
Figure 332: New Query wizard, Result Set page

Press **Next** to Continue.

**Order By Page**

Select the column(s) by which the result set should be sorted.
Press Finish to generate the code and close the Report Data Wizard.

The Wizard creates a BDL file (.4gl) containing the generated code, which you can save as part of your application files.

**Example Wizard code**

This example illustrates code generated by the Report Data Wizard.

This is the code that was generated by the example covered in Using the Report Data Wizard on page 693

```sql
SCHEMA officestore

TYPE officestore_data RECORD
    account RECORD
        userid LIKE account.userid,
        email LIKE account.email
    END RECORD,
    orders RECORD
        orderid LIKE orders.orderid,
        orderdate LIKE orders.orderdate,
        shipaddr1 LIKE orders.shipaddr1,
        shipaddr2 LIKE orders.shipaddr2,
        shipcity LIKE orders.shipcity,
        shipstate LIKE orders.shipstate,
        shipzip LIKE orders.shipzip
    END RECORD
END RECORD

FUNCTION run_officestore_to_handler(handler)
```
DEFINE
data officestore_data,
handler om.SaxDocumentHandler
DECLARE cur CURSOR FOR
SELECT
    account.userid,
    account.email,
    orders.orderid,
    orders.orderdate,
    orders.shipaddr1,
    orders.shipaddr2,
    orders.shipcity,
    orders.shipstate,
    orders.shipzip
FROM
    account,
    orders
WHERE
    orders.userid = account.userid
ORDER BY
    account.userid
START REPORT officestore_report TO XML HANDLER handler
FOREACH cur INTO data.*
    OUTPUT TO REPORT officestore_report(data. *)
END FOREACH
FINISH REPORT officestore_report
CLOSE cur
END FUNCTION

REPORT officestore_report(data)
DEFINE
data officestore_data
ORDER EXTERNAL BY
    data.account.userid

FORMAT
    ON EVERY ROW
        PRINT data. *
END REPORT

Note:

- Lines 3-17 define a user-type named office_data, which is a RECORD consisting of sub-records for the fields that were selected in the Result Set page from the tables selected in the Table Selection page.
- Lines 19-48 define the run_officestore_to_handler function.
  - Lines 23-40 declare a cursor for the SQL SELECT statement to retrieve the data. The table join and the sort order were specified in the Define Joins and Order By pages.
  - Lines 42-46 fetch the report data and send it to the REPORT function.
- Lines 50-59 define the REPORT function, which outputs the data to the Genero Report Engine.

See The BDL File for additional information about the BDL functions and statements used in conjunction with reports.

Use the example code
Use the example code that was generated by the Report Data Wizard.

About this task
You can create a report application using the generated BDL file, or the file can be incorporated as part of a larger application. We recommend that you use a different BDL file to contain the additional code that is required, rather
than making changes in the generated file. This allows you to re-execute the Report Data Wizard without risking the loss of your changes.

   a) Create the rdd file that is used to populate the Data View of the Report Design Document, by executing this from the command line:

   ```
   fglcomp --build-rdd <generated_file>.4gl
   ```

   b) Add the resulting `<sourcefile>.rdd` file to your application files.

2. Design the Report.
   Specify the rdd file on the Data View page. When your design is complete, save the report definition file (4rp) as part of your application. See Designing a Report for additional information about the design process.

3. Add Additional BDL Code to the program
   a) To configure the report, call the mandatory report API functions, which identify the report definition file and create the XML handler for the report:
      - `fgl_report_loadCurrentSettings` accepts the name of the Report Design document (4rp) as a parameter. The return value is a Boolean indicating whether the load of the settings from the 4rp file was successful.

      Additional report API functions can be called to change the default output options, if needed.
   b) To connect to the Database, add a `CONNECT TO` or `DATABASE` statement to the program.
   c) Call the `run_<databasename>_to_handler` function (in the file generated by the Report Data Wizard) to execute the report.

Example MAIN function

This simple function illustrates the addition of BDL code:

```main
DEFINE handler om.SaxDocumentHandler -- report handler
DATABASE officestore -- connect to the database
-- call the mandatory functions that configure the report
IF fgl_report_loadCurrentSettings("myreport.4rp") THEN -- if file loaded
   OK
   LET handler = fgl_report_commitCurrentSettings() -- commit the
   -- file settings
ELSE
   EXIT PROGRAM
END IF
-- run the report by calling the report driver contained in your
-- generated function
IF handler IS NOT NULL THEN
   CALL run_officestore_to_handler(handler)
END IF
END MAIN
```

Modifying the output

In addition to creating the data model, a BDL report application can control the report output.

- Change report output options on page 701
- Using localized strings on page 716
**Change report output options**
Output options are initially set by the report design document. You can override the report settings in the report application.

The table below outlines the available output options and their capabilities. Choose the appropriate option for your requirements.

**Table 191: Report output options**

<table>
<thead>
<tr>
<th>Format</th>
<th>Archiving</th>
<th>Integrated dialog printing</th>
<th>Silent printing</th>
<th>Viewing (streaming preview)</th>
<th>Editing</th>
<th>Computing</th>
<th>Mailing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Printer</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDF</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Postscript</td>
<td>Yes</td>
<td></td>
<td>With O.S. printing command line tools, e.g. <code>lp</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVG</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Excel</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTF</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Browser</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each report design document can include settings regarding the output format, rendering settings, page range to include in the output, and image settings. These values are set when a report is first created, and can be altered by the report designer. See [The Report output](#) on page 779 for details about setting these values within the report design document (.4rp).

- Change paper settings and output format on page 702
- Print a report on page 703
  - Send Output Directly to Printer
  - Set Page Margins
  - Set Print Quality
  - Print Double-sided
  - Select a Paper Tray
- Support for custom paper sources on page 705
- Preview a report on page 707
- Create an SVG report on page 708
- Create an HTML report on page 708
- Create an Excel spreadsheet report on page 708
- Create a Microsoft RTF report on page 713
- Create an XML data file on page 714
- Create multi-page ISO reports on page 714
- Optimize the rendering process on page 715

**Related concepts**
Designing a Report on page 781
Design a simple or complex report using the views, menus, and tools in the Report Designer.

**Configure fonts** on page 657
To output well-formatted reports, verify that the necessary fonts are available.

**Configure printers** on page 656
To output well-formatted reports, verify that the required printer is available to your server and supported by your system.

**Change paper settings and output format**
There are several places where you can specify the paper settings and output format for a report.

**Set default paper settings for all new reports**
The default settings for a report design document (.4rp) are set when you create a new report. They are based on the default settings for all new reports, as set by the local installation of Genero Studio.

To set the default paper settings for all new reports, go to **Tools > Preferences, Report Writer, Paper Settings**.

**Set the paper settings for a report design document**
For an existing report, you can change the paper settings and the output format in the report design document (.4rp).
Go to **File > Report properties**, and:

- Select **Paper Settings...** to update the paper settings for the report.

**Set the paper settings and output format in the report application**
In your report application, you can override the paper settings and output format defined in the report design document.
For a Genero BDL report program (.4gl), using the functions provided by the Reporting API allows you to change:

- Page settings
- Output device options
- Printer settings

When adding these functions to your Genero BDL report application, place the functions between the mandatory configuration functions of fgl_report_loadCurrentSettings() and fgl_report_commitCurrentSettings():

```plaintext
IF fgl_report_loadCurrentSettings(r_filename) THEN -- load the 4rp file
  mandatory
  CALL fgl_report_selectPreview(prev) -- functions to change
  the output
  here
    LET handler = fgl_report_commitCurrentSettings() -- commit
    changes mandatory
END IF
```

**Related concepts**
- **Specify the paper settings of a report** on page 807
- Paper settings set the paper orientation, the page size format, and the report margins for a report.

**Reporting API Functions** on page 719
- Genero Report Writer provides a variety of reporting API functions for use with Genero BDL applications.

**Mandatory functions** on page 720
These functions are required in the Report Driver section of the Genero BDL file associated with a Genero BDL reporting application.

**Print a report**
Set up and configure output to a printer from your reporting application. The report is printed silently, without prompting the user.

- **Send output directly to the printer** - you can send the output to a printer without previewing by calling the `fgl_report_selectDevice` function as part of your report configuration function with the parameter "Printer":

  ```
  CALL fgl_report_selectDevice("Printer")
  ```

**Note:** For other output formats, the parameter can be one of: "Printer", "PDF", "Image", "SVG", "HTML", "XLS" or "XLSX" (output to an Excel™ spreadsheet), "RTF" (output in Microsoft™ RTF format)

- **Set page margins** - the logical margins of the report page are read from the 4rp file (Report Design document.) The `fgl_report_setPageMargins` function overrides those margins. The syntax is:

  ```
  CALL fgl_report_setPageMargins (topMargin String, bottomMargin String, leftMargin String, rightMargin String)
  ```

  For example:

  ```
  CALL fgl_report_setPageMargins(
    ".05cm", # topMargin String,
    ".05cm", # bottomMargin String,
    ".05cm", # leftMargin String,
    ".05cm" # rightMargin String
  )
  ```

- **Set Print Quality** - the `fgl_report_setPrinterPrintQuality` function is used to control the quality used by the printer. By default this is not set. Setting this option reduces the set of usable printers to those matching it.

  One of these values are passed to the function as a parameter: draft|high|normal. The value is passed as a String, for example:

  ```
  CALL fgl_report_setPrinterPrintQuality("high")
  ```

- **Print Double-sided** - the function `fgl_report_setPrinterSides` is used to specify whether the report pages should be one-sided or two-sided. By default this is not set. Setting this option reduces the set of usable printers to those matching it.

  One of these values are passed to the function as a parameter. The value is passed as a String:

  - **one-sided**: Each consecutive print-stream page is printed on the same side of consecutive media sheets.
  - **two-sided-short-edge**: Consecutive pairs of print stream pages are printed on the front and back sides of consecutive media sheets. The orientation of the pages is correct as if the pages were to be bound along the short edge of the paper.
  - **two-sided-long-edge**: Consecutive pairs of print stream pages are printed on the front and back sides of consecutive media sheets. The orientation of the pages is correct as if the pages were to be bound along the long edge of the paper.

    ```
    CALL fgl_report_setPrinterSides("two-sided-long-edge")
    ```

- **Select a Paper Tray** - function `fgl_report_setPrinterMediaTray` controls what tray of the printer to use. This function and the functions `fgl_report_setPrintermediaTray` and `fgl_report_setPrintermediaName` are mutually exclusive. Setting this option reduces the set of usable printers to those matching it.
One of these values are passed to the function as a parameter: bottom|envelope|large-capacity|main|manual|middle|side|top. The value is passed as a String.

```c
CALL fgl_report_setPrinterMediaTray("top")
```

**Printing without a driver using PJL**

Use the Printer Job Language (PJL) to print documents without the need to install drivers.

Printer Job Language (PJL) is a method for switching printer languages and for reading printer status. PJL allows you to configure your printer for server-side silent printing on network Postscript printers.

**Advantages of PJL printing**

- Does not require driver installation.
- Full streaming available.
- Better support for configuration properties.
- Full support for printer-specific properties.
- The software and the printer communicate directly with each other, which makes debugging easier.

**Potential issues with PJL printing**

PJL printing bypasses the system spooler and occurs synchronously with the printer. Therefore, long print jobs can block the production process on printers with limited memory. You can avoid this in interactive applications by executing the printing job in a separate process or thread.

In case of printer failure, such as a paper jam, the application must manage reprinting. You can cause the printer byte stream to be written to a different destination by using the `fgl_report_setDestinationURL` function, and the file in the specified destination is completed even in case of error. This file contains all the PJL commands so that the job can be restarted by opening a client socket connection at port 9100 to the printer and then sent the contents of this file.

**Printer support**

The printer must be reachable over the network at TCP port 9100, and support both the PJL and the Postscript language. The printer capabilities must be listed using the PrinterInfo utility, which can also be used to query a specific printer for a list of its PJL environment variables and permissible values.

**PJL printer name syntax**

PJL printers are identified by their network addresses. A PJL printer is identified by a printer name having the following syntax: `pjl:password:host`, where `password` is optional.

Examples without password:

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>pjl:FrontDesk</td>
</tr>
<tr>
<td>pjl:10.0.1.052</td>
</tr>
</tbody>
</table>

Example with password:

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>pjl:1234:FrontDesk</td>
</tr>
</tbody>
</table>

**Configuring and selecting a PJL Printer**

Configure the variables for the PJL printer using the `fgl_report_setPrinterPJLVariables` function.

**Important:** String values must be specified using double quotes, for example, "quoted string".
Select the PJL printer using the `fgl_report_setPrinterName` function.

**Important:** Unlike with driver-based printers, you must specify the printer name, because PJL printers are not discovered automatically.

For example:

```
DEFINE pjlSet om.SaxAttributes
LET pjlSet=om.SaxAttributes.create()
CALL pjlSet.addAttribute("BINDING","SHORTEDGE")
CALL pjlSet.addAttribute("OUTBIN","UPPER")

CALL fgl_report_setPrinterPJLVariables(pjlSet)

CALL fgl_report_setPrinterName("pjl:10.0.1.052")
```

**Support for custom paper sources**

On Windows® systems the reporting API function `fgl_report_setSVGPaperSource()` can be used to identify a custom paper source for your report.

The `fgl_report_setSVGPaperSource()` must be used in conjunction with these functions:

- `fgl_report_setSVGPrinterName(printerName)` to specify the printer
- `fgl_report_configureSVGPreview("PrintOnNamedPrinter")` to bypass the preview and print silently on the named printer

**The printerinfo Script**

On Windows® systems, the command-line executable `$GDCDIR/bin/printerinfo.exe` is provided to list the supported paper sources that can be changed programmatically using this function.

**Note:** This `printerinfo` executable is part of the Genero Desktop Client, so it is in the GDC directory. This is a separate file to the server-side `printerinfo` executable in the GRE directory.

The output of the script lists the device-specific source names in the left column, with the general constant in brackets on the right. Either can be used to identify the paper source in calls to `fgl_report_setSVGPaperSource()`.

This excerpt lists seven paper sources for the printer "Balzac":

```
```

```
Figure 334: printerinfo output

Example:

```c
CALL fgl_report_setSVGPrinterName("Balzac")
CALL fgl_report_setSVGPaperSource("Tray 2")
CALL fgl_report_configureSVGPreview("PrintOnNamedPrinter")
```

This excerpt of the printerinfo output is for a printer that has 27 paper sources. Since there are only 15 predefined constants for the paper sources, most of the paper sources are identified by integer constants, which can be used in calls to `fgl_report_setSVGPaperSource()` Using the integer constant can prevent encoding issues when the device-specific name contains non-ASCII characters, such as "Sélection automatique".
A report can be previewed in a desktop application (normally using SVG format) or in a browser.

To preview a report in the Genero Desktop Client (GDC), you normally use the integrated "SVG" format as the output device. To preview a report in the Genero Browser Client (GBC), you normally use Genero Report Viewer for HTML5 by specifying "Browser" as the output device. If the appropriate software is installed on the client, you can preview reports in other output formats using the `fgl_report_selectPreview` function. See Table 192: Previewing reports on page 707 for details.

Table 192: Previewing reports

<table>
<thead>
<tr>
<th>Front end</th>
<th>SVG</th>
<th>Browser</th>
<th>PDF</th>
<th>Excel</th>
<th>RTF</th>
<th>OORTF</th>
<th>HTML</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDC</td>
<td>Integrated</td>
<td>Not possible</td>
<td>With Acrobat® Reader (or other PDF reader)</td>
<td>With Microsoft™ Excel</td>
<td>With Microsoft™ Word</td>
<td>With Open Office (or any free Office)</td>
<td>With any browser</td>
<td>Not possible</td>
</tr>
<tr>
<td>GBC</td>
<td>Not possible</td>
<td>Integrated</td>
<td>With Acrobat® Reader (or other PDF reader)</td>
<td>With Microsoft™ Excel</td>
<td>With Microsoft™ Word</td>
<td>With Open Office (or any free Office)</td>
<td>With any browser</td>
<td>Not possible</td>
</tr>
</tbody>
</table>

Related concepts

Create an SVG report on page 708
The SVG report format uses the integrated Genero Report Viewer to display a report.

View report in a browser on page 713
You can view a report in a browser with Genero Report Viewer for HTML5. Specific reporting API functions must be included in your program to display to this client.

Create an SVG report
The SVG report format uses the integrated Genero Report Viewer to display a report.

You can use the API function `fgl_report_selectDevice("SVG")` to output the data in the SVG Format.

The following API functions provide more SVG options:

- `fgl_report_configureSVGDevice` - configures the SVG output.
- `fgl_report_configureSVGPreview` - configures SVG preview options.
- `fgl_report_setSVGCopies` - specifies the number of copies to be printed
- `fgl_report_setSVGCompression` - specifies whether the output should be compressed.
- `fgl_report_setSVGOrientationRequested` - specifies whether the report is printed in portrait or landscape.
- `fgl_report_setSVGPageRange` - selects which pages should be printed
- `fgl_report_setSVGPrinterName` - selects a specific printer by name
- `fgl_report_setSVGPaperSource` - selects the input source of the printer
- `fgl_report_setSVGPaginationRequested` - specifies whether the report is printed in portrait or landscape.
- `fgl_report_setSVGPrinterName` - selects a specific printer by name
- `fgl_report_setSVGSheetCollate` - controls the collation of multiple copies

Create an HTML report
Set up and configure output to HTML from your reporting application.

HTML output can be selected by passing the string "HTML" to the function `fgl_report_selectDevice`. You can use this function to output any existing report (including ASCII) as an HTML document.

Graphical elements such as bar codes are implemented by rendering them as images. These images are included in the document. An option to generate the images on disk and to use external URLs instead is provided by the function `fgl_report_configureHTMLDevice`.

When elements overlay each other (or text strings intersect), the default behavior is to create one image from the two elements and to include the image. An exception to this occurs when the overlaid element is an image: the "background" image will be removed from the resulting document.

There is an option in the function `fgl_report_configureHTMLDevice` to ignore the horizontal or vertical alignment in the document. The default is FALSE for HTML document output.

Create an Excel™ spreadsheet report
Set up and configure output to Microsoft® Excel™ from your reporting application. This option is useful if you want to perform custom calculations in Excel™.

Note: Excel™ reports, as output from Genero Report Writer, do not print at high quality. If you require high-fidelity printing, use one of the other output options, such as SVG.

The report output can be sent to an Excel™ spreadsheet by passing the string "XLS" or "XLSX" to the function `fgl_report_selectDevice`. First, you should configure the report output using the `fgl_report_configureXLSDevice` or `fgl_report_configureXLSXDevice` functions.

This code fragment illustrates the functions enabling Excel output:

```plaintext
... 
CALL fgl_report_configureXLSXDevice ( 
    NULL,  #fromPage INTEGER, 
    NULL,  #toPage INTEGER, 
    NULL,  #removeWhitespace INTEGER, 
    NULL,  #ignoreRowAlignment INTEGER, 
    NULL,  #ignoreColumnAlignment INTEGER, 
    NULL,  #removeBackgroundImages INTEGER, 
    TRUE ) #mergePages BOOLEAN 
CALL fgl_report_selectDevice("XLSX")
```
Excel™ report output

The output is in the specified Excel™ format, with the XLS format limited to 65,536 rows. In contrast, for the XLSX format the document is generated with constant memory consumption, so that very large documents can be produced without exhausting heap space.

Note: The XLSX format can only be opened with newer versions of Microsoft™ Excel™ (2007 or later). A backward compatibility pack can be downloaded from Microsoft™; however, the XLSX format is then subject to the 65,536 limit of the earlier versions.

Any existing report, including ASCII, can be output to the spreadsheet.

The layout of the cells (size, font color, etc.) can be predictably controlled from the Report Designer. The goal is to put the report layout into the cells of the spreadsheet efficiently. There is an option in the functions fgl_report_configureXLSDevice and fgl_report_configureXLSXDevice to ignore the vertical or horizontal alignment. The default is TRUE for Excel™ document output, because the emphasis is on the ability to compute the values in the cells, rather than on the appearance.

When items in the report design overlap, by default the placement but not the alignment is preserved in the spreadsheet. To make the necessary decisions, Genero Studio marks the column and row boundaries internally with tabs. If two consecutive tabs have no element that bounds on them, the column or row collapses. The implication of this is:

• White space (empty columns, empty rows) is eliminated when possible.
• Elements that overlap may be placed in the same column, but they maintain their relative placement (above/below).
• Contiguous items are never placed in the same column, but maintain their relative placement and alignment.

The values in the cells are generated for the report, not the Excel™ formulas.

The following report elements are specially rendered in Excel™:

• Graphical elements such as bar codes are rendered as images.
• Business graphs are rendered as tables.
• For pivot tables, only the fact rows are shipped. The pivot table is therefore rendered as a flat table structure without being disrupted by group totals. You can then use the data to create an Excel™ pivot table with business charts and row-spanning formulas.

Output a single-sheet document to Excel

By default, when you send report data to an Excel™ spreadsheet, each page produces a separate Excel™ sheet. You can merge the pages into a single sheet, but you must then configure the report to avoid problems with alignment and calculation.

If the page breaks are not desirable, you can change the page size to a larger custom value. However, creating a huge page can adversely affect memory reclamation and performance. It is often better to use a standard page size to merge the pages into a single-sheet result.

Note: If you specify "XLS" in the call to fgl_report_selectDevice, and the resulting sheet has more than 65536 rows, the exceeding rows will spill over into extra sheets. If you specify "XLSX" as the Excel™ format, the size of the sheet is not limited by available memory.

In the fgl_report_configureXLSDevice or fgl_report_configureXLSXDevice method, use the mergepages parameter to write all data to a single sheet:

```plaintext
CALL fgl_report_configureXLSXDevice (NULL, #fromPage INTEGER, NULL, #toPage INTEGER, NULL, #removeWhitespace INTEGER, NULL, #ignoreRowAlignment INTEGER,
```
Fixing issues with column alignment

When you merge sheets, elements such as page headers and group totals might cause problems with layout and calculation. For example, Figure 336: Single Page Excel -- Unaligned on page 710 shows how a page header can disrupt the columns.

To avoid this problem, remove all page footers and turn all page headers to first page headers. You can remove these elements at runtime by using the RTL Runtime class method `producingExcelOutput()`:

- In all Page Footers, and in the First Page Header, set the Visibility Condition property to:

  ```java
  !Runtime.producingExcelOutput()
  ```

Figure 336: Single Page Excel™ -- Unaligned

To avoid this problem, remove all page footers and turn all page headers to first page headers. You can remove these elements at runtime by using the RTL Runtime class method `producingExcelOutput()`:

- In all Page Footers, and in the First Page Header, set the **Visibility Condition** property to:
In the Any Page Header, set the **Section** property to:

```
Runtime.producingExcelOutput()?Port.FirstPageHeader:Port.AnyPageHeader
```

When you merge sheets, the page headers and footers now line up correctly, as in **Figure 337: Single Page Excel -- Aligned** on page 711.

![Table Image]

**Figure 337: Single Page Excel™ -- Aligned**

**Group totals**

Group totals can also disrupt the column alignment. For example, **Figure 338: Report Design Document -- With Group Total** on page 712 shows a report design document where the "Total" is created from two separate elements: a Word Box and a Decimal Format Box.
Figure 338: Report Design Document -- With Group Total

For the first page of the report, the product ID is listed in column A and the product name is listed in column B. However, for the last page of the report, the overall total is printed in column B and the product name is moved to the right into column C, as in Figure 339: Single Page Excel -- With Group Total on page 712

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FU-004</td>
<td>Grandfather clock</td>
</tr>
<tr>
<td>2</td>
<td>FU-008</td>
<td>Office chair</td>
</tr>
<tr>
<td>3</td>
<td>FU-011</td>
<td>Table lamp</td>
</tr>
<tr>
<td>4</td>
<td>SU-001</td>
<td>Blue and green pens</td>
</tr>
<tr>
<td>5</td>
<td>SU-002</td>
<td>Blue and green pens</td>
</tr>
<tr>
<td>52</td>
<td>EN-004</td>
<td>Chess game</td>
</tr>
<tr>
<td>53</td>
<td>FU-011</td>
<td>Table lamp</td>
</tr>
<tr>
<td>54</td>
<td>TR-012</td>
<td>T-shirt</td>
</tr>
<tr>
<td>55</td>
<td>FU-008</td>
<td>Office chair</td>
</tr>
<tr>
<td>56</td>
<td>SU-009</td>
<td>Organizer book</td>
</tr>
<tr>
<td>57 Total:</td>
<td></td>
<td>8,291.20</td>
</tr>
</tbody>
</table>

Figure 339: Single Page Excel™ -- With Group Total

Note: This problem occurs because of the way the report engine renders a page into Excel output. The elements of the original output with floating point coordinates are placed into cells of the spreadsheet in a process called “tabulation”. The edge of an element that appears above the edge of another element in the pixel exact output (for example, SVG) will end up in a row with a lower index than the other element. The edge of an element that appears to the right of the edge of another element will be placed in a column with a higher index than the other element. Tabulation happens for each page as it is streamed, and so the last page might be tabulated in a different way to the first page.

To avoid this problem, edit the report design document using one of the following strategies:

• Force a page break by adding a vertical box before the total and setting the Y-size of the box to “rest”. The page break moves the total line to a separate page where it is tabulated apart from the main table data.
• Change the size and position of the total so it does not require new cells. In this case, move the decimal format box to the right so it is directly beneath the product name.

Note: You can also increase the page length of the report so it fits on one page. This strategy is not recommended because it can cause performance and memory problems.
Create a Microsoft™ RTF report
Set up and configure output to Rich Text Format (RTF) from your reporting application. This option is useful if you want to edit your report after output.

You can use the API function `fgl_report_selectDevice("RTF")` to output the data in the RTF Format. Configure the output using the function `fgl_report_configureRTFDevice`.

In order to prevent exhaustion of main memory when processing large documents, the processor can be instructed to swap parts of the document to a temporary disk file when the document size exceeds a threshold. Use the function `fgl_report_setRTFMemoryThreshold` to specify the threshold.

View report in a browser
You can view a report in a browser with Genero Report Viewer for HTML5. Specific reporting API functions must be included in your program to display to this client.

When you start the demo program `OrderReport.4gl`, the application gives the user the option to output the report data to the Genero Report Viewer. If the Genero Studio configuration is set to use the Web configuration, this option displays the report to a browser, using Genero Report Viewer for HTML5.

To do this, the device must be set to "Browser".

```
CALL fgl_report_selectDevice("Browser");
```

When the device is set to Browser, some additional APIs must be called. You must set the browser directory and URL, and you must set the font directory and URL. For example:

```
LET uuid=security.RandomGenerator.CreateUUIDString()
CALL fgl_report_setBrowserDocumentDirectory(fgl_getenv("GRE_PRIVATE_DIR")||"/"||uuid)
CALL fgl_report_setBrowserFontDirectory(fgl_getenv("GRE_PRIVATE_DIR"))
CALL fgl_report_setBrowserDocumentDirectoryURL(fgl_getenv("GRE_PRIVATE_URL_PREFIX")||"/"||uuid)
CALL fgl_report_setBrowserFontDirectoryURL(fgl_getenv("GRE_PRIVATE_URL_PREFIX"))

IF preview THEN
  CALL ui.Interface.frontCall( "standard", "launchurl",
  [fgl_getenv("GRE_REPORT_VIEWER_URL_PREFIX")||"/viewer.html?reportId="||uuid||"&privateUrlPrefix="||
  fgl_getenv("GRE_PRIVATE_URL_PREFIX")], [] )
END IF
```

Note: To use the `viewer.html` file residing on the FGL/GAS Server, use the `GRE_REPORT_VIEWER_URL_PREFIX` variable. To use the `viewer.html` file residing on the GRE Server side, use the `GRE_REPORT_VIEWER_R_URL_PREFIX` variable and copy the `GRE/viewer` directory into the GRE document root directory.

The report files
When you select Browser as the output for your report, the report files are written to the document directory.

If you are using a private directory (`GRE_PRIVATE_DIR` and `GRE_PRIVATE_URL_PREFIX`), only the current session will be able to view the report, and the report will be deleted when the session ends.

If you are using a public directory (`GRE_PUBLIC_DIR` and `GRE_PUBLIC_URL_PREFIX`), the report can be shared and bookmarked.

Warning: The environment variables `GRE_PRIVATE_DIR`, `GRE_PRIVATE_URL_PREFIX`, `GRE_PUBLIC_DIR`, and `GRE_PUBLIC_URL_PREFIX` are not documented. You should never explicitly set these variables.
Create an XML data file
Set up and configure output to XML from your reporting application.

The demo program OrderReport.4gl gives the user the option to output the report data to a file (OrderReportData.xml) in XML format.

The Reporting API function fgl_report_createProcessLevelDataFile() instructs the Genero Report Engine to execute the report, outputting the data specified in the PRINT function of the REPORT program block to an XML file rather than to a Report Design Document. The output file is stored in your current project. The name of the data file is passed to the function as a String.

Changing the SimpleReport.4gl example program to use this technique, the MAIN program block becomes:

```
MAIN
    DEFINE handler om.SaxDocumentHandler, -- return value
dataFile STRING -- name of XML file to be created

    --configure the report engine to output the XML file
    LET dataFile = ".\OrderReportData.xml"
    LET handler = fgl_report_createProcessLevelDataFile(dataFile)

    -- run the report, creating the XML file
    CALL runReportFromDatabase(handler)
END MAIN
```

Create multi-page ISO reports
For reports printed on ISO and JIS-sized pages, you can configure the output to print several logical pages per physical page.

Use these Reporting API functions to change the output options:

- **fgl_report_selectLogicalPageMapping** - use the parameter multipage to change the logical page mapping, allowing the printing of multiple pages per physical page.
- **fgl_report_configureMultipageOutput** - to configure the required number of pages. The parameters are:
  1. pageExponent String - Specifies the number of pages to print. The actual number of pages is calculated by multiplying the page exponent number that you specify by 2. In the example there will be 2 * 2 = 4 pages per physical page.
  2. isoNumber Integer - Specifies the ISO number. This parameter is optional (indicated by passing a null value). If no value is specified, the page size of the logical page is taken from the 4rp file (if specified). In the example, 4 refers to ISOA4.
  3. portrait boolean - TRUE specifies that the page is in portrait orientation; FALSE= landscape. This parameter is optional (indicated by passing a null value). If no value is specified, the orientation of the logical page is taken from the 4rp file.

Example function:

```
FUNCTION configure_report()
    DEFINE handler om.SaxDocumentHandler           -- configure report engine

    IF fgl_report_loadCurrentSettings(NULL) THEN   -- there is no 4rp file
        -- change default output options
        CALL fgl_report_selectLogicalPageMapping("multipage")
        CALL fgl_report_configureMultipageOutput(2,4,TRUE)
        LET handler = fgl_report_commitCurrentSettings() -- commit settings
    ELSE
        EXIT PROGRAM
    END IF
END FUNCTION
```
Related concepts
Reporting API Functions on page 719
Genero Report Writer provides a variety of reporting API functions for use with Genero BDL applications.

Optimize the rendering process
You can optimize the rendering process by setting rendering hints.
Use the fgl_report_setRenderingHints function to set the following values.

Optimizing GRV embedded font rendering
You can use the grvRenderToBitmap rendering hint to facilitate the optimization of your report printing. If grvRenderToBitmap is TRUE, GRV renders print pages as bitmaps, which can improve speed on some printers. If grvRenderToBitmap is FALSE, GRV renders print pages using shape drawing commands. The default value is false.

Optimizing for local fonts
You can set your printer to use local fonts instead of embedded fonts (call fgl_report_configureSVGDevice with embedFonts=FALSE). In this case, the font renderer is dependent on the system, and therefore different systems produce different sized output for identical input strings and font parameters. You can use rendering hints to improve the fidelity and the printing speed:

- svgTextLengthAdjust - Adjusts the glyph width and spacing. The value can be Spacing, SpacingAndGlyphs, or Deactivated. By default, the SVG source documents contain the expected length for each string and adjust the glyph width and spacing.
- grvUseEmbeddedFontMetrics - If TRUE, embeds the font metrics of the original server side font in the document and then positions each character individually by using the individual advance values from the metrics table. The default value is false.

Optimizing for postscript
If the font is not in the standard 16 Postscript fonts, the text is rendered as outlines, which can be very slow on some printers. For these printers, you can optimize the postscript printing by setting psOptimize to TRUE.
You can also shrink the stream size by setting psOptimizeUseDictionary to TRUE. On some printers, this improves the printer speed.

Note: In some cases, setting psOptimizeUseDictionary to TRUE will worsen the printer speed instead of improving it, so you must test the options to get the best result.

These hints are applicable for the server-side silent printing to PJL printers and Postscript files. They are also available for driver-based printers where the PrinterInfo utility lists the printer as an IPP printer and the document as application/postscript; class="java.io.InputStream.

Example
This sample code sets the svgTextLengthAdjust and grvUseEmbeddedFontMetrics hints:

```c
DEFINE renderingHints om.SaxAttributes
LET renderingHints=om.SaxAttributes.create()
CALL renderingHints.addAttribute("svgTextLengthAdjust","SpacingAndGlyphs")
CALL renderingHints.addAttribute("grvUseEmbeddedFontMetrics","true")
CALL fgl_report_setRenderingHints(renderingHints)
```

Related concepts
Specifying classic or default style on page 832
The business graphs use a "flat design" by default, but you can change this to use the more 3D "classic" design. You can also choose between three different color schemes.

**Using localized strings**
Use localized strings to change the text at runtime depending on the territory or site where the application is running.

- **Localized Strings**
  - The Source String File
  - Using the translate function
  - Entries in fglprofile
  - Finding the Translated String
  - Change localization settings at runtime on page 717

**Localized strings**
Localized strings can be used to implement internationalization in your application or to use site-specific text (for example, when business terms change based on territory).

The **Localized Strings** feature allows:

- the **captions** (titles of report objects) to be customized.
- The translation of the **Text** property to be customized.

You could have multiple external string files, each containing the translated string in a different language. The DVM (runtime system) searches the files in order to assign text in the report definition. The localized text replaces the captions and other specified text at runtime, choosing the correct string file based on entries in the file `fglprofile`.

**The Source String file**
A text file, with a `.str` extension, defines the string to be displayed for the report object captions or the specified Word Box or WordWrap Box text. You can check the **Text** property of the report object to determine the exact string to be replaced.

Syntax:

```
"identifier" = "string"
```

where:

- `identifier` is the value of the **Text** property of the caption or label that you want to localize
- `string` is the text that you want to be displayed instead

The backslash `\` is accepted as the escape character, to define non-printable characters.

Example file `captions_en.str`:

```
"UserID" = "Client ID"
"Shipcity" = "City"
```

At runtime, the text 'UserID' or 'Shipcity' displays in the report as "Client ID" and "City".

This source string file should be added to the Genero Studio Project Manager application node for your report program, so it can be compiled when the application is built. The compiled file (`.42s`) is not linked into the Genero executable.

**Note:** Although you can call the API function `fgl_report_setCallbackLocalization()` in your 4GL source code to specify that the caption of the report object be used, we recommend using localized strings.
**Using the translate function**

Entries in the source string file can also be used by the `translate` PXML function to replace text in an expression that sets a property value. For example, if you set the text of a Word Box to this value:

```
"hello "+"world".translate()
```

and you have this entry in the `str` file:

```
"world" = "universe"
```

then the final text displayed on your WordBox is “hello universe”.

**Entries in fglprofile**

You can specify the compiled string file, or a list of these files, with entries in an `fglprofile` configuration file. For example:

```
fglrun.localization.warnKeyNotFound= false     -- don't display warnings on errors
fglrun.localization.file.count = 1             -- number of string files
fglrun.localization.file.1.name = "captions_en.42s" -- name of the string file
```

Use the FGLPROFILE environment variable to specify the path to your `fglprofile` configuration file.

See the FGLPROFILE topics in the *Genero Business Development Language User Guide* for complete information about the `fglprofile` file. See the Localized Strings topics in the *Genero Business Development Language User Guide* for complete information and examples.

**Finding the translated string**

The search behavior that Report Writer uses to find a string in a compiled string file (`42s`) is similar to that of Genero BDL. Once the compiled string files are located, Report Writer uses the first occurrence of the string that it finds.

The `42s` files are sought in this order:

1. Any localization (`42s`) files specified in a BDL configuration file. First, the current directory, and then all directories in DBPATH or FGLRESOURCEPATH, are searched for the specified `42s` files. The precedence of BDL configuration files is:
   a. the BDL configuration file, `<FGLDIR>/etc/fglprofile`
   b. the BDL configuration file specified by the FGLPROFILE environment variable, if set.
   c. any BDL configuration file having the same name as the report program, and stored in the directory `<FGLDIR>/defaults`
2. Any localization (`42s`) file that has the same name as the report definition file, minus the `4rp` extension. First, the current directory, and then all directories listed in the DBPATH or FGLRESOURCEPATH environment variable, are searched for the `<report-name>.42s` file.
3. Any localization (`42s`) file that has the same name as the report application, minus the `42r` extension. First, the current directory, and then all directories in DBPATH or FGLRESOURCEPATH are searched for the `<application-name>.42s` file.

**Change localization settings at runtime**

Use the `fgl_report_configureLocalization` function to change the directory where the program seeks the translation files and to change the formatting pattern for numerical data at runtime.

**Change the language directory**

Change the language directory in which the translation files (`42s`) are sought with the `fgl_report_configureLocalization` function. The names of the translation files (`captions.42s` and
mappings.42s) remain constant, regardless of what language is selected. It is the directory that is changed by the API call.

```bash
$echo $FGLPROFILE
fglprofile
$cat $FGLPROFILE
fglrun.localization.file.count = 2
fglrun.localization.file.1.name = "captions.42s"
fglrun.localization.file.2.name = "mappings.42s"

$find translation_files
translation_files
translation_files/en
translation_files/en/captions.str
translation_files/en/captions.42s
translation_files/fr
translation_files/fr/captions.str
translation_files/fr/captions.42s
translation_files/common
translation_files/common/mappings.str
translation_files/common/mappings.42s

$cat test.4gl
MAIN
... #select French translation
  LET handler=configureReport("translation_files/fr:translation_files/
common");
  START REPORT invoice TO XML HANDLER handler
... END MAIN
... FUNCTION configureReport(resourcePath)
...  LET handler=fgl_report_loadCurrentSettings("localizable_invoice.4rp")
...  CALL fgl_report_configureLocalization(NULL,resourcePath,NULL,NULL)
...  RETURN fgl_report_commitCurrentSettings()
END FUNCTION
...
```

**Change the formatting pattern for numerical data**

Change the formatting pattern for numerical data at runtime with the `fgl_report_configureLocalization` function.

```bash
$echo $DBFORMAT
$:,:.
$cat test.4gl
MAIN
... #select German number formatting
  LET handler=configureReport(":,:EUR");
  START REPORT invoice TO XML HANDLER handler
... END MAIN
... FUNCTION configureReport(numberFormat)
...  LET handler=fgl_report_loadCurrentSettings("invoice.4rp")
...```
CALL fgl_report_configureLocalization(NULL,NULL,numberFormat,NULL)
...
RETURN fgl_report_commitCurrentSettings()
END FUNCTION
...

GRW reference for Genero BDL applications

Reference information for Genero BDL applications.

- Reporting API Functions on page 719
- Pivot table library on page 771

Reporting API Functions

Genero Report Writer provides a variety of reporting API functions for use with Genero BDL applications.

- General
- Mandatory report functions
  - fgl_report_loadCurrentSettings()
  - fgl_report_commitCurrentSettings()
  - fgl_report_loadAndCommit()
- Usage
- Additional Functions to Change Default Report and Output options
  - Data
  - Form
  - Page
  - Output
  - Printer
- Additional functions to set Report Metadata for compatibility reports
- Additional functions to introspect reports at runtime (librdd)

Overview

The Genero Report Writer reporting API for Genero BDL relies upon several files.

1. The `libgre.42m` file contains the Genero BDL functions that handle Genero Report Writer output and other features. There are mandatory functions that must be called in your Genero BDL report application, and there are optional functions that allow you to change the output format, output features and printer features.

2. These files contain some functions that are used internally by Genero Report Writer:
   - CaptionCustom.42m
   - CompatCustom.42m
   - EncodingCustom.42m
   - isotools.42m

The compiled versions of these files are provided in `<GREDIR>\lib`.

**Important:** These compiled files are linked in the `libgre.42x` library, which must be listed in the external dependencies property of any Genero Studio Project Manager application node that uses Genero Report Writer. This file should be listed by name only, without a path. Click the ... button in the Value column of the Properties View to open the Edit List window for that property.

Error Handling

In case of an error, functions from this library will write error messages to stdout using the DISPLAY statement. The error condition is indicated by a return value.
Verbosity level

Depending on the value of the environment variable "DEBUGLEVEL" some of the functions issue warnings and other useful debugging information. Currently any value greater than zero will cause debug output. Debug information is also output using the DISPLAY statement.

Mandatory functions

These functions are required in the Report Driver section of the Genero BDL file associated with a Genero BDL reporting application.

See Usage: load and commit on page 722.

Table 193: Mandatory functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fgl_report_loadCurrentSettings</strong> (reportFileName STRING) RETURNING ok INTEGER</td>
<td>Loads the report definition and configures the in-memory settings accordingly. <strong>Note:</strong> The fgl_report_loadAndCommit function may be used in place of the other mandatory functions if no changes to the default settings are required.</td>
</tr>
<tr>
<td><strong>fgl_report_commitCurrentSettings</strong> () RETURNING driver om.SaxDocumentHandler</td>
<td>Configures the Genero Report Engine based on the initial report definition and any subsequent function calls. <strong>Note:</strong> The fgl_report_loadAndCommit function may be used in place of the other mandatory functions if no changes to the default settings are required.</td>
</tr>
<tr>
<td><strong>fgl_report_loadAndCommit</strong> (reportFileName STRING) RETURNING handler om.SaxDocumentHandler</td>
<td>Function that loads a 4rp file and configures the report engine to execute the report.</td>
</tr>
</tbody>
</table>

### fgl_report_loadCurrentSettings

Loads the report definition and configures the in-memory settings accordingly.

### Syntax

```c
fgl_report_loadCurrentSettings (reportFileName STRING) RETURNING ok INTEGER
```

1. **reportFileName** - Name of the 4rp report file to process (extract the settings information). The settings are made using the Configuration menu of the Report Design page. If a relative path is specified, then it is first converted to an absolute path against the current working directory, then against FGLRESOURCEPATH, and finally against DBPATH.

   The value denotes a path on the machine where the Genero Report Engine is running. In the case of distributed processing, this may be a different machine than the machine running the dynamic virtual machine (DVM). When running a legacy Genero BDL report using Report Writer, the **reportFileName** should be NULL.

   This indicates the ASCII output from the BDL file is to be written to the "SVG" device in preview mode.

2. **ok** - Returns TRUE if no error occurred.

### Important: If your files are loaded asynchronously, a successful call to the fgl_report_loadCurrentSettings() function does not guarantee that the file has loaded successfully. If your system is configured to load files asynchronously, it is recommended that you call
fgl_report_getErrorStatus() after every call to START REPORT, OUTPUT TO REPORT, and FINISH REPORT.

Usage
This function loads the specified 4rp file and configures the current in-memory settings accordingly. Calling this function is a required prerequisite to calling fgl_report_commitCurrentSettings().

The call sequence is the following:

```plaintext
CALL fgl_report_loadCurrentSettings(filename)
--- Optional functions to change default settings
CALL fgl_report_commitCurrentSettings()
```

See Usage.

fgl_report_commitCurrentSettings
Configures the Genero Report Engine based on the initial report definition and any subsequent function calls.

Syntax

```plaintext
fgl_report_commitCurrentSettings ()
RETURNING driver om.SaxDocumentHandler
```

1. `driver` - the created XMLhandler.

Usage
This function configures the report engine based on the current settings in the 4rp file that have previously been loaded by a call to fgl_report_loadCurrentSettings(), and have possibly been modified.

It is mandatory to call fgl_report_commitCurrentSettings to set up the report engine, unless there are no changes to the default settings; in that case, fgl_report_loadAndCommit can be substituted for both of the functions.

See Usage: load and commit.

fgl_report_loadAndCommit
Function that loads a 4rp file and configures the report engine to execute the report.

Syntax

```plaintext
fgl_report_loadAndCommit  (reportFileName STRING)
RETURNING handler om.SaxDocumentHandler
```

1. `reportFileName` - Name of the 4rp report file to process (extract the settings information). The settings are made using the Configuration menu of the Report Design page. If a relative path is specified, then it is first converted to an absolute path against the current working directory, then against FGLRESOURCEPATH, and finally against DBPATH.

The value denotes a path on the machine where the Genero Report Engine is running. In the case of distributed processing, this may be a different machine than the machine running the dynamic virtual machine (DVM). When running a legacy Genero BDL report using Report Writer, the `reportFileName` should be NULL.

This indicates the ASCII output from the BDL file is to be written to the "SVG" device in preview mode.

2. `handler` - Null if an error occurred.

Usage
Use this function when no change of the default settings of the specified report is required.
This function is a convenience function, replacing the need to call both a load function and a commit function.

```
CALL fgl_report_loadCurrentSettings(reportFileName)
CALL fgl_report_commitCurrentSettings()
```

See **Usage: load and commit**.

**Usage: load and commit**

The functions `fgl_report_loadCurrentSettings` and `fgl_report_commitCurrentSettings` work together to configure the report.

The call sequence is:

```
IF fgl_report_loadCurrentSettings(reportfilename) THEN -- function returns 
  TRUE -- if settings loaded OK
  -- optional functions to change settings would go here
  LET rephandler = fgl_report_commitCurrentSettings() -- function returns 
  the
      -- SAXDocumentHandler
END IF
```

These functions must appear in the Report Driver section of the BDL file associated with the report. If no settings are to be changed, the function `fgl_report_loadAndCommit` may be substituted for these two functions.

See **Writing the Genero BDL report program** on page 672 for complete examples.

**Functions to change reporting options**

Use these functions to change the default settings for a report, and to configure output and printers. The functions are provided as part of the `libgre.42x` library.

**Table 194: Environment Configuration functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fgl_report_setEnvironment</code></td>
<td>Specifies variable values in the private environment of the report.</td>
</tr>
</tbody>
</table>
### Table 195: Data functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fgl_report_createProcessLevelDataFile</code>&lt;br&gt;<code>dataFileName STRING</code>&lt;br&gt;<code>RETURNING driver om.SaxDocumentHandler</code></td>
<td>Configures the report execution to output an XML datafile.</td>
</tr>
<tr>
<td><code>fgl_report_runFromXML</code>&lt;br&gt;<code>dataFileName STRING</code>&lt;br&gt;<code>RETURNING ok INTEGER</code></td>
<td>Replays the report from an XML file.</td>
</tr>
<tr>
<td><code>fgl_report_runReportFromProcessLevelDataFile</code>&lt;br&gt;<code>out om.SaxDocumentHandler,&lt;br&gt;fileName STRING</code>&lt;br&gt;<code>RETURNING ok INTEGER</code></td>
<td>Runs a report from a process level file.</td>
</tr>
<tr>
<td><code>fgl_report_setProcessLevelDataFile</code>&lt;br&gt;<code>dataFileName STRING</code></td>
<td>Configures the report execution to output an XML datafile in addition to the regular processing.</td>
</tr>
</tbody>
</table>

### Table 196: Form functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fgl_report_findResourcePath</code>&lt;br&gt;<code>reportName STRING</code></td>
<td>Returns the path to a resource searching first the working directory, then FGLRESOURCEPATH, and finally DBPATH. This function is deprecated.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>fgl_report_configureLabelOutput</code> (paperWidth STRING, paperHeight STRING, labelWidth STRING, labelHeight STRING, labelsPerRow INTEGER, labelsPerColumn INTEGER)</td>
<td>Configures the physical layout of a label page.</td>
</tr>
<tr>
<td><code>fgl_report_configureMultipageOutput</code> (pageExponent STRING, isoNumber INTEGER, portrait BOOLEAN)</td>
<td>Configure the multipage output for ISO or JIS formats.</td>
</tr>
<tr>
<td><code>fgl_report_configurePageSize</code> (pageWidth STRING, pageHeight STRING)</td>
<td>Set values for the page height and page width for Genero BDL ASCII reports being run in graphical mode.</td>
</tr>
<tr>
<td><code>fgl_report_selectLogicalPageMapping</code> (mapping STRING)</td>
<td>Configures the mapping of logical pages to physical pages.</td>
</tr>
<tr>
<td><code>fgl_report_setPageMargins</code> (topMargin STRING, bottomMargin STRING, leftMargin STRING, rightMargin STRING)</td>
<td>Configure the logical margins of a report.</td>
</tr>
<tr>
<td><code>fgl_report_setPageSwappingThreshold</code> (value INTEGER)</td>
<td>Sets the threshold for page-to-disk swapping.</td>
</tr>
<tr>
<td><code>fgl_report_setPaperMargins</code> (topMargin STRING, bottomMargin STRING, leftMargin STRING, rightMargin STRING)</td>
<td>Configure the physical margins of a report.</td>
</tr>
</tbody>
</table>
Table 198: Output functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fgl_report_configureAutoformatOutput</code></td>
<td>Configure the output when auto-formatting is enabled.</td>
</tr>
<tr>
<td></td>
<td>(fontName STRING, fontName STRING, fontSize INTEGER, fidelity BOOLEAN, reportTitle STRING,</td>
</tr>
<tr>
<td></td>
<td>fieldNamePatterns STRING, systemId STRING)</td>
</tr>
<tr>
<td><code>fgl_report_configureCompatibilityOutput</code></td>
<td>Configure the output for BDL ASCII reports (compatibility reports) being run in graphical mode using Genero Report Writer.</td>
</tr>
<tr>
<td></td>
<td>(pageWidthInCharacters INTEGER, fontName STRING, fidelity BOOLEAN, reportName NULL,</td>
</tr>
<tr>
<td></td>
<td>reportCategory STRING, systemId STRING)</td>
</tr>
<tr>
<td><code>fgl_report_configureHTMLDevice</code></td>
<td>Configure the HTML output.</td>
</tr>
<tr>
<td></td>
<td>(fromPage INTEGER, toPage INTEGER, embeddedImages INTEGER,</td>
</tr>
<tr>
<td></td>
<td>imageGenerationDirectoryName STRING, imageUrlPrefix STRING, removeWhitespace INTEGER,</td>
</tr>
<tr>
<td></td>
<td>ignoreRowAlignment INTEGER, ignoreColumnAlignment INTEGER,</td>
</tr>
<tr>
<td></td>
<td>removeBackgroundImages INTEGER)</td>
</tr>
<tr>
<td><code>fgl_report_configureImageDevice</code></td>
<td>Configure the image output.</td>
</tr>
<tr>
<td></td>
<td>(antialiasFonts BOOLEAN, antialiasShapes BOOLEAN,</td>
</tr>
<tr>
<td></td>
<td>monochrome BOOLEAN, fromPage INTEGER, toPage INTEGER,</td>
</tr>
<tr>
<td></td>
<td>fileType STRING, imagePath STRING, fileNamePrefix STRING, resolution INTEGER)</td>
</tr>
<tr>
<td><code>fgl_report_configureLocalization</code></td>
<td>Configures the localization for the current report.</td>
</tr>
<tr>
<td></td>
<td>(charSet STRING, resourcePath STRING, numberFormat STRING,</td>
</tr>
<tr>
<td></td>
<td>dateFormat STRING)</td>
</tr>
<tr>
<td><code>fgl_report_configureOORTFDevice</code></td>
<td>Configure RTF output for Open Office.</td>
</tr>
<tr>
<td></td>
<td>(fromPage INTEGER, toPage INTEGER, imagesResolution INTEGER,</td>
</tr>
<tr>
<td></td>
<td>imagesFormat STRING)</td>
</tr>
<tr>
<td><code>fgl_report_configurePDFDevice</code></td>
<td>Configure the PDF output.</td>
</tr>
<tr>
<td></td>
<td>(fontDirectory STRING, antialiasFonts BOOLEAN, antialiasShapes BOOLEAN,</td>
</tr>
<tr>
<td></td>
<td>monochrome BOOLEAN, fromPage INTEGER, toPage INTEGER)</td>
</tr>
</tbody>
</table>
Table 199: Printer functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fgl_report_setPrinterChromaticity(chromaticity STRING)</td>
<td>Control color selection of the printer.</td>
</tr>
<tr>
<td>fgl_report_setPrinterCopies(copies INTEGER)</td>
<td>Specify the number of copies to be printed.</td>
</tr>
<tr>
<td>fgl_report_setPrinterDestinationUrl(destination STRING)</td>
<td>Specify an alternate destination for the spooled printer formatted data.</td>
</tr>
<tr>
<td>fgl_report_setPrinterFidelity(fidelity INTEGER)</td>
<td>Select printer high fidelity mode.</td>
</tr>
<tr>
<td>fgl_report_setPrinterJobImpressions(jobImpressions INTEGER)</td>
<td>Specify the total size in number of impressions.</td>
</tr>
<tr>
<td>fgl_report_setPrinterJobMediaSheets(jobMediaSheets INTEGER)</td>
<td>Specify the total number of media sheets.</td>
</tr>
<tr>
<td>fgl_report_setPrinterJobName(jobName STRING)</td>
<td>Specify a name for the job.</td>
</tr>
<tr>
<td>fgl_report_setPrinterJobSheets(jobSheets STRING)</td>
<td>Control job sheet printing.</td>
</tr>
<tr>
<td>fgl_report_setPrinterMediaName(mediaTray STRING)</td>
<td>Select the type of media to use.</td>
</tr>
<tr>
<td>fgl_report_setPrinterMediaSizeName(mediaSizeName STRING)</td>
<td>Select the media size to be used.</td>
</tr>
<tr>
<td>fgl_report_setPrinterMediaTray(mediaTray STRING)</td>
<td>Select the tray of the printer.</td>
</tr>
<tr>
<td>fgl_report_setPrinterName(printerName STRING)</td>
<td>Select a specific printer by name.</td>
</tr>
<tr>
<td>fgl_report_setPrinterNumberUp(numberUp INTEGER)</td>
<td>Specify the number of print stream pages for a single side of an instance.</td>
</tr>
<tr>
<td>fgl_report_setPrinterOrientationRequested(orientationRequested STRING)</td>
<td>Control the paper orientation.</td>
</tr>
<tr>
<td>fgl_report_setPrinterPageRanges(pageRanges STRING)</td>
<td>Specify the ranges of pages to print.</td>
</tr>
<tr>
<td>fgl_report_setPrinterPJLVariables(values om.SaxAttributes STRING)</td>
<td>Specify the variables for printers that support Printer Job Language (PJL).</td>
</tr>
<tr>
<td>fgl_report_setPrinterPrintQuality(printQuality STRING)</td>
<td>Control the quality used by the printer.</td>
</tr>
<tr>
<td>fgl_report_setPrinterRequestingUserName(requestingUserName STRING)</td>
<td>Specify end user's name.</td>
</tr>
<tr>
<td>fgl_report_setPrinterResolution(resolution STRING)</td>
<td>Specify an exact resolution for the printer.</td>
</tr>
<tr>
<td>fgl_report_setPrinterSheetCollate(sheetCollate STRING)</td>
<td>Controls the collation of multiple copies.</td>
</tr>
<tr>
<td>fgl_report_setPrinterSides(sides STRING)</td>
<td>Specify the mapping of pages on the physical media.</td>
</tr>
<tr>
<td>fgl_report_setPrinterWriteToFile(file STRING)</td>
<td>Specify a file where the report is written in postscript.</td>
</tr>
<tr>
<td></td>
<td>(deprecated!)</td>
</tr>
</tbody>
</table>
### Table 200: Distributed Mode functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fgl_report_configureDistributedEnvironment(FGLDIR STRING, FGLPROFILE STRING, FGLRESOURCEPATH STRING, DBPATH STRING)</td>
<td>Configure the environment in the case of distributed processing.</td>
</tr>
<tr>
<td>fgl_report_configureDistributedProcessing(host STRING, port INTEGER)</td>
<td>Configure processing via a dedicated server.</td>
</tr>
<tr>
<td>fgl_report_setDistributedRequestingUserName(requestingUserName STRING)</td>
<td>Specify end user’s name for the purpose of identifying log entries in the case of distributed processing.</td>
</tr>
<tr>
<td>fgl_report_configureDistributedURLPrefix(urlPrefix STRING)</td>
<td>Configures the URL prefix containing the host and optionally the port and prepended paths for previewing a document.</td>
</tr>
</tbody>
</table>

### Using report output functions

When you load the current settings of a report, you load report output options specified in the report design document. To change how the report is output, the output functions must be called after the loading of the current settings (fgl_report_loadCurrentSettings on page 720), but before committing the current settings (fgl_report_commitCurrentSettings on page 721).

In this example, the Genero BDL code fragment uses the fgl_report_selectDevice on page 743 function to change the output device to PDF and the fgl_report_selectPreview on page 745 function to select to preview the report.

```bulat
IF fgl_report_loadCurrentSettings(r_filename) THEN -- load the 4rp file
  CALL fgl_report_selectDevice("PDF")               -- change to PDF
  CALL fgl_report_selectPreview(TRUE)               -- preview file using default previewer
  -- ADD ADDITIONAL API CALLS HERE, BEFORE YOU COMMIT CURRENT SETTINGS.
  LET handler = fgl_report_commitCurrentSettings()  -- commit changes
ELSE
  EXIT PROGRAM
END IF
```

### Device-specific function summary list

Device-specific configuration functions listed by device.

### Table 201: Device-specific functions

<table>
<thead>
<tr>
<th>Device</th>
<th>Available functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML</td>
<td>fgl_report_configureHTMLDevice on page 733.</td>
</tr>
<tr>
<td>Device</td>
<td>Available functions</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td>OORTF</td>
<td><code>fgl_report_configureOORTFDevice</code> on page 735,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setRTFMemoryThreshold</code> on page 763.</td>
</tr>
<tr>
<td>PDF</td>
<td><code>fgl_report_configurePDFDevice</code> on page 736,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_configurePDFFontEmbedding</code> on page 736,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPDFJPEGImageEncoding</code> on page 751,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPDFImageResolution</code> on page 751.</td>
</tr>
<tr>
<td>Postscript</td>
<td>None.</td>
</tr>
<tr>
<td>Printer</td>
<td><code>fgl_report_setPrinterChromaticity</code> on page 752,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterCopies</code> on page 752,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterFidelity</code> on page 753,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterJobImpressions</code> on page 753,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterJobMediaSheets</code> on page 754,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterJobName</code> on page 754,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterJobPriority</code> on page 754,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterJobSheets</code> on page 755,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterMediaName</code> on page 755,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterMediaSizeName</code> on page 756,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterMediaTray</code> on page 756,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterName</code> on page 757,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterNumberUp</code> on page 757,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterOrientationRequested</code> on page 757,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterPageRanges</code> on page 758,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterPrintQuality</code> on page 759,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterRequestingUserName</code> on page 759,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterResolution</code> on page 760,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterSheetCollate</code> on page 760,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterSides</code> on page 761,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setPrinterWriteToFile</code> on page 761.</td>
</tr>
<tr>
<td>RTF</td>
<td><code>fgl_report_configureRTFDevice</code> on page 737,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setRTFMemoryThreshold</code> on page 763.</td>
</tr>
<tr>
<td>SVG</td>
<td><code>fgl_report_configureSVGDevice</code> on page 737,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setSVGCompression</code> on page 763,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setSVGCopies</code> on page 764,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setSVGOrientationRequested</code> on page 764,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setSVGPaperSource</code> on page 765,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setSVGPaperSource</code> on page 765,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setSVGPreview</code> on page 738,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setSVGPrinterName</code> on page 765,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setSVGSheetCollate</code> on page 766.</td>
</tr>
<tr>
<td>XLS</td>
<td><code>fgl_report_configureXLSDevice</code> on page 739,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setXLSMergeCells</code> on page 766.</td>
</tr>
<tr>
<td>XLSX</td>
<td><code>fgl_report_configureXLSXDevice</code> on page 740,</td>
</tr>
<tr>
<td></td>
<td><code>fgl_report_setXLSXMergeCells</code> on page 767.</td>
</tr>
</tbody>
</table>

`fgl_report_configureAutoformatOutput`
Configure the output when auto-formatting is enabled.

**Syntax**

```c
fgl_report_configureAutoformatOutput ( 
    fontName STRING,
    fontSize INTEGER,
    fidelity BOOLEAN,
    reportTitle STRING,
    fieldNamePatterns STRING,
    systemId STRING )
```
1. **fontName** specifies the font to use. For the COMPATIBILITY type, the default font is Lucida Console on Windows® systems and Lucida Sans Typewriter on all other systems. For the other FLAT LIST and NEW LIST types, the default font is the logical SansSerif font.

2. **fontSize** specifies the font size to use. For the COMPATIBILITY type, this value is ignored because the best font size is automatically selected. For the other FLAT LIST and NEW LIST types, the default font size is 12.

3. **fidelity** specifies whether or not to set the **fidelity** property for the produced WORDBOX objects. See WORDBOX for more information.

4. **reportTitle** - Title of the report. For the COMPATIBILITY type, this value is ignored. For the FLAT LIST type, the default value is "Report Title Here". For the NEW LIST type, the default value is an empty string.

5. **fieldNamePatterns** - A comma separated list of field name patterns; fields not matching any of the patterns are not printed. The patterns may contain literal characters, the ? question mark, the * star character, and character ranges, as defined for the Genero BDL MATCHES operator. The columns of the output are sorted in order of the patterns matched and within one pattern by the relative position of the field in the PRINT statement. By default all fields are printed.

6. **systemId** specifies an absolute URL against which relative resources such as images in overlays are resolved. By default, external resources must be specified with absolute URLs.

**Usage**

This function is applicable when no 4rp template has been specified in the call to either `fgl_report_loadCurrentSettings` or `fgl_report_loadAndCommit`, and auto-formatting with a value other than COMPATIBILITY has been selected by a call to `fgl_report_setAutoformatType`.

All arguments to this function are optional (indicated by passing a null value).

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_configureCompatibilityOutput**

Configure the output for BDL ASCII reports (compatibility reports) being run in graphical mode using Genero Report Writer.

**Syntax**

```
fgl_report_configureCompatibilityOutput (
    pageWidthInCharacters INTEGER,
    fontName STRING,
    fidelity BOOLEAN,
    reportName NULL,
    reportCategory STRING,
    systemId STRING)
```

1. **pageWidthInCharacters** - This value overwrites any RIGHT MARGIN specification in the report definition. If the report does not contain a RIGHT MARGIN specification and this value is not set, a value of 132 is assumed.

2. **fontName** - Specifies the font to use. The default is a fixed pitch font.

3. **fidelity** - Ensures that the text preview and text printout are 100% the same. The font is not embedded in the report document, it is drawn similar to an image. As a result, you may not be able to select the text in the resulting report, depending on the output format chosen (pdf, for example). For further information, see the **fidelity** property.

4. **reportName** - the value for this parameter is now provided internally. However, because of backwards compatibility, you must set the value to NULL when you call this function.

5. **reportCategory** - Specifies the category of the report. The value specified is passed to the overloadable callback function `compat_placePageBackground` in $GREDIR/src/overloadables/CompatCustom.4gl. By default, the Genero Report Engine calls `compat_placePageBackground(out, reportName, reportCategory, pageNumber)`; if the report category is "demo" or "form", specific actions are taken to the report as written in the source code of CompatCustom.4gl. To provide for custom report categories, create a copy of CompatCustom.4gl in a
different path, then update \texttt{FGLLDPATH} and put the path containing your copy BEFORE the path containing the original; the DVM will load your file instead of the default. For example, you may want to place a company logo in all reports with the category "correspondence", such as an invoice, delivery receipt, and so on.

6. \textit{systemId} - Specifies an absolute URL against which relative resources such as images in overlays are resolved.

\textbf{Usage}

Function to optionally configure the output for BDL ASCII reports (compatibility reports) being run in graphical mode using Genero Report Writer.

This function is applicable when no \texttt{4rp} template has been specified in the call to either \texttt{fgl_report_loadCurrentSettings} or \texttt{fgl_report_loadAndCommit}. All arguments to this function are optional (indicated by passing a null value).

For an example of Genero code using a reporting function, see \textbf{Using report output functions} on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

\texttt{fgl_report_configureDistributedEnvironment}

Configure the environment in the case of distributed processing.

\textbf{Syntax}

\begin{verbatim}
fgl_report_configureDistributedEnvironment ( 
    FGLDIR STRING, 
    FGLPROFILE STRING, 
    FGLRESOURCEPATH STRING, 
    DBPATH STRING )
\end{verbatim}

1. \textit{FGLDIR} specifies the value of the environment variable \texttt{FGLDIR}. Passing NULL for this value will default to the process environment variable value.

2. \textit{FGLPROFILE} specifies the value of the environment variable \texttt{FGLPROFILE}. Passing NULL for this value will default to the process environment variable.

3. \textit{FGLRESOURCEPATH} specifies the value of the environment variable \texttt{FGLRESOURCEPATH}. Passing NULL for this value will default to the process environment variable value.

4. \textit{DBPATH} specifies the value of the environment variable \texttt{DBPATH}. Passing NULL for this value will default to the environment process environment variable value.

\textbf{Usage}

Configure the environment. It is intended for the case of distributed processing with a server running on a different physical machine with different resource paths. It is not necessary to call this function if the daemon is running on the local machine or if the remote machine has identical resource directories.

For an example of Genero code using a reporting function, see \textbf{Using report output functions} on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

\texttt{fgl_report_configureDistributedProcessing}

Configure processing via a dedicated server.

\textbf{Syntax}

\begin{verbatim}
fgl_report_configureDistributedProcessing ( 
    host STRING, 
    port INTEGER )
\end{verbatim}

1. \textit{host} specifies the host name.

2. \textit{port} specifies the port number.
Usage

Specify the server where the Genero Report Engine is running in server mode. The engine is started on the remote machine via the command $GREDIR/bin/greportwriter -l port.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_configureDistributedURLPrefix

Configures the URL prefix containing the host and optionally the port and prepended paths for previewing a document.

Syntax

\[
\text{fgl_report_configureDistributedURLPrefix (}
\text{urlPrefix STRING }
\)\]

1. \textit{urlPrefix} is a URL by which the web server can be reached.

Usage

This function is applicable when these conditions apply:

- Previewing is selected by a call to the function \textit{fgl_report_selectPreview}.
- The preview format is PDF, XLS, XLSX, HTML, RTF, OORTF, or Postscript.
- The Genero Report Engine (GRE) is running in distributed mode on a remote machine.

\textbf{Note:} This option is not needed and not taken into consideration when the GRE is running in distributed mode on the same machine as the DVM.

- The web server deviates from the default "http://HOST:8080", where "HOST" denotes the value "host" passed in the call to \textit{fgl_report_configureDistributedProcessing(host, port)}.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_configureImageDevice

Configure the image output.

Syntax

\[
\text{fgl_report_configureImageDevice (}
\text{antialiasFonts BOOLEAN,}
\text{antialiasShapes BOOLEAN,}
\text{monochrome BOOLEAN,}
\text{fromPage INTEGER,}
\text{toPage INTEGER,}
\text{fileType STRING,}
\text{filePath STRING,}
\text{fileNamePrefix STRING,}
\text{resolution INTEGER)}
\]

1. \textit{antialiasFonts} - Configures whether fonts are rendered using antialiasing. The default value is true.
2. \textit{antialiasShapes} - Configures whether shapes are rendered using antialiasing. The default value is true.
3. \textit{monochrome} - Configures whether color values are converted to monochrome output. The default value is true.
4. \textit{fromPage} - Selects the lower bound of the range of pages to create images for. The default value is 1.
5. \textit{toPage} - Selects the upper bound of the range of pages to create images for. By default, images are created for all pages.
6. \textit{fileType} - One of jpg|png|bmp|gif. The default is jpg.
7. `filePath` - Path of the target directory where the images are created. If the specified path is relative, the
GREOUTPUTDIR environment variable is used to determine the full path. The default file path is the value
of the GREOUTPUTDIR environment variable, or the current working directory of the report engine if
GREOUTPUTDIR isn't set.

8. `fileNamePrefix` - Name prefix of the generated files. For example, setting namePrefix to "Chart" causes the
creation of files called "Chart0001", "Chart0002", and so on. The default value is "Image".

9. `resolution` - Controls the resolution used for creating the images. If the image is later viewed unscaled on a
device with the specified resolution, all items will have their correct metric length. Beware that high values may
require enormous amounts of memory and the resulting files may become very large. The formula for calculating
the memory consumption in bytes is `resolution_in_dpi_x * page_width_in_inches * resolution_in_dpi_y *`
`page_height_in_inches * 3byte` for a color image. For a page of format "letter" at 96 DPI we therefore get
96DPI*8.5"*96DPI*11"*3byte=2.6 MB. At 300 DPI this is 25 MB (color) and 8 MB (grayscale). The renderer
currently requires the entire page to be in memory. The default value is 96.

**Usage**

This function is applicable when image output has been selected by a call to the function `fgl_report_selectDevice`.

All arguments to this function are optional (indicated by passing a null value).

For an example of Genero code using a reporting function, see Using report output functions on page 727. This
element may not use the specific function discussed in this topic, however it provides details on where you would
place this (and other) report output functions.

`fgl_report_configureLabelOutput`

Configures the physical layout of a label page.

**Syntax**

```plaintext
fgl_report_configureLabelOutput (  
  paperWidth STRING,  
  paperHeight STRING,  
  labelWidth STRING,  
  labelHeight STRING,  
  labelsPerRow INTEGER,  
  labelsPerColumn INTEGER)
```

1. `paperWidth` - width of the page (e.g. a4width). You must specify this value.
2. `paperHeight` - height of the page (e.g. a4length). You must specify this value.
3. `labelWidth` - Physical width of a label (e.g. 99.1mm) The value specified here should be the same or larger than
   the width in the 4rp file. The value is optional (indicated by passing null). In this case the width specified in the
   4rp file is used.
4. `labelHeight` - Physical height of a label (e.g. 42.3mm) The value specified here should be the same or larger than
   the height in the 4rp file. The value is optional (indicated by passing null). In this case the height specified in the
   4rp file is used.
5. `labelsPerRow` - the number of labels across. You must specify this value.
6. `labelsPerColumn` - the number of labels down. You must specify this value.

For additional information about the strings that can be used to specify the parameter values, see Dimensions on page
977.

**Usage**

Function that configures the physical layout of a label page.

This function is applicable when selecting "label" as the mapping option by calling the function
`fgl_report_selectLogicalPageMapping`. This figures the physical layout by specifying the paper
dimensions, the physical label size and the \( n \times m \) layout. The physical margins (distance between page border and
label) are specified by calling `fgl_report_setPaperMargins`.

---

**Report Writer**

---

---

---

---

---
For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**Related concepts**

Create labels: the report program (Genero BDL) on page 692

A common use of reports is to create a page of labels, such as address labels. Creating labels requires you to create a report design document representing a single label and a report application that prints multiple labels on a page.

*fgl_report_configureLocalization*

Configures the localization for the current report.

**Syntax**

```c
fgl_report_configureLocalization(
    charSet STRING,
    resourcePath STRING,
    numberFormat STRING,
    dateFormat STRING)
```

1. `charSet` - Specifies the encoding of the translation files (str and 42s files). By default, the system encoding is used.
2. `resourcePath` - A colon- or semicolon-delimited (Windows®) list of directories specifying the search path for compiled translation files (42s). The default is the value of the FGLRESOURCEPATH environment variable.
3. `numberFormat` - A formatting string to be used for number formatting. The format is compatible with the format of the Genero Business Development Language DBFORMAT environment variable. The default is the value of the DBFORMAT environment variable.
4. `dateFormat` - A formatting string to be used for date formatting. The format is compatible with the format of the Genero Business Development Language DBDATE environment variable. The default is the value of the DBDATE environment variable.

**Usage**

Function that configures the localization for the current report.

The function is not applicable for callback localization (See *fgl_report_setCallbackLocalization*).

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**Related concepts**

Change localization settings at runtime on page 717

Use the *fgl_report_configureLocalization* function to change the directory where the program seeks the translation files and to change the formatting pattern for numerical data at runtime.

*fgl_report_configureHTMLDevice*

Configure the HTML output.

**Syntax**

```c
fgl_report_configureHTMLDevice(
    fromPage INTEGER,
    toPage INTEGER,
    embedImages INTEGER,
    imageGenerationDirectory STRING,
    imageURLPrefix STRING,
    removeWhitespace INTEGER,
    ignoreRowAlignment INTEGER,
)```
ignoreColumnAlignment INTEGER, removeBackgroundImages INTEGER )

1. fromPage - Selects the lower bound of the range of pages to include in the HTML document. The default value is 1.
2. toPage - Selects the upper bound of the range of pages to include in the HTML document. By default all pages are included.
3. embedImages - Specifies whether to embed images in the resulting HTML output. By default this is not the case.
4. imageGenerationDirectory - If images are not embedded, this property specifies the directory into which generated images are written. The directory needs to exist, it is not created. Note that the urls that are created will not take this value into account. By default, the urls that are created contain the image name only, so that the images are expected to reside in the same directory as the document. If needed, the url prefix for the generated urls can be changed with the property imageUrlPrefix.
5. imageUrlPrefix - If images are not embedded, this property specifies the prefix of the urls of the generated images. As an example consider that an image of the name "12345.png" is created, and that this property is set to the value "/images/"; then the "src" attribute of the generated "img" element would be set to "/images/12345.png".
6. removeWhitespace - Controls whether cells should be created for empty strings. By default whitespace is stripped from the document.
7. ignoreRowAlignment - When set, only those objects that are entirely above or entirely below each other will go in separate rows. When set, the option reduces the amount of rows, thereby losing the horizontal alignment. The placement is not changed so that stacked items remain stacked. By default row alignment is not ignored.
8. ignoreColumnAlignment - When set, only those objects that are entirely to the left or entirely to the right of each other will go in separate columns. When set, the option reduces the amount of columns, thereby losing the vertical alignment. The placement is not changed so that adjacent items remain adjacent. By default column alignment is not ignored.
9. removeBackgroundImages - Controls the behavior when an IMAGEBOX is partially obscured by another element. When set, the image is removed from the resulting document; otherwise, the handling is as in any other case of overlapping items. By default, background images are removed.

Usage

Function to configure the HTML output.

This function is applicable when HTML output has been selected by a call to the function fgl_report_selectDevice. All arguments to this function are optional (indicated by passing a null value). If the HTML document should be written to a file, the general functions fgl_report_setOutputFileName and fgl_report_selectPreview are available for this purpose.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_configureMultipageOutput

Configure the multipage output for ISO or JIS formats.

Syntax

fgl_report_configureMultipageOutput ( pageExponent STRING, isoNumber INTEGER, portrait BOOLEAN )

1. pageExponent - Specifies the number of pages to print. The actual number of pages is calculated by the formula count=2^pageExponent.
2. isoNumber - Specifies the ISO number (e.g. 4 to ISOA4) This parameter is optional (indicated by passing an null value). If no value is specified, the page size of the logical page is taken from the 4rp file (if specified).
3. **portrait** - TRUE specifies that the page is in portrait orientation; FALSE = landscape. This parameter is optional (indicated by passing a null value). If no value is specified, the value of the logical page is taken from the 4rp file (if specified), or from the value specified in a call to `fgl_report_configurePageSize`.

**Usage**

Function to configure the multipage output for ISO or JIS formats.

This function is applicable for ISO or JIS formats and enables the printing of several logical pages per physical page. The number of pages per page is always a power of 2.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**Related concepts**

[Create multi-page ISO reports](#) on page 714

For reports printed on ISO and JIS-sized pages, you can configure the output to print several logical pages per physical page.

**fgl_report_configureOORTFDevice**

Configure RTF output for Open Office.

**Syntax**

```c
fgl_report_configureOORTFDevice (  
    fromPage INTEGER,  
    toPage INTEGER,  
    imagesResolution INTEGER,  
    imagesFormat STRING )
```

1. **fromPage** - Selects the lower bound of the range of pages to include in the RTF document. The default value is 1.
2. **toPage** - Selects the upper bound of the range of pages to include in the RTF document. By default all pages are included.
3. **imagesResolution** - Specifies the resolution of embedded images. In addition to ImageBoxes, content from BarCodeBoxes, business charts, and HTMLBoxes are embedded as images. The default image resolution is 300.
4. **imagesFormat** - One of: "png", "jpg". Controls the format of images embedded in the RTF document. Select jpg for compactness, png for lossless compression. The default image format is png.

**Usage**

Function to configure RTF output for Open Office.

This function is applicable when RTF output has been selected by a call to the function `fgl_report_selectDevice`. All arguments to this function are optional (indicated by passing a null value).

If the RTF document should be written to a file, the general functions `fgl_report_setOutputFileName` and `fgl_report_selectPreview` are available for this purpose.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_configurePageSize**

Set values for the page height and page width for Genero BDL ASCII reports being run in graphical mode.

**Syntax**

```c
fgl_report_configurePageSize(  
    pageWidth STRING,  
```
### pageHeight STRING

1. `pageWidth` - width of the page (e.g. a4width).
2. `pageHeight` - height of the page (e.g. a4length).

### Usage

The page dimensions of the report are read from the 4rp file. This function is used to override the values found there.

For BDL ASCII reports being run in graphical mode (compatibility reports, no 4rp file is being used), this function is used to set values for the page height and page width.

See Dimensions on page 977 for additional examples of the strings that can be used in these parameters.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

### fgl_report_configurePDFDevice

Configure the PDF output.

#### Syntax

```c
fgl_report_configurePDFDevice(fontDirectory STRING,
                               antialiasFonts BOOLEAN,
                               antialiasShapes BOOLEAN,
                               monochrome BOOLEAN,
                               fromPage INTEGER,
                               toPage INTEGER)
```

1. `fontDirectory` - Absolute path to the directory containing the font files. This new font directory is added to the list of standard font directories for your platform.
2. `antialiasFonts` - Configures whether fonts should be rendered using antialiasing. The default value is false.
3. `antialiasShapes` - Configures whether shapes should be rendered using antialiasing. The default value is false.
4. `monochrome` - Configures whether color values should be converted to monochrome output. The default value is false.
5. `fromPage` - Selects the lower bound of the range of pages to include in the PDF document. The default value is 1.
6. `toPage` - Selects the upper bound of the range of pages to include in the PDF document. By default all pages are included.

#### Usage

Function to configure the PDF output.

This function is applicable when PDF output has been selected by a call to the function `fgl_report_selectDevice`. All arguments to this function are optional (indicated by passing a null value).

If the PDF document should be written to a file, the general function `fgl_report_setOutputFileName` and `fgl_report_selectPreview` are available for this purpose.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

### fgl_report_configurePDFFontEmbedding

Configure the font embedding in PDF output.

#### Syntax

```c
fgl_report_configurePDFFontEmbedding()
```

1. `fromPage` - Selects the lower bound of the range of pages to include in the PDF document. The default value is 1.
2. `toPage` - Selects the upper bound of the range of pages to include in the PDF document. By default all pages are included.

#### Usage

Function to configure the font output.

This function is applicable when PDF output has been selected by a call to the function `fgl_report_selectDevice`. All arguments to this function are optional (indicated by passing a null value).

If the PDF document should be written to a file, the general function `fgl_report_setOutputFileName` and `fgl_report_selectPreview` are available for this purpose.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.
1. preferUnicodeEncoding - Specified to encode characters in UNICODE whenever possible. This option should be set if non-Latin characters are used in the document. Unsetting the option improves processing speed and yields slightly smaller documents. By default the parameter has a value of TRUE.

Usage
This function configures the font embedding in PDF output.

This function is applicable when PDF output has been selected by a call to the function `fgl_report_selectDevice`.

The argument to this function is optional, indicated by passing a null value.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureRTFDevice`
Configure RTF output for Microsoft™.

Syntax

```genero
fgl_report_configureRTFDevice(
   fromPage INTEGER,
   toPage INTEGER,
   imagesResolution INTEGER,
   imagesFormat STRING )
```

1. `fromPage` - Selects the lower bound of the range of pages to include in the RTF document. The default value is 1.
2. `toPage` - Selects the upper bound of the range of pages to include in the RTF document. Per default all pages are included.
3. `imagesResolution` - Specifies the resolution of embedded images. In addition to ImageBoxes, content from BarCodeBoxes, business charts and HTMLBoxes are embedded as images.
4. `imagesFormat` - One of : "png", "jpg". Controls the format of images embedded in the RTF document. Select jpg for compactness, png for lossless compression.

Usage
Function to configure RTF output.

This function is applicable when RTF output has been selected by a call to the function `fgl_report_selectDevice`. All arguments to this function are optional (indicated by passing a null value).

If the RTF document should be written to a file, the general functions `fgl_report_setOutputFileName` and `fgl_report_selectPreview` are available for this purpose.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureSVGDevice`
Configure the SVG output.

Syntax

```genero
fgl_report_configureSVGDevice(
   antialiasFonts BOOLEAN,
   antialiasShapes BOOLEAN,
   embedFonts BOOLEAN,
   ```
1. **antialiasFonts** - Configures whether fonts should be rendered using antialiasing. The default value is false.
2. **antialiasShapes** - Configures whether shapes should be rendered using antialiasing. The default value is false.
3. **embedFonts** - Controls whether fonts are embedded within the document. The default value is true.
4. **charsetToEmbed** - SVG offers the possibility to embed fonts within the document. Documents with embedded fonts are slightly larger, but they offer the advantage of exact reproduction. Normally one would embed exactly those characters that were used throughout the document, but that would require reading the entire document before creating any SVG output. This attribute provides a solution that does not compromise streaming. The characters in the specified character set are embedded, not requiring all of them to have been used. Note that the character sets are neither restricted to 255 characters, nor is there any restriction on which unicode characters are used. Furthermore, an entry in the character set can be a sequence of unicode characters, thus allowing for ligatures. Valid values are:
   - **DEFAULT** - embeds all characters from the code pages iso-8859-1 through iso-8859-10 (about 600 characters).
   - **ISO-8859-15** - embeds all characters from the named code page (about 200 characters).
   - **ALL** - embeds all characters available in the selected fonts. This option should only be used when Asian characters are needed, since the size of the glyph definitions will significantly grow the document size.

**Usage**

Function to configure the SVG output.

This function is applicable when SVG output has been selected by a call to the function `fgl_report_selectDevice`. All arguments to this function are optional (indicated by passing a null value). If the SVG document should be written to a file, the general function `fgl_report_setOutputFileName` and `fgl_report_selectPreview` are available for this purpose.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_configureSVGPreview**

Select how the document is handled by the SVG previewer.

This function is available only for the device SVG when previewing is selected.

**Syntax**

```genero
fgl_report_configureSVGPreview(
    preview STRING)
```

1. **preview** - There are four possible options:

   - **Preview**
     - Makes the previewer visible and shows the document in a tab folder.
   - **ShowPrintDialog**
     - Pops up the print dialog allowing the user to select and configure a printer. The document is printed in the background and the previewer main window is not shown.
   - **PrintOnDefaultPrinter**
     - Prints the report silently on the default printer. The previewer main window is not shown.
   - **PrintOnNamedPrinter**
     - Prints the report silently on a specific printer. The previewer main window is not shown. The printer is named by a call to `fgl_report_setSVGPrinterName()`. 
and the page range can be set by calling fgl_report_setSVGPagination(). If the previewer is on Windows, then the paper source can be selected with the function fgl_report_setSVGPaperSource. Additional parameters that can be set for printing from the SVG previewer include fgl_report_setSVGCopies and fgl_report_setSVGSheetCollate.

Usage

This function is available only for the device SVG when previewing is selected.

Important: Preview options set by fgl_report_configureSVGPreview are only valid when previewing with Genero Report Viewer in a Desktop configuration (fgl_report_selectDevice("SVG")). They are not valid when previewing with Genero Report Viewer for HTML5 in a Web configuration (fgl_report_selectDevice("Browser")).

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_configureXLSDevice

Configure the XLS (Excel) output.

Syntax

```c
fgl_report_configureXLSDevice(
    fromPage INTEGER,
    toPage INTEGER,
    removeWhitespace INTEGER,
    ignoreRowAlignment INTEGER,
    ignoreColumnAlignment INTEGER,
    removeBackgroundImages INTEGER,
    mergePages INTEGER
)
```

1. **fromPage** - Selects the lower bound of the range of pages to include in the XLS document. The default value is 1.
2. **toPage** - Selects the upper bound of the range of pages to include in the XLS document. By default all pages are included.
3. **removeWhitespace** - Controls whether or not cells should be created for empty strings. By default whitespace is stripped from the document.
4. **ignoreRowAlignment** - When set, only those objects that are entirely above or entirely below each other will go in separate rows. When set, the option reduces the amount of rows thereby losing the horizontal alignment. The placement is not changed so that stacked items remain stacked. By default row alignment is ignored.
5. **ignoreColumnAlignment** - When set, only those objects that are entirely to the left or entirely to the right of each other will go in separate columns. When set, the option reduces the amount of columns thereby losing the vertical alignment. The placement is not changed so that adjacent items remain adjacent. By default column alignment is ignored.
6. **removeBackgroundImages** - Controls the behavior in case an IMAGEBOX is partially obscured by another element. When set, the image is removed from the resulting document otherwise the handling is as with any other case of overlapping items. By default, background images are removed.
7. **mergePages** - Controls the behavior when the report has more than one page. By default a separate sheet is created per page. Setting this parameter causes the pages to be merged, creating a single result sheet unless the sheet has more that 65536 rows; in that case, the exceeding rows spill over into extra sheets. Setting this parameter and using a standard page size is the recommended way to produce single-sheet output; using a huge custom page size instead can adversely affect memory reclamation and performance. See also Output a single-sheet document to Excel on page 709.
Usage

Function to configure the XLS (Excel) output.

This function is applicable when XLS output has been selected by a call to the function fgl_report_selectDevice. All arguments to this function are optional, indicated by passing a null value. If the XLS document should be written to a file, the general functions fgl_report_setOutputFileName and fgl_report_selectPreview are available for this purpose.

See also Sending Data to an Excel spreadsheet.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_configureXLSXDevice

Configure Excel™ output in XLSX format.

Syntax

```gen
fgl_report_configureXLSXDevice ( 
  fromPage INTEGER, 
  toPage INTEGER, 
  removeWhitespace INTEGER, 
  ignoreRowAlignment INTEGER, 
  ignoreColumnAlignment INTEGER, 
  removeBackgroundImages INTEGER, 
  mergePages INTEGER)
```

1. `fromPage` - Selects the lower bound of the range of pages to include in the XLS document. The default value is 1.
2. `toPage` - Selects the upper bound of the range of pages to include in the XLS document. By default all pages are included.
3. `removeWhitespace` - Controls whether or not cells should be created for empty strings. By default whitespace is stripped from the document.
4. `ignoreRowAlignment` - When set, only those objects that are entirely above or entirely below each other will go in separate rows. When set, the option reduces the amount of rows thereby losing the horizontal alignment. The placement is not changed so that stacked items remain stacked. By default row alignment is ignored.
5. `ignoreColumnAlignment` - When set, only those objects that are entirely to the left or entirely to the right of each other will go in separate columns. When set, the option reduces the amount of columns thereby losing the vertical alignment. The placement is not changed so that adjacent items remain adjacent. By default column alignment is ignored.
6. `removeBackgroundImages` - Controls the behavior in case an IMAGEBOX is partially obscured by another element. When set, the image is removed from the resulting document otherwise the handling is as with any other case of overlapping items. By default, background images are removed.
7. `mergePages` - Controls the behavior when the report has more than one page. By default a separate sheet is created per page. See also Output a single-sheet document to Excel on page 709.

Usage

Function to configure Excel™ output in XLSX format. The functionality is identical to the existing "XLS" output except for:

- The format can be opened only by newer versions of Microsoft™ Excel™ (beginning with 2007). It is possible to download a backward compatibility pack from Microsoft™ that will allow opening of these files with older versions. Beware that in this case worksheets containing more then 65536 rows will be truncated at this limit.
- The new format overcomes the 65536 row limit of the "XLS" format. This is the main motivation for introducing this format.
- The new format is generated with constant memory consumption so that very large documents can be produced without exhausting heap space.
This function is applicable when "XLSX" output has been selected by a call to the function fgl_report_selectDevice. All arguments to this function are optional, indicated by passing a null value. If the XLSX document should be written to a file, the general functions fgl_report_setOutputFileName and fgl_report_selectPreview are available for this purpose.

See also Sending Data to an Excel spreadsheet.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_createProcessLevelDataFile
Configures the report execution to output an XML datafile.

Syntax

```
fgl_report_createProcessLevelDataFile (dataFileName STRING )
RETURNSING driver om.SaxDocumentHandler
```

1. dataFileName - Name of the data file to generate. You must specify this value.
2. driver - Returns an om.SaxDocumentHandler if no error occurred.

Usage

Function that configures the report execution to output an XML datafile.

This function will configure the report engine to output the data to a file. Calling this function will configure the report execution to output data to the specified file. You can use the function fgl_report_runReportFromProcessLevelDataFile to run a report getting the data from this XML file.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_findResourcePath
Returns the path to a resource searching first the working directory, then FGLRESOURCEPATH, and finally DBPATH. This function is deprecated.

Important: The function is deprecated, as it does not work in distributed mode. The function fgl_report_loadCurrentSettings now searches the path variables, making this function obsolete.

Syntax

```
fgl_report_findResourcePath (reportName STRING )
```

1. reportName - Specify the name of the report.

Usage

The function returns the absolute path to the resource specified in the argument. If a file cannot be found, the function returns the argument passed.

The function is typically used in conjunction with the function fgl_report_loadCurrentSettings() as in this example:

```
IF
  fgl_report_loadCurrentSettings(fgl_report_findResourcePath("OrderReport.4rp"))
THEN
```
Assuming FGLRESOURCEPATH is set to /home/fjs/reports/lists:/home/fjs/correspondence and the report OrderReport.4rp is located at /home/fjs/correspondence/OrderReport.4rp, that report will be loaded.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

*fgl_report_runReportFromProcessLevelDataFile*

Runs a report from a process level file.

**Syntax**

```c
fgl_report_runReportFromProcessLevelDataFile(
    out om.SaxDocumentHandler,
    fileName STRING
) RETURNING ok INTEGER
```

1. `out` - A output pipe obtained by a call to `fgl_report_commitCurrentSettings` or a similar function.
2. `fileName` - A file name of an XML file previously created by a call to `fgl_report_createProcessLevelDataFile`.
3. `ok` INTEGER - "true" if no error occurred.

**Usage**

Function that runs a report from a process level file, previously created by a call to `fgl_report_createProcessLevelDataFile`.

This function will replay the report from the file thereby replacing the running of the report (START REPORT, OUTPUT TO REPORT, FINISH REPORT statements).

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

*fgl_report_runFromXML*

Replays the report from an XML file.

**Syntax**

```c
fgl_report_runFromXML(
    dataFileName STRING
) RETURNING ok INTEGER
```

1. `dataFileName` - file name of an XML file containing the data.
2. `ok` - "true" if no error occurred.

**Usage**

Function that can run a report using as the data source an arbitrary XML file. This would be a file that was not created by a Genero program using `fgl_report_createProcessLevelDataFile`.

This function will replay the report from the file, replacing the running of the report (the START REPORT, OUTPUT TO REPORT, and FINISH REPORT statements).

The function sets up the report engine based on the current settings that have previously been loaded by a call to `fgl_report_loadCurrentSettings`, and may have been modified by calls to
The function automatically calls fgl_report_commitCurrentSettings.

See Data From an XML File for an example program.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

The Report Design

When it comes to the report design document (.4rp), you must declare the XSD associated with the XML file, as no data schema (.rdd) is created, as reports using this function do not include the Genero BDL report statements. See Support for arbitrary XML data sources on page 798.

fgl_report_selectDevice
Select the output device.

Parameters

See Usage.

Syntax

```c
fgl_report_selectDevice(
    device STRING )
```

1. **device** STRING - Sets the output device from this list:
   - HTML
   - Image
   - PDF
   - Postscript
   - Printer - Setting this parameter value selects server-side silent printing. Report Writer selects the printer that best meets the criteria specified by the functions listed in the Printer Functions section.
   - RTF (Microsoft™ RTF)
   - OORTF (Open Office RTF)
   - SVG
   - XLS
   - XLSX
   - Browser

   If a value is specified that is not in this list, an error message is displayed.

Usage

Function to select the output device.

Selecting a different output device changes the current settings. The choices are listed in Parameters.

Device-specific configuration functions are also available, as shown in Device-specific function summary list on page 727.

For more information about report output, see Change report output options on page 701.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.
**fgl_report_setEnvironment**
Specifies variable values in the private environment of the report.

**Syntax**

```plaintext
fgl_report_setEnvironment(
    values om.SaxAttributes)
```

1. `values` - Attribute set containing the values.

**Usage**
Can be used to specify the value of environment variables like `GREOUTPUTDIR` or user-defined variables, for the purpose of using the values in calls to the RTL function `Runtime.getEnvironmentVariable()`.

**Example**
This sample code sets a user-defined environment variable:

```plaintext
DEFINE envSet om.SaxAttributes
LET envSet=om.SaxAttributes.create()
CALL envSet.addAttribute("MYVARIABLE","1234")
CALL fgl_report_setEnvironment(envSet)
```

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_selectLogicalPageMapping**
Configures the mapping of logical pages to physical pages.

**Syntax**

```plaintext
fgl_report_selectLogicalPageMapping(
    mapping STRING)
```

1. `mapping` - one of oneToOne, labels, or multipage

**Usage**
Function that configures the mapping from logical pages (page dimensions in the 4rp file) to physical pages (actual page dimensions on the device).

**Table 202: Mapping options**

<table>
<thead>
<tr>
<th>Mapping Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oneToOne</td>
<td>The logical and physical pages are identical. Default value.</td>
</tr>
<tr>
<td>labels</td>
<td>For label printing it is advisable to design one label rather than creating a fixed ( n \times m ) layout. Such a layout can be printed on an arbitrary physical ( n \times m ) layout as long as the physical labels have at least the size of the logical labels. The physical dimensions of the layout must be configured using the function <code>fgl_report_configureLabelOutput</code> and <code>fgl_report_setPaperMargins</code>. The margins within a label can be configured by calling <code>fgl_report_setPageMargins</code>.</td>
</tr>
</tbody>
</table>
### Mapping Option | Description
--- | ---
multipage | For ISO and JIS sized pages it is possible to print several pages per page. This is achieved by setting the mapping to this value and calling the function `fgl_report_configureMultipageOutput` to configure the required number of pages.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

#### Related concepts
- **Create labels: the report program (Genero BDL) on page 692**
  A common use of reports is to create a page of labels, such as address labels. Creating labels requires you to create a report design document representing a single label and a report application that prints multiple labels on a page.
- **Create multi-page ISO reports on page 714**
  For reports printed on ISO and JIS-sized pages, you can configure the output to print several logical pages per physical page.

#### `fgl_report_selectPreview`
The `fgl_report_selectPreview` function determines whether the report is shown in the previewer or written to a file on disk.

#### Syntax
```c
fgl_report_selectPreview(
    preview INTEGER
)
```

1. `preview` should be set to 0 (FALSE) or 1 (TRUE). When set to FALSE, the file is generated on disk rather than being previewed. The default is TRUE.

#### Usage
Function to select preview mode.

This function sets the output to be shown in the previewer. Suitable to all output formats except "Image". The appropriate software needs to be installed on the client:

- For PDF: Acrobat™ Reader (or any other PDF reader)
- For SVG: the Genero Report Viewer is a part of the Genero Desktop Client
- For RTF: Microsoft™ Word
- For XLS and XLSX: Microsoft™ Excel
- For OORTF: Open Office (or any free Office), Open Office Writer is sufficient
- For HTML and Browser: Any browser

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

#### Related concepts
- **`fgl_report_setOutputFileName` on page 749**
Configure the file location for the device output. This function works for all output formats except Image.

*fgl_report_setAutoformatType*

Sets the auto-formatting type if no r template is specified.

**Syntax**

```c
fgl_report_setAutoformatType(
    type STRING )
```

1. `type` - One of "COMPATIBILITY", "FLAT LIST"

**Usage**

Function that sets the auto-formatting type if no r template is specified. By default such a report would be rendered in compatibility mode; this function provides additional rendering options:

- A compatibility report - the report will be output in ASCII format, as in legacy reports
- A flat list - this is a simple list design, similar to the output from the List Report template available from the Genero Studio main menu option File > New > Reports > Report Designs. This design is particularly well suited to produce Excel™ output from arbitrary reports.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

*fgl_report_setBrowserDocumentDirectory*

Configures the document directory for browser viewing.

**Syntax**

```c
fgl_report_setBrowserDocumentDirectory(
    directory STRING)
```

1. `directory` - The directory to hold the report files for use by Genero Report Viewer for HTML5, when Browser is selected as the output device.

**Usage**

See View report in a browser on page 713.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

*fgl_report_setBrowserDocumentDirectoryURL*

Configures the URL by which the document directory for browser viewing will be visible from a web server.

**Syntax**

```c
fgl_report_setBrowserDocumentDirectoryURL(
    directory STRING)
```

1. `directory` - The URL by which the document directory for browser viewing will be visible from a web server.

**Usage**

For use by Genero Report Viewer for HTML5, when Browser is selected as the output device. See View report in a browser on page 713.
For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setBrowserFontDirectory**

Configures the font directory for browser viewing.

**Syntax**

```c
fgl_report_setBrowserFontDirectory(
    directory STRING)
```

1. *directory* - The directory to hold the font files for use by Genero Report Viewer for HTML5, when Browser is selected as the output device.

**Usage**

See View report in a browser on page 713.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setBrowserFontDirectoryURL**

Configures the URL by which the font directory for browser viewing will be visible from a web server.

**Syntax**

```c
fgl_report_setBrowserFontDirectoryURL(
    directory STRING)
```

1. *directory* - The URL by which the font directory for browser viewing will be visible from a web server.

**Usage**

For use by Genero Report Viewer for HTML5, when Browser is selected as the output device. See View report in a browser on page 713.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setCallbackLocalization**

Configure the report to use a function to retrieve localized field titles.

**Syntax**

```c
fgl_report_setCallbackLocalization(
    share BOOLEAN)
```

1. *share* - when set, the report engine will call the GRE library function `report_getFieldCaption` for each field, to retrieve the field caption.

**Usage**

By default the field titles in a report are retrieved from Genero localization files compiled by the command `fglmkstr`. This function configures the report engine to call the function `report_getFieldCaption(matchName, fieldName)` instead, and doesn't use the localization file.
For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**Related concepts**

**Localized strings** on page 716
Localized strings can be used to implement internationalization in your application or to use site-specific text (for example, when business terms change based on territory).

**`fgl_report_setDistributedRequestingUserName`**
Specify end user's name for the purpose of identifying log entries in the case of distributed processing.

**Syntax**

```
fgl_report_setDistributedRequestingUserName(
    requestingUserName STRING )
```

1. `requestingUserName` is the user name.

**Usage**

In order to distinguish between log entries originating from different users, the messages can be prefixed with the “requestingUserName” value. The requesting user name is an arbitrary string defined by the client, where you can set the name of the end user who submitted the job (for example). The default is “not set”.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**`fgl_report_setImageShrinkImagesToPageContent`**
Configure image cropping.

**Syntax**

```
fgl_report_setImageShrinkImagesToPageContent(
    value BOOLEAN )
```

1. `value` - true or false.

**Usage**

Function to configure image cropping.

Sets whether the images produced are cropped to the page content (size of the page box) or have full page size.

The default value is false.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**`fgl_report_setImageUsePageNamesAsFileNames`**
Configure image file name generation.

**Syntax**

```
fgl_report_setImageUsePageNamesAsFileNames(
    value BOOLEAN )
```

1. `value` - true or false.
**Usage**

Function to configure image file name generation.

Sets whether the page names ("name" attribute) in the document should be used as image file names. In the case that a name is not unique, a disambiguation number is appended. In the case that a page does not set the "name" attribute, the default naming scheme explained in fgl_report_configureImageDevice on page 731 is used.

The default value is false.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setOutputFileName**

Configure the file location for the device output. This function works for all output formats except Image.

**Syntax**

```c
fgl_report_setOutputFileName(
    fileName STRING )
```

1. `fileName` - Enter a file name, with or without a file suffix, or a complete path with a file name.

   If no suffix is specified, a device-specific suffix is appended, with the extension corresponding to the output type chosen by the fgl_report_selectDevice() API call.

   If you do not specify a path, the file is written to the current directory. The default file path is the value of the GREOUTPUTDIR environment variable, or the current working directory of the report engine if GREOUTPUTDIR isn't set.

**Usage**

Function to configure the file location for the device output. This function works for all output formats except for the Image output format. For the Image output format, you must call the function fgl_report_configureImageDevice.

You can use this function when file preview has been disabled, by a call to fgl_report_selectPreview(FALSE).

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setPageMargins**

Configure the logical margins of a report.

**Syntax**

```c
fgl_report_setPageMargins(
    topMargin STRING,
    bottomMargin STRING,
    leftMargin STRING,
    rightMargin STRING)
```

1. `topMargin` - top margin value of the logical page (e.g. 0.5cm).
2. `bottomMargin` - bottom margin value of the logical page (e.g. 0.5cm).
3. `leftMargin` - left margin value of the logical page (e.g. 0.5cm).
4. `rightMargin` - right margin value of the logical page (e.g. 0.5cm).
**Usage**

Function that configures the logical margins of a report.

The logical margins of the report are read from the 4rp file. This function is used to override the values found there.

For BDL ASCII reports (compatibility reports, no 4rp file is specified), this function is used to set the values of the parameters:

For additional information about the units that can be used to specify the margin values, see Dimension Resolver.

In the case of label printing (see fgl_report_selectLogicalPageMapping), this function specifies the margins within a label. Similarly, when multi page output is selected (see fgl_report_selectLogicalPageMapping) the values specify the margins of the logical pages which can be smaller than the physical margins since the required width is not limited by device limitations but by aesthetic aspects only.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setPaperMargins**

Configure the physical margins of a report.

**Syntax**

```plaintext
fgl_report_setPaperMargins (  
    topMargin STRING,  
    bottomMargin STRING,  
    leftMargin STRING,  
    rightMargin STRING  )
```

1. **topMargin** - top margin value of the physical page (e.g. 0.5cm).
2. **bottomMargin** - bottom margin value of the physical page (e.g. 0.5cm).
3. **leftMargin** - left margin value of the physical page (e.g. 0.5cm).
4. **rightMargin** - right margin value of the physical page (e.g. 0.5cm).

**Usage**

Function that configures the physical margins of a report

The physical margins of the report can be set by this function for the case that either label printing or multi page output has been selected by a call to fgl_report_selectLogicalPageMapping.

For additional information about the units that can be used to specify the margin values, see Dimension Resolver.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setPageSwappingThreshold**

Sets the threshold for page-to-disk swapping.

**Syntax**

```plaintext
fgl_report_setPageSwappingThreshold (  
    value INTEGER  )
```

1. **value** - A positive integer greater than zero that specifies the maximum number of pages that may be held in main memory.
Usage

Function that sets the threshold for page-to-disk swapping to prevent memory exhaustion on very large documents that use "Page N of M".

The function specifies the maximum number of pages that may be held in main memory. When the value is exceeded, pages are swapped to the disk. This parameter is only needed for very large reports that contain references to the total number of pages. Alternatively, it is possible to grow the amount of memory that the Java™ JVM may allocate by setting the parameter -Xmx (e.g. "java -Xmx512m" for 512 MB) in the script $GREDIR/bin/greportwriter[.bat]. By default, the value is not set so that pages are never swapped to disk.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPDFJPEGImageEncoding`
Configure the encoding method of embedded images in PDF output.

Syntax

```java
fgl_report_setPDFJPEGImageEncoding (  
    encodeImagesAsJPEG BOOLEAN,  
    jpegQuality FLOAT )
```

1. `encodeImagesAsJPEG` - Specifies images to be encoded in JPEG format. The default value is False.
2. `jpegQuality` - Sets the compression quality to a value between 0 and 1. By default a value of 1 is used (highest quality).

Usage

Function to configure the encoding method of embedded images in PDF output.

Optional JPEG image encoding can be configured to shrink the size of PDF documents using this function. This function is applicable when PDF output has been selected by a call to the function `fgl_report_selectDevice`. All arguments to this function are optional (indicated by passing a null value).

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPDFImageResolution`
Configure the resolution of embedded images in PDF output.

Syntax

```java
fgl_report_setPDFImageResolution (  
    imagesResolution INTEGER )
```

1. `imagesResolution` - Specifies the maximum resolution of embedded images.

Usage

Function to configure the resolution of embedded images in PDF output.

This function is applicable when PDF output has been selected by a call to the function `fgl_report_selectDevice`.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.
**fgl_report_setPrinterChromaticity**
Control color selection of the printer.

**Syntax**

```plaintext
fgl_report_setPrinterChromaticity(
    chromaticity STRING )
```

1. `chromaticity` - one of: monochrome|color.

**Usage**

Function to control color selection of the printer

Controls how the print data should be generated or processed. It does not confine the printer selection to printers with the specified capability. Default value is color. Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setPrinterCopies**
Specify the number of copies to be printed.

**Syntax**

```plaintext
fgl_report_setPrinterCopies(
    copies INTEGER )
```

1. `copies` is the number of copies to print.

**Usage**

Function to specify the number of copies to be printed.

Specifies the number of copies to be printed. Default value: 1

Setting this option reduces the set of usable printers to those matching it.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setPrinterDestinationUrl**
Specify an alternate destination for the spooled printer formatted data.

**Syntax**

```plaintext
fgl_report_setPrinterDestinationUrl(
    destination STRING )
```

1. `destination` - the destination URL.

**Usage**

Function to specify an alternate destination for the spooled printer formatted data.
A URL indicating an alternate destination for the spooled printer formatted data. Some print services will not support the notion of a destination other than the printer device and so will not support this attribute. A common use for this attribute will be applications which want to redirect output to a local disk file: e.g. "file:out.prn". A more platform independent way is to set the `fgl_report_setPrinterName` to "stdout" so that postscript is written to stdout or to specify the `fgl_report_setPrinterWriteToFile` attribute which causes postscript to be written to the specified file. Another alternative may be the use of the `fgl_report_setDevice` to generate a PDF file.

By default, it is not set. Setting this option reduces the set of usable printers to those matching it.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterFidelity`
Select printer high fidelity mode.

**Syntax**

```
fgl_report_setPrinterFidelity( fidelity INTEGER )
```

1. *fidelity* - true or false

**Usage**

Function to select printer high fidelity mode.

When set, all attributes of the printer have to meet the requested values, otherwise the printout will fail. When set to false, a reasonable attempt to print the document is acceptable. Default value is false. Setting this option reduces the set of usable printers to those matching it.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterJobImpressions`
Specify the total size in number of impressions.

**Syntax**

```
fgl_report_setPrinterJobImpressions( jobImpressions INTEGER )
```

1. *jobImpressions* - number of jobs.

**Usage**

Function to specify the total size in number of impressions.

 Specifies the total size in number of impressions of the document. An "impression" is the image (possibly many print stream pages in different configurations) imposed onto a single media page. The *jobImpressions* attribute describes the size of the job. This attribute is not intended to be a counter; it is intended to be useful routing and scheduling information. The printer may try to compute the attribute's value if it is not supplied. Even if a value is supplied, the printer may choose to change the value if the printer is able to compute a value that is more accurate than the supplied value. The printer may be able to determine the correct value for the *jobImpressions* attribute right at job submission or at a later time. Unlike the `fgl_report_setPrinterJobMediaSheets` function, the value must not include the multiplicative factors contributed by the number of copies, specified by the `fgl_report_setPrinterCopies` function. Setting this option reduces the set of usable printers to those matching it.
For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterJobMediaSheets`
Specify the total number of media sheets.

**Syntax**

```plaintext
fgl_report_setPrinterJobMediaSheets (jobMediaSheets INTEGER)
```

1. `jobMediaSheets` INTEGER - number of sheets

**Usage**

Function to specify the total number of media sheets.

Specifies the total number of media sheets to be produced for this job. The `jobMediaSheets` attribute describes the size of the job. This attribute is not intended to be a counter; it is intended to be useful routing and scheduling information. The printer may try to compute the attribute's value if it is not supplied. Even if a value is supplied, the printer may choose to change the value if the printer is able to compute a value that is more accurate than the supplied value. The printer may be able to determine the correct value for the `jobMediaSheets` attribute right at job submission or at a later time. The value must include the multiplicative factors contributed by the number of copies, specified by the `fgl_report_setPrintercopies` function.

By default, this is not set. Setting this option reduces the set of usable printers to those matching it.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterJobName`
Specify a name for the job.

**Syntax**

```plaintext
fgl_report_setPrinterJobName (jobName STRING)
```

1. `jobName` - name of the job.

**Usage**

Function to specify a name for the job.

Name of the print job useful for tracking the job. The value does not have to be unique. Default: "not set" Setting this option reduces the set of usable printers to those matching it.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterJobPriority`
Specify a priority for the job.

**Syntax**

```plaintext
fgl_report_setPrinterJobPriority (jobPriority INTEGER)
```
1. jobPriority - priority of the job. Fails if the argument is not between 1 and 100.

Usage

Function to specify a priority for the job.

If supplied, the value specifies a priority for scheduling the job. A higher value specifies a higher priority. The value 1 indicates the lowest possible priority. The value 100 indicates the highest possible priority. Among those jobs that are ready to print, a printer must print all jobs with a priority value of n before printing those with a priority value of n-1 for all n. Default: "not set" Setting this option reduces the set of usable printers to those matching it.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

*fgl_report_setPrinterJobSheets*

Control job sheet printing.

Syntax

```interpreted
fgl_report_setPrinterJobSheets(
    jobSheets STRING)
```

1. jobSheets - one of: none|standard

Usage

Function to control job sheet printing.

Controls if job start and end sheets are to be printed. Default: none. Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

*fgl_report_setPrinterMediaName*

Select the type of media to use.

Syntax

```interpreted
fgl_report_setPrinterMediaName(
    mediaTray STRING)
```

1. mediaTray - one of: iso-a4-transparent|iso-a4-white|na-letter-transparent

Usage

Function to select the type of media to use.

Controls the type of media to choose. This function and the functions *fgl_report_setPrinterMediaSizeName* and *fgl_report_setPrinterMediaTray* are mutually exclusive. Default: "not set" Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.
**fgl_report_setPrinterMediaSizeName**

Select the media size to be used.

**Syntax**

```c
fgl_report_setPrinterMediaSizeName(
    mediaSizeName STRING)
```

1. **mediaSizeName** - one of: a|b|c|d|executive|folio|invoice|iso-a0|iso-a1|iso-a2|iso-a3|iso-a4|iso-a5|iso-a6|iso-a7|iso-a8|iso-a10|iso-b0|iso-b1|iso-b2|iso-b3|iso-b4|iso-b5|iso-b6|iso-b7|iso-b8|iso-b10|iso-c0|iso-c1|iso-c2|iso-c3|iso-c4|iso-c5|iso-c6|iso-designated-long|iso-italian-envelope|iso-oufuko-postcard|jis-b0|jis-b1|jis-b2|jis-b3|jis-b4|jis-b5|jis-b6|jis-a7|jis-b8|jis-b10|ledger|monarch-envelope|na-10x13-envelope|na-10x14-envelope|na-10x15-envelope|na-5x7|na-6x9-envelope|na-8x10|na-9x11-envelope|na-9x12-envelope|na-legal|na-letter|na-number-10-envelope|na-number-12-envelope|na-number-14-envelope|na-number-9-envelope|personal-envelope|quarto|tabloid

**Usage**

Function to select the media size to be used.

Selects the media size to be used. Normally this does not need to be specified, as the program will automatically select the smallest media onto which the current document can be printed without clipping or scaling. This function and the functions `fgl_report_setPrinterMediaTray` and `fgl_report_setPrinterMediaName` are mutually exclusive. Default value not set (is automatically selected). Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setPrinterMediaTray**

Select the tray of the printer.

**Syntax**

```c
fgl_report_setPrinterMediaTray(
    mediaTray STRING)
```

1. **mediaTray** STRING - one of: bottom|envelope|large-capacity|main|manual|middle|side|top

**Usage**

Function to select the tray of the printer.

Controls what tray to take the media from. This function and the functions `fgl_report_setPrintermediaTray` and `fgl_report_setPrintermediaName` are mutually exclusive. Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.
fgl_report_setPrinterName
Select a specific printer by name.

**Syntax**

```plaintext
fgl_report_setPrinterName (printerName STRING)
```

1. *printerName* - the name of the printer.

**Usage**

Function to select a specific printer by name.

Name of requested printer listed as "SERVICE". A special meaning is attached to the name `stdout`. If this value is specified, then postscript is written to `stdout`. Default value is "Not Set". Setting this option reduces the set of usable printers to those matching it.

For an example of Genero code using a reporting function, see [Using report output functions](#) on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_setPrinterNumberUp
Specify the number of print stream pages for a single side of an instance.

**Syntax**

```plaintext
fgl_report_setPrinterNumberUp (numberUp INTEGER )
```

1. *numberUp* - number of pages

**Usage**

Function to specify the number of print stream pages for a single side of an instance.

Specifies the number of print stream pages to impose upon a single side of an instance of selected medium. That is, if the numberUp value is *n*, the printer must place *n* print stream pages on a single side of an instance of the selected medium. To accomplish this, the printer may add some sort of translation, scaling or rotation.

**Note:** Since this feature is available only on a few printers, it is advisable to perform the necessary transformations using this API. Default: "not set" Setting this option reduces the set of usable printers to those matching it.

For an example of Genero code using a reporting function, see [Using report output functions](#) on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_setPrinterOrientationRequested
Control the paper orientation.

**Syntax**

```plaintext
fgl_report_setPrinterOrientationRequested (orientationRequested STRING)
```

1. *orientationRequested* - one of: landscape|portrait|reverse-landscape|reverse-portrait

**Usage**

Function to control the paper orientation.
Controls the paper orientation. Normally this value should not be set. The style sets the value internally by analyzing the page's width and height values. Default: not set Setting this option reduces the set of usable printers to those matching it.

Fails if argument is not one of the specified values.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

\textit{fgl\_report\_setPrinterPageRanges}

Specify the ranges of pages to print.

\textbf{Syntax}

\begin{verbatim}
fgl_report_setPrinterPageRanges ( pageRanges STRING )
\end{verbatim}

1. \textit{pageRanges} - page ranges following the syntax \([1-9][0-9]*-[1-9][0-9]*[, [1-9][0-9]*-[1-9][0-9]*]*

\textbf{Usage}

Function to specify the ranges of pages to print.

Specifies the ranges of print stream pages that the printer uses for each copy of the document printed. Nothing is printed for any pages identified that do not exist in the document. Default: "not set" (everything is printed) Setting this option reduces the set of usable printers to those matching it.

Fails if argument does not follow syntax.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

\textit{fgl\_report\_setPrinterPJLVariables}

Specify the variables for printers that support Printer Job Language (PJL).

\textbf{Syntax}

\begin{verbatim}
fgl_report_setPrinterPJLVariables ( values om.SaxAttributes STRING )
\end{verbatim}

1. \textit{values} - Attribute set containing the key/value pairs.

\textbf{Usage}

Use this function to configure your printer for server-side silent printing on network Postscript printers that support PJL.

\textbf{Important}: String values must be specified using double quotes, for example, "quoted string".

\textbf{Example}

This sample code sets the PJL variables \textit{BINDING} and \textit{OUTBIN}, and also sets the printer name:

\begin{verbatim}
DEFINE pjlSet om.SaxAttributes
LET pjlSet=om.SaxAttributes.create()
CALL pjlSet.addAttribute("BINDING","SHORTEDGE")
CALL pjlSet.addAttribute("OUTBIN","UPPER")
CALL fgl_report_setPrinterPJLVariables(pjlSet)
\end{verbatim}
CALL fgl_report_setPrinterName("pjl:10.0.1.052")

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**Related concepts**

*Printing without a driver using PJL* on page 704

Use the Printer Job Language (PJL) to print documents without the need to install drivers.

**fgl_report_setPrinterPrintQuality**

Control the quality used by the printer.

**Syntax**

```plaintext
fgl_report_setPrinterPrintQuality (printQuality STRING)
```

1. `printQuality` - one of: draft|high|normal

**Usage**

Function to control the quality used by the printer.

Specifies the print quality used by the printer. Default: not set Setting this option reduces the set of usable printers to those matching it.

Fails if argument is not one of the specified values

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setPrinterRequestingUserName**

Specify end user's name.

**Syntax**

```plaintext
fgl_report_setPrinterRequestingUserName (requestingUserName STRING)
```

1. `requestingUserName` - the user name.

**Usage**

Function to specify end user's name.

Specifies the end user's name who submitted the print job. A requesting user name is an arbitrary string defined by the client. The printer does not put the client-specific requestingUserName attribute into the print job's attribute set; rather, the printer puts in a jobOriginatingUserName attribute. This means that services which support specifying a username with this attribute should also report a jobOriginatingUserName attribute in the job's attribute set. Note that many print services may have a way to independently authenticate the user name, and so may state support for a requesting user name, but in practice will then report the user name authenticated by the service rather than that specified by this attribute. Default: "not set" Setting this option reduces the set of usable printers to those matching it.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.
**fgl_report_setPrinterResolution**
Specify an exact resolution for the printer.

**Syntax**

```
fgl_report_setPrinterResolution(
    resolution STRING)
```

1. **resolution** - the resolution of the printer. You can set a resolution for both the X-axis and the Y-axis, or you can set the resolution for the X-axis and the Y-axis separately. You can specify DPI (dots per inch) or DPCM (dots per centimeter). DPI is the default.

**Table 203: Valid resolution entries**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description (X-axis, Y-axis, Measure)</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>300,300.dpi</td>
</tr>
<tr>
<td>300 dpcm</td>
<td>300,300.dpcm</td>
</tr>
<tr>
<td>300,150</td>
<td>300,150.dpi</td>
</tr>
<tr>
<td>300,150,dpi</td>
<td>300,150.dpi</td>
</tr>
<tr>
<td>300,150,dpcm</td>
<td>300,150.dpcm</td>
</tr>
</tbody>
</table>

**Usage**

Function to specify an exact resolution supported by a printer or to be used for the job. This function assumes that printers have a small set of device resolutions at which they can operate, rather than a continuum.

By default, a resolution is not specified.

Setting this option reduces the number of usable printers to those matching the specified value.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setPrinterSheetCollate**
Controls the collation of multiple copies.

**Syntax**

```
fgl_report_setPrinterSheetCollate(
    sheetCollate STRING)
```

1. **sheetCollate** - Can be either **collated** or **uncollated** (default)

**Usage**

Specifies whether or not the pages of the document are to be in sequence, when multiple copies of the document are specified by the fgl_report_setPrinterCopies attribute. When **sheetCollate** is "collated", each copy of the document is printed with the pages in sequence. When **sheetCollate** is "uncollated", each page is printed a number of times equal to the value of the fgl_report_setPrinterCopies attribute in succession.

For example, suppose a document produces two pages as output, fgl_report_setPrinterCopies is 6, and **sheetCollate** is "uncollated"; in this case six copies of the first page are printed, followed by six copies of the second page.

**Tip**: It is discouraged to set **sheetCollate** to "collated" since it requires caching the entire document which is undesirable for large documents. If necessary, produce the document the required number of times.
Setting this option reduces the set of usable printers to those who are able to perform the collation as requested.

The function fails if the argument is not one of the specified values.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

Related concepts

fgl_report_setPrinterCopies on page 752
Specify the number of copies to be printed.

fgl_report_setPrinterSides
Specify the mapping of pages on the physical media.

Syntax

fgl_report_setPrinterSides(sides STRING)

1. sides - One of: one-sided|two-sided-short-edge|two-sided-long-edge

Usage

Function to specify the mapping of pages on the physical media.

Specifies how print-stream pages are to be imposed upon the sides of an instance of a selected medium, i.e., an impression.

• one-sided: Imposes each consecutive print-stream page upon the same side of consecutive media sheets.
• two-sided-short-edge: Imposes each consecutive pair of print stream pages upon front and back sides of consecutive media sheets, such that the orientation of each pair of print stream pages on the medium would be correct as if for binding along the short edge.
• two-sided-long-edge: Imposes each consecutive pair of print stream pages upon front and back sides of consecutive media sheets, such that the orientation of each pair of print stream pages on the medium would be correct as if for binding along the long edge.

By default, this is not set. Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_setPrinterWriteToFile
Specify a file where the report is written in postscript. (deprecated!)

Syntax

fgl_report_setPrinterWriteToFile(file STRING)

1. file - the destination file.

Usage

Function to specify a file where the report is written in postscript.
**Important:** The function is deprecated and replaced by the "Postscript" output format in calls to the function `fgl_report_selectDevice` in conjunction with the standard output specification function `fgl_report_setOutputFileName`.

A file name specifying a location where the report output is written in postscript format. If this attribute is set, all other IPP attribute values are ignored. Default: "not set" Setting this option reduces the set of usable printers to those matching it.

The default file path is the value of the GREOUTPUTDIR environment variable, or the current working directory of the report engine if GREOUTPUTDIR isn’t set.

The function fails if file cannot be opened for writing.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setProcessLevelDataFile`
Configures the report execution to output an XML datafile in addition to the regular processing.

**Syntax**

```plaintext
fgl_report_setProcessLevelDataFile (dataFileName STRING)
```

1. *dataFileName* - Name of the data file to generate.

**Usage**

Function that configures the report execution to output an XML datafile in addition to the regular processing.

Like the function `fgl_report_createProcessLevelDataFile`, this function causes a data file to be produced that can be used for archiving and/or reformatting, using the function `fgl_report_runReportFromProcessLevelDataFile`. However, this function causes the file to be created in addition to the regular processing; for example, a PDF file and a data file can be created at the same time. Or, you could create a completely different layout for one of the files by specifying a different 4rp template.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setRenderingHints`
Specifies the hints to be used in the rendering process.

**Syntax**

```plaintext
fgl_report_setRenderingHints (values om.SaxAttributes)
```

1. *values* - Attribute set containing the key/value pairs.

**Example**

This sample code sets the *svgTextLengthAdjust* and *grvUseEmbeddedFontMetrics* hints:

```plaintext
DEFINE renderingHints om.SaxAttributes
LET renderingHints=om.SaxAttributes.create()
CALL renderingHints.addAttribute("svgTextLengthAdjust","SpacingAndGlyphs")
CALL renderingHints.addAttribute("grvUseEmbeddedFontMetrics","true")

CALL fgl_report_setRenderingHints(renderingHints)
```
For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

Related concepts
Optimize the rendering process on page 715
You can optimize the rendering process by setting rendering hints.

Specifying classic or default style on page 832
The business graphs use a "flat design" by default, but you can change this to use the more 3D "classic" design. You can also choose between three different color schemes.

**fgl_report_setRTFMemoryThreshold**
Set the RTF memory threshold.

**Syntax**

```c
fgl_report_setRTFMemoryThreshold (memoryThreshold INTEGER)
```

1. *memoryThreshold* - The threshold in bytes above which documents are swapped to disk.

**Usage**

Function to set the RTF memory threshold. Applies to both RFT (Microsoft™) and OORFT (Open Office).

In order to prevent exhaustion of main memory when processing large documents, the processor can be instructed to swap parts of the document to a temporary disk file when the document size exceeds this threshold. The default value is set to 10% of the total available heap space. The default heap space is 64Mb.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setSharePortWithGDC**
Configures the report engine to use the same port as the Genero Desktop Client for previewing.

**Syntax**

```c
fgl_report_setSharePortWithGDC (share BOOLEAN)
```

1. *share* - Boolean that, when set to true, specifies that the engine will use FGLSERVER as the client port.

**Usage**

Function that configures the report engine to use the same port as the Genero Desktop Client for previewing.

By default, the port is shared and the port value specified in the FGLSERVER environment variable is used.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setSVGCompression**
Configure SVG compression.

**Syntax**

```c
fgl_report_setSVGCompression(
```
1. **compressOutput** - Boolean specifying whether the output should be compressed. The default value is false.

**Usage**

Function to configure SVG compression.

This function is applicable when SVG output has been selected by a call to the function `fgl_report_selectDevice`. When set, the function causes SVG to be written in compressed form, causing files and streams to be shrunk to about a tenth of their original size. This can benefit the overall performance on slow networks.

The Genero Report Viewer automatically detects if files or streams are compressed and decompresses them on the fly as necessary.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**Syntax**

```plaintext
fgl_report_setSVGCopies(
    copies INTEGER)
```

1. **copies** - the number of copies to print.

**Usage**

Specifies the number of copies to be printed. The default is one (1).

The function applies only if the option **PrintOnNamedPrinter** is chosen in a call to `fgl_report_configureSVGPreview`.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**func**

Function to control the paper orientation.

**Syntax**

```plaintext
fgl_report_setSVGOrientationRequested(
    orientationRequested String)
```

1. **orientationRequested** - Valid values are "landscape" or "portrait".

**Usage**

This function can pre-select (in the case of printing via the print dialog) or set the value of the orientation property.

Normally this value should not be set. The value is set internally by analyzing the page's width and height values. If a page's height is greater than its width, the orientation is portrait, otherwise it is landscape.

However, some printers can be loaded with both portrait and landscape paper. Without the printing software being aware of what type is currently loaded, the wrong orientation setting can cause a misprint. In the case of matrix or label printers which can be loaded with paper of various dimensions (without the printer or the driver being aware
of the current format), the value should be set to "portrait" for landscape reports when the printer contains landscape paper.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setSVGPageRange**
Select which pages should be printed.

**Syntax**

```c
fgl_report_setSVGPageRange(
    fromPage INTEGER,
    toPage INTEGER
)
```

1. *fromPage* - Selects the lower bound of the range of pages to print. The default value is 1.
2. *toPage* - Selects the upper bound of the range of pages to create print. By default, all pages are printed.

**Usage**

Function to select which pages should be printed.

The function applies only if the option `PrintOnNamedPrinter` is chosen in a call to `fgl_report_configureSVGPreview`.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setSVGPaperSource**
Select the input source of the printer.

**Syntax**

```c
fgl_report_setSVGPaperSource(
    paperSource STRING
)
```

1. *paperSource* - one of: Auto, Cassette, Envelope, EnvelopeManual, FormSource, LargeCapacity, LargeFormat, Lower, Middle, Manual, OnlyOne, Tractor, SmallForm. The default value is specific to the printer driver and can typically be selected in the Windows® printer configuration dialog.

**Usage**

Function to select the input source of the printer. Controls what source to take the paper from.

The function applies only if the option `PrintOnNamedPrinter` is chosen in a call to `fgl_report_configureSVGPreview`. The functionality is available only on Windows®.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setSVGPrinterName**
Select a specific printer by name.

**Syntax**

```c
fgl_report_setSVGPrinterName(
    printerName STRING
)
```
1. *printerName* is the name of the printer.

**Usage**

Function to select a specific printer by name.

The function applies only if the option **PrintOnNamedPrinter** is chosen in a call to fgl_report_configureSVGPreview.

To get the printer name, you can use the $GDCDIR/bin/printerinfo tool.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setSVGSheetCollate**

Control the collation of multiple copies.

**Syntax**

```plaintext
fgl_report_setSVGSheetCollate  (  
    sheetCollate  STRING  )
```

1. *sheetCollate* - one of collated|uncollated.
   
   If *sheetCollate* is:
   
   - **collated**, each copy of the document is printed with the pages in sequence.
   - **uncollated**, each page is printed a number of times equal to the value of the fgl_report_setSVGCopies attribute in succession. This is the default.

**Usage**

Function to control the collation of multiple copies.

Specifies whether the pages of the document are to be in sequence, when multiple copies of the document are specified by the fgl_report_setSVGCopies function.

For example, if a document produces two pages as output, fgl_report_setSVGCopies is 6, and sheetCollate is "uncollated", then six copies of the first page are printed followed by six copies of the second page.

**Note:** We discourage setting sheetCollate to "collated" since it requires caching the entire document, which is undesirable for large documents. If necessary, produce the document the required number of times.

This function applies only if the option **PrintOnNamedPrinter** is chosen in a call to fgl_report_configureSVGPreview.

This function fails if the argument is not one of the specified values.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**fgl_report_setXLSMergeCells**

Configure cell merging in XLS (Excel) output.

**Syntax**

```plaintext
fgl_report_setXLSMergeCells  (  
    mergeCells  BOOLEAN  )
```

1. *mergeCells* - Controls the behavior when an item of the report occupies more than one cell. Setting this value causes the cells to be merged into one cell. The value is TRUE by default.
**Usage**

Function to configure cell merging in XLS (Excel) output.

This function is applicable when XLS output has been selected by a call to the function `fgl_report_selectDevice`.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setXLSXMergeCells`
Configure cell merging in XLSX (Excel) output.

**Syntax**

```c
fgl_report_setXLSXMergeCells (mergeCells BOOLEAN)
```

1. `mergeCells` - Controls the behavior when an item of the report occupies more than one cell. Setting this value causes the cells to be merged into one cell. The value is TRUE by default.

**Usage**

Function to configure cell merging in XLSX (Excel) output.

This function is applicable when XLSX output has been selected by a call to the function `fgl_report_selectDevice`.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_stopGraphicalCompatibilityMode`
Restore text-based report output.

**Syntax**

```c
fgl_report_stopGraphicalCompatibilityMode()
```

**Usage**

Function to restore text-based report output.

The graphical compatibility mode switched on by calling `fgl_report_loadCurrentSettings(NULL)` is switched off using this function.

For an example of Genero code using a reporting function, see Using report output functions on page 727. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

**Functions to set report metadata**

Use these functions to set report metadata.

These functions are useful when you want to:

- Set the metadata for compatibility reports and generic reports.
- Localize metadata for .4rp reports.
- Override the metadata properties of a .4rp report. For example, you can make the title dynamic at runtime.
### Table 204: Functions to set report metadata

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fgl_report_setAuthor (author STRING)</code></td>
<td>Function to set the report author metadata value.</td>
</tr>
<tr>
<td><code>fgl_report_setTitle (title STRING)</code></td>
<td>Function to set the report title metadata value.</td>
</tr>
<tr>
<td><code>fgl_report_setCreator (creator STRING)</code></td>
<td>Function to set the report creator metadata value.</td>
</tr>
<tr>
<td><code>fgl_report_setSubject (subject STRING)</code></td>
<td>Function to set the report subject metadata value.</td>
</tr>
<tr>
<td><code>fgl_report_setKeywords (keywords STRING)</code></td>
<td>Function to set the report keywords metadata value.</td>
</tr>
</tbody>
</table>

If the target file format supports metadata, then the value is inserted into the target document. Typically this is a whitespace-separated list of key words. Calling this function to set the value supersedes the value specified in the 4rp template.

**Related concepts**
- [Report Design Document metadata](#) on page 796

The Title, Author, Creator, Subject, and Keywords properties of the document root allow you to add metadata for a report.

**Functions to introspect reports at runtime (librdd)**

Use these functions to access the structures contained in the rdd files.

The module **librdd** (part of the standard reporting library **libgre.42x**) provides programmatic access to the structures contained in rdd files (files produced by the 4GL compiler from 4GL sources with the option `--build-rdd`).

### Table 205: Functions to introspect reports at runtime

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rdd_getEveryRowFields (rddFileName STRING, reportName STRING) RETURNING fieldNames DYNAMIC ARRAY OF STRING</code></td>
<td>Function that loads an rdd file and returns the field names of the fields contained in the first PRINT statement from the ON EVERY ROW section of the specified report.</td>
</tr>
<tr>
<td><code>rdd_loadRddAndGetReportInfos (rddFileName STRING) RETURNING reportInfos DYNAMIC ARRAY OF rddReportInfo</code></td>
<td>Function that loads a rdd file and returns a data structure describing the reports contained in this file.</td>
</tr>
<tr>
<td><code>rdd_debugReportInfos (reportInfos DYNAMIC ARRAY of rddReportInfo)</code></td>
<td>Function that displays the report information contained in a rdd file for debugging purposes.</td>
</tr>
</tbody>
</table>
The source of the module is located in $GREDIR/src/api/librdd.4gl.

*rdd_getEveryRowFields*

Function that loads an rdd file and returns the field names of the fields contained in the first PRINT statement from the ON EVERY ROW section of the specified report.

**Syntax**

```
rdd_getEveryRowFields  (  
  rddFileName STRING,  
  reportName STRING)  
RETURNING fieldNames DYNAMIC ARRAY OF STRING  
```

1.  *rddFileName* STRING - Full path to a rdd file.
2.  *reportName* STRING - Name of the report as specified in the 4gl source file.
3.  Returns *fieldNames* DYNAMIC ARRAY OF STRING - Null if error occurred.

**Usage**

Function that loads an rdd file and returns the field names of the fields contained in the first PRINT statement from the ON EVERY ROW section of the specified report.

Files of type rdd files are produced from 4gl source files by the 4GL compiler using the option "--build-rdd".

*rdd_loadRddAndGetReportInfos*

Function that loads a rdd file and returns a data structure describing the reports contained in this file.

**Syntax**

```
rdd_loadRddAndGetReportInfos  (  
  rddFileName STRING)  
RETURNING reportInfos DYNAMIC ARRAY OF rddReportInfo  
```

1.  *rddFileName* STRING - Full path to a rdd file.
2.  Returns *reportInfos* DYNAMIC ARRAY OF rddReportInfo - Null if error occurred.

**Usage**

Function that loads a rdd file and returns a data structure describing the reports contained in this file.

Files of type rdd files are produced from 4gl source files by the compiler using the option "--build-rdd".

*rdd_debugReportInfos*

Function that displays the report information contained in a rdd file for debugging purposes.

**Syntax**

```
rdd_debugReportInfos  (  
  reportInfos DYNAMIC ARRAY OF rddReportInfo)  
```

1.  *reportInfos* DYNAMIC ARRAY OF rddReportInfo - Structure obtained by a call to *rdd_loadRddAndGetReportInfos*.

**Usage**

Function that displays the report information contained in a rdd file for debugging purposes.

The function displays debugging information for the data structure returned by a call to the function *rdd_loadRddAndGetReportInfos*.
Types used in librdd

These data types are used in the librdd module.

**rddReportPrintElementInfo**

TYPE rddReportPrintElementInfo RECORD
  unionType CHAR(1), #v=variable, e=expression, l=literal
  value STRING, #the value in case of type 'l'
  variableName STRING, #the variable name case of type 'v'
  variableType STRING #the variable or expression type in case of type 'v'
END RECORD

**rddReportPrintInfo**

TYPE rddReportPrintInfo RECORD
  controlPath DYNAMIC ARRAY OF rddTree, # Path into tree, e.g.
  # REPORT/ON EVERY ROW/IF/THEN/FOR,
  # REPORT/ON EVERY ROW/IF/ELSE/FOR
  printxName STRING,
  printElements DYNAMIC ARRAY OF rddReportPrintElementInfo
END RECORD

**rddReportSectionInfo**

TYPE rddReportSectionInfo RECORD
  name STRING, # FIRST PAGE HEADER, BEFORE GROUP customer_id, ON EVERY
  ROW, ...
  reportPrints DYNAMIC ARRAY OF rddReportPrintInfo
END RECORD

**rddReportInfo**

TYPE rddReportInfo RECORD
  reportName STRING,
  reportSections DYNAMIC ARRAY OF rddReportSectionInfo
END RECORD

**rddTree**

TYPE rddTree RECORD
  firstLine INTEGER,
  lastLine INTEGER,
  type CHAR(1), # l: loop: FOR, FOREACH, WHILE
  # c: conditional: CASE, IF, OTHERWHISE, WHEN
  # i: interaction: MENU, {DISPLAY|INPUT} ARRAY, INPUT,
  # PROMPT,
  # a: trigger in interaction or report
  fileId INTEGER, # references rddFiles (rddFile.id=fileId)
  fileIdEnd INTEGER, # references rddFiles (rddFile.id=fileIdEnd) =
  # the file where the declaration ends
  name STRING # for example BEFORE GROUP
END RECORD
Functions to get error details
Use these functions to get the most current error details. The functions are provided as part of the libgre.42x library.

Table 206: Functions to return error details

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fgl_report_getErrorStatus () RETURNING ok INTEGER</td>
<td>Returns the most recent error status.</td>
</tr>
<tr>
<td>fgl_report_getErrorString () RETURNING message STRING</td>
<td>Returns a human-readable error string.</td>
</tr>
</tbody>
</table>

fgl_report_getErrorStatus
Returns the most recent error status.

Syntax

```
fgl_report_getErrorStatus ()
RETURNING ok INTEGER
```

1.  `ok` - TRUE if an error occurred, FALSE otherwise.

Usage
This function returns the most recent error status. The function should be called after calling OUTPUT TO REPORT and FINISH REPORT. Calling the function at any other point in time may return random results. The function returns TRUE in case of errors in GRE.

fgl_report_getErrorString
Returns a human-readable error string.

Syntax

```
fgl_report_getErrorString ()
RETURNING message STRING
```

1.  `message` - An error description if an error occurred, an empty string otherwise.

Usage
This function returns the most recent error string. The function should be called after calling OUTPUT TO REPORT and FINISH REPORT. Calling the function at any other point in time may return random results. In case of an error, the function returns a human readable description of the error.

Pivot table library
The library libpivot.4gl provides functions to retrieve information about pivot tables contained in a .4rp file

- Types defined in the pivot library on page 772
- Pivot library API on page 773
- pivot_debugPivotTables on page 773
- pivot_getLastErrorString on page 773
- pivot_load4rpAndGetPivotTables on page 773
Related concepts

Working with pivot tables on page 836
The pivot table element defines a table element with fixed roles and types for its columns.

Types defined in the pivot library
Three types are defined for the pivot library: pivotHierarchy, pivotMeasure, and pivotPivotTable.

```
PUBLIC TYPE pivotHierarchy RECORD
    name STRING,
    title STRING,
    format STRING,
    value STRING,
    isNumeric STRING,
    enumValues STRING,
    computeTotal STRING,
    computeCount STRING,
    computeDistinctCount STRING,
    computeAverage STRING,
    computeMinimum STRING,
    computeMaximum STRING
END RECORD

PUBLIC TYPE pivotMeasure RECORD
    name STRING,
    title STRING,
    format STRING,
    value STRING,
    isNumeric STRING
END RECORD

PUBLIC TYPE pivotPivotTable RECORD
    name STRING, #PIVOTTABLE
    title STRING, #PIVOTTABLE
    drawAs STRING, #PIVOTTABLE
    computeAggregatesOnInnermostDimension STRING, #PIVOTTABLE
    displayRecurringValues STRING, #PIVOTTABLE
    hierarchiesDisplaySelection STRING, #PIVOTTABLE
    inputOrder STRING, #PIVOTTABLE
    displayFactRows STRING, #FACT
    measuresDisplaySelection STRING, #FACT
    outputOrder STRING, #FACT
    topN INTEGER, #FACT
    hierarchies DYNAMIC ARRAY OF pivotHierarchy,
    measures DYNAMIC ARRAY OF pivotMeasure
END RECORD
```
Pivot library API
Functions from the pivot table library.

Table 207: Pivot library functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTION pivot_debugPivotTables (pivotTables DYNAMIC ARRAY OF pivotPivotTable)</td>
<td>Displays the pivot table information contained in a 4rp file for debugging purposes</td>
</tr>
<tr>
<td>FUNCTION pivot_getLastErrorString () RETURNING errorMessage STRING</td>
<td>Retrieves the error string of the last function called that failed.</td>
</tr>
<tr>
<td>FUNCTION pivot_load4rpAndGetPivotTables (fileName String ) RETURNING status BOOLEAN, pivotTables DYNAMIC ARRAY OF pivotPivotTable</td>
<td>This function loads a 4rp file and returns a data structure describing the pivot tables in this file.</td>
</tr>
</tbody>
</table>

**pivot_debugPivotTables**
Displays the pivot table information contained in a 4rp file for debugging purposes

The function displays debugging information for the data structure returned by a call to the function pivot_load4rpAndGetPivotTables.

**Parameters**
- pivotTables DYNAMIC ARRAY OF pivotPivotTable - Structure obtained by a call to pivot_load4rpAndGetPivotTables

**Related concepts**
- Types defined in the pivot library on page 772
  Three types are defined for the pivot library: pivotHierarchy, pivotMeasure, and pivotPivotTable.
- pivot_load4rpAndGetPivotTables on page 773
  This function loads a 4rp file and returns a data structure describing the pivot tables in this file.

**pivot_getLastErrorString**
Retrieves the error string of the last function called that failed.

**Returns**
- errorMessage STRING - null if no error occurred.

**pivot_load4rpAndGetPivotTables**
This function loads a 4rp file and returns a data structure describing the pivot tables in this file.
Parameters

- **fileName** STRING - Full path to a 4rp file

Returns

- **status** BOOLEAN - FALSE if an error occurred.
- **pivotTables** DYNAMIC ARRAY OF pivotPivotTable

Related concepts

- Types defined in the pivot library on page 772
  Three types are defined for the pivot library: pivotHierarchy, pivotMeasure, and pivotPivotTable.

Create the data schema

For each report program, a corresponding data schema must be created. The data schema then populates the Data View in the Genero Report Designer, providing a hierarchical representation of the data objects to be dragged and dropped into the report design.

Tools have been provided to allow for the generation of a data schema for an application.

Generate a data schema from a Genero BDL report program

After you write or modify a Genero report program, you must generate the data schema (rdd) file. This file is used by the Genero Report Designer to provide a list of data objects for use in the report design.

The data schema (rdd) file is based on the REPORT statement in your Genero report application source file (4gl). This rdd file is used in the report design document (4rd) to populate the Data View, providing details about the fields that will be streamed by the application. The schema contains the list of database columns that make up your data record, as well as grouping details.

Although the data for the report originally may have come from several different data tables, the PRINT statement in your BDL REPORT program block outputs the data as part of a single record. See the Genero Studio >> Report Writer documentation topic "Writing the BDL Program" for more information.

From the command line

Use the --build-rdd command-line option of the fglcomp tool to create a data schema (rdd file). For example:

```bash
fglcomp --build-rdd SimpleReport.4gl
```

The output of this command will be SimpleReport.rdd. The rdd file will be stored in the same location as the 4gl file.

**Note:** If your Genero program contains multiple 4gl files, run fglcomp against the file containing your REPORT program block.

From Genero Studio

Add --build-rdd to the Compiler options property for your Genero source (4gl) file to generate the rdd file automatically each time the 4gl file is compiled. Select the 4gl file listing in the application node of the Projects view to display its properties in the Properties view. The rdd file will be stored in the directory specified in the Target Directory property of the application node that contains the 4gl file.

Related concepts

- Compiler / Runtime configuration (Genero Installations) on page 200
  A Genero Installation contains information about the compiler and runtime version of Genero that will be used by Genero Studio.
- Adding report data (Data view) on page 796
The Data View specifies the structure of the data record for the report.

**Report schema transformations**

A report schema transformation (`.rst`) is a type of schema file that provides access to data from pre-defined sources.

You can use report schema transformations for:

- Random access to data. For example, you print a table of delivery addresses followed by a list of packaging instructions, even though the data stream delivers the packaging instructions before the delivery addresses.
- Enlarged capabilities without programming. For example, the sales data in the data source is grouped by customer and season, but you regroup this data by country and product line.
- Restructuring of data from pre-defined data sources such as SAP-BAPI or Pentaho Kettle.

The report schema transformation contains:

- A reference (r) to an input schema file or a transformed schema file.
- A description of the transformation to apply to the input schema.

Report schema transformations are used in the Report Designer in the same way as other data schemas (.rdd and .xsd). When the transformation is applied, the result displays in the Data View.

**Transformation features**

The features of report schema transformations include:

- Grouping data using variables.
- Sorting data, either before or after grouping.
- Duplication of any part of the document. For example, you can use the same data to draw a chart and a table.
- Shifting record lists and variables. For example, you can shift the order total from the bottom to the top of the list.
- Computing aggregates, such as average, minimum, and maximum.
- Computing running aggregates. These are useful for printing values that depend on page breaks, such as "Carried forward" and "Total until this point".
- Cascading transformations.

**Transformations at runtime**

Report schema transformations are also used in the report engine at runtime. When a report design file (.4rp) is based on a transformed schema file (which references a .rst), then the engine applies the transformation to the data. If r references a transformed schema file, then the process is repeated recursively. The data is always transformed in serial manner, but some transformations (such as sorting) may cause latency.

**Transformations in the demo application**

The OrderReport demo application (Reports.4pw) includes sample report designs that use report schema transformations:

- The `RevenueAnalysis.4rp` report demonstrates duplication of data.
- The `PivotOrderList.4rp` report demonstrates regrouping and aggregations.
- The `FancyPivotOrderList.4rp` report also demonstrates regrouping and aggregations, as well as a running total used to print "Carried forward" and "Total until this point" values.

**Related concepts**

[Adding report data (Data view) on page 796](#)  
The Data View specifies the structure of the data record for the report.

[Options for data operations in reports on page 670](#)
Genero Studio contains several options for performing data operations such as duplication, grouping, and aggregation. Your choice is determined by your requirements, including the type of data source, the flexibility of access, and importance of efficient performance.

**Create a report schema transformation**
You can create a report schema transformation (.rst) from an existing data source (.rdd, an .xsd or another .rst).

1. Select **File > New > Reports > Report Schema > Schema Transformation (.rst)** and click **OK**.

2. On the **Source selection and file location** page, select the input and output files:
   - **Schema Location** - The file name of the schema to transform. This can be an .rdd, an .xsd or .rst file. If the .rdd file contains multiple reports, or the .xsd file contains multiple root elements, you must also disambiguate the selection using the **Schema Root** field.
   - **Schema transformation file path** - The full path and file name of the report schema transformation. You can choose whether to insert the file in the current project.

3. On the **Data building** page, select the data by using the arrows or by dragging and dropping items.
   - **Tip:** You can duplicate information by dragging and dropping it multiple times.

4. Optional: The **Grouping data** page displays the items to group by. To change the default grouping:
   a) In the Results section, select the item you want to change,
   b) In the Group By section, click the + button.
   c) Select the new Group By item and select the field from the drop-down list.
   d) Add more Group By items as required.
   The Repetitions section shows the original grouping order, and the Results section shows the transformed grouping order. You can rename items in the Results section using F2.

5. Optional: On the **Aggregations** page, add computed aggregations to the data using the + button. For each aggregation, select:
   - **Variable** - The variable to aggregate.
   - **Aggregation** - The rule that determines the aggregation type, for example, Sum or Average.
   - **Running** - If selected, the running total is calculated, depending on the aggregation type. For example, the running total or the running maximum.
   - **Sort Key** - Choose Ascending or Descending.

6. Click **Finish**.
You can now attach the report schema transformation to the report design document.

**Duplicate data using a report schema transformation**
You can use the report schema transformation (.rst) to duplicate data in the report.

**About this task**
With other data schemas (.rdd and .xsd), the data is streamed only once, one record at a time. With a report schema transformation, the data can be reused as many times as you like. You can then use the same data in multiple ways, for example, provide a summary at the top and details at the bottom of the report.

1. Create the report schema transformation by selecting **File > New > Reports > Report Schema > Schema Transformation (.rst)** and clicking **OK**.

2. On the Source Selection and File Location page, set the **Schema Location** to the data schema you want to transform, and the **Schema transformation file path** to the file name of your new .rst file. Click **Next**.

3. On the **Data building** page, select one or more fields in the left panel and click the right arrow. The data appears in the right panel under **Report**.

4. Click the right arrow again to add the data as **Report2** in the right panel.

5. Repeat as many times as required. The number is incremented for each subsequent report, **Report3**, **Report4**, and so on.

6. Click **Finish** to create the .rst file.
What to do next

In the report design document (.rst file), attach the report schema transformation as the data source. Add the report objects to contain the data, and rearrange the Report Structure so that the data is displayed as required.

Figure 340: Report Structure with duplicated reports on page 777 shows a sample structure (from the RevenueAnalysis.4rp report in the Reports.4pw demo).

Create a report design document

The Report Design document (.4rp) defines the report page, including the report data and the report elements for organizing and displaying this data.

- GRD 3.10 new features on page 778
- Overview of Genero Report Designer on page 778
- Designing a Report on page 781
- Working with tables on page 819
- Working with business graphs on page 821
- Working with pivot tables on page 836
- Expressions in properties on page 847
- Report Designer Reference on page 859
- Upgrading GRD on page 983

Related concepts

Steps to a Report on page 650
To use Genero Report Writer to create a report, you need to properly prepare your environment, create the report application and report, and run and test the report.

**GRD 3.10 new features**

Genero Report Designer v 3.10 includes information about new features and changes in existing functionality.

**Table 208: Genero Report Designer**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original data produced by an application can now be transformed within the report designer without coding. Transformations include selecting, duplicating, moving, re-ordering, pivotizing and computing aggregations.</td>
<td>See Report schema transformations on page 775.</td>
</tr>
<tr>
<td>The Image Box can now embed the first page of a PDF file.</td>
<td>See Image Box on page 865.</td>
</tr>
<tr>
<td>The new Toolbox object, PDFBOX, can embed an entire PDF.</td>
<td>See PDF Box on page 864.</td>
</tr>
<tr>
<td>You can create a Spider Web chart.</td>
<td>See Category charts and drawAs property.</td>
</tr>
<tr>
<td>The business graphs now use a new “flat” design by default, although you can still specify the “classic” style.</td>
<td>See Specifying classic or default style on page 832.</td>
</tr>
<tr>
<td>If the parent object is a propagating container, the child object does not fit in the remaining space for the parent object, and you set the <strong>Y-Size</strong> property to <strong>rest</strong>, the child now expands to the maximum extent of the parent rather than just the remainder of the parent.</td>
<td>See Y-size property.</td>
</tr>
<tr>
<td>For <strong>X-Size Adjustment</strong> and <strong>Y-Size Adjustment</strong>, a value of <strong>expandToParent</strong> now causes the box to stretch as much as possible without intersecting the borders of a parent or sibling.</td>
<td>See X-Size Adjustment and Y-Size Adjustment properties.</td>
</tr>
<tr>
<td>For business graphs and pivot tables, the new <strong>rangeUpperBound</strong> and <strong>rangeLowerBound</strong> properties define the highest and lowest values on the Y-Axis.</td>
<td>See rangeUpperBound and rangeLowerBound properties.</td>
</tr>
<tr>
<td>For XY charts, the new <strong>domainUpperBound</strong> and <strong>domainLowerBound</strong> properties define the highest and lowest values on the X-Axis.</td>
<td>See domainUpperBound and domainLowerBound properties.</td>
</tr>
<tr>
<td>Report elements now include a <strong>comment</strong> property.</td>
<td>See comment property.</td>
</tr>
</tbody>
</table>

**Overview of Genero Report Designer**

When you launch the Genero Report Designer, you have the tools you need to design Genero reports.

- **Genero reports** on page 779
- **The Report Design window** on page 779
- **The Report output** on page 779
- **The work area** on page 780
- **The Tool Box view** on page 781

**Related concepts**

- **Designing a Report** on page 781
- Design a simple or complex report using the views, menus, and tools in the Report Designer.
- **Create and manage report templates** on page 986
A report template defines the layout of a professionally-designed report that you can use to quickly create an initial report design.

**Genero reports**
Genero reports can take a variety of formats, such as invoices or business graphs.

Genero reports can be:

- General Documents, such as invoices, corporate documents, and accounting reports; you define the contents, page size, page headers and footers, output format, and other attributes.
  
  **Tip:** You can use the report templates to create some of these report formats.

- Pre-printed forms, where you define the content for a pre-printed form.

- Labels, where you define the content and label size to be printed on pages of labels.

- Business Graphs, where you specify the type of graph and the data items to graph.

- Compatibility reports, where you output a legacy Genero (4gl) report in ASCII text, using Genero Report Writer.

A report can be displayed in various output formats.

**The Report Design window**
Use the Report Design window to create a Report Design document.

The *Report Design document* (4rp) defines the report page, including the report data and the report elements for organizing and displaying this data.

When creating a new report, you can begin with:

- An empty report.
- A list report that has a basic structure already in place.
- A report template. See Create a report from an existing template on page 986.

Views and windows provide the tools and work areas for the report. Use the **Window > Views** main menu option to display and hide views:

- The **work area**- main window of the report. When you open a report, this area contains the report page.

- **Structure View** - a tree of the report containers and their contents.

- **Properties View** - a list of the properties for a selected report element. In addition to literal values, **expressions** can be used to change the value of report elements properties.

- **Data View** - a list of the data objects that are available for the report.

- **Toolbox** - a list of the containers that are available.

- The **Output view** - display of messages written to standard output.

- The **Document Errors view** - a list of errors in the opened report design document or template.

- The **Tasks view** - a task manager showing running applications.

**Note:** Metadata for the report design document can be stored in the properties of the report node.

**The Report output**
The report application uses reporting API functions to specify output details. Default output and printer options can be set for each report design document.

To set default paper settings for a report, open the report (4rp) and select **File > Report properties > Paper Settings**. Paper settings include:

- The orientation of the page (portrait or landscape)
- The units of measure for the page (centimeter or inch)
- The page size format (standard or custom) as well as the type of paper (letter, legal, and so on)
- Margins.

To set a default output configuration for a report, open the report (4rp) and select **File > Report properties > Output Configuration**.
• Choose an output format, such as SVG, PDF, or Image. If you choose SVG format, it displays in the Genero Report Viewer. The Genero Report Viewer is provided as part of the Genero Desktop Client. It provides a preview of the report, and allows the user to select a printer. To view a report output as PDF, you must have Acrobat Reader.

• Set rendering defaults.
• Set a default page range.
• If you select Image, you can then set Image Settings to include the file type, the resolution in dpi, and the image prefix name for the created report file.

Related concepts
Designing a Report on page 781
Design a simple or complex report using the views, menus, and tools in the Report Designer.

The work area
The work area in the Main Window provides a GUI interface to the report.

Figure 341: Report Designer work area

You design a report by initially dragging and dropping containers (such as such as Mini Pages and Layout Nodes) from the Tool Box view into the work area, stacking and arranging them to create the report page. Next, report elements such as Word Boxes, Decimal Format Boxes, and Image Boxes are dragged and dropped into the containers.

From the Data View, if you have specified the data schema, you can drag and drop data values into a container.

Page headers and footers can also be defined.

When you select a report element in the work area, its properties are listed in the Properties view. In the Properties view, you can change a property value. For example, a Word Box has a text property where you can enter text to be displayed in the report.

If you select multiple elements, all items in the current selection are affected by the current operation, such as moving, sizing, or changing the type or text.
Use the **View > Toggle View** menu or the Toggle View icon to toggle the work area between the report design and a preview of the report. When you preview a report, sample data is displayed on the page.

**Figure 342: The Toggle View icon**

Zoom buttons on the Toolbar allow you to zoom in and out on the report design document.

**Figure 343: Zoom buttons**

**The Tool Box view**

The Tool Box view provides report elements to place on a report design document.

The Tool Box view, typically displayed as a tab, provides the following report elements:

- Containers - for grouping other elements on the report page; see [Choose the right container](#) on page 784
- Drawables - the report elements contained by a container; see [Choose other report elements](#) on page 785
- Business Graphs - the specific charts and items; see [Working with business graphs](#) on page 821
- References - to define layout-specific elements; see [Choose other report elements](#) on page 785
- Bar Codes - specific bar codes; see [Bar Code type details](#) on page 912

These elements can be dragged and dropped into a report design document.

**Related concepts**

[Report Design Elements (The Toolbox)](#) on page 859

The Toolbox contains report object that can be placed on a report design document.

**Designing a Report**

Design a simple or complex report using the views, menus, and tools in the Report Designer.

- [Launch the Report Designer](#) on page 782
- [Placing elements on the report page](#) on page 782
- [Organizing the report structure (the Report Structure view)](#) on page 790
- [Changing a property value (The Properties view)](#) on page 795
- [Adding report data (Data view)](#) on page 796
- [Add headers and footers to a report](#) on page 792
- [Using page numbers](#) on page 799
- [Print group totals and report totals](#) on page 801
- [Specify different pages](#) on page 803
• Design labels on page 805
• Specify the paper settings of a report on page 807
• Other Report Designer tasks on page 809
• Some tips for legacy report designers on page 815

Related concepts
Overview of Genero Report Designer on page 778
When you launch the Genero Report Designer, you have the tools you need to design Genero reports.

Create and manage report templates on page 986
A report template defines the layout of a professionally-designed report that you can use to quickly create an initial report design.

Launch the Report Designer
Launch the Report Designer from the Genero Studio main menu or the Project Manager.

To open the Report Designer, do one of the following:
• Choose File > New > Reports from the Genero Studio main menu.
  Tip: Before you can use a new report, you must select the data schema (rdd) on the Data View tab.
• Double-click an existing report design document (.rep) in the Project Manager.
• Create a report from an existing template.

Placing elements on the report page
When placing elements on a report page, you determine whether the positioning is absolute or relative by setting element properties that determine how the element acts as the report changes.

• Absolute or relative positioning on page 782
• Drag multiple objects on page 784
• Choose the right container on page 784
• Choose other report elements on page 785
• Modify the sizing policy of containers on page 785
• Modify an object's borders, margins, or padding on page 786
• Align and format numbers on page 789
• Center elements on page 789
• Position elements at the bottom right on page 790

Absolute or relative positioning
The position of a report element can be absolute, to enable reports that use pre-printed forms, or relative, adjusting as needed based on the length of the report element as well as its parents and siblings.

Relative positioning (dynamic layout)
If you CTRL+drag the report element onto the report page (work area), the element is positioned relative to the existing elements. Because the design changes dynamically based on the actual size of the report elements, this is the recommended method for most cases. When you drag the element, a moving red line indicates where the element will be located when you drop it. A green dot on the element indicates its attachment point.

If you select a container and then double-click an element in the Tool Box view or Data View, the element is automatically positioned after the last existing child object in the container.

Relative positioning offers the following advantages:
• The reports adapt to changes of font face, style, and size.
• The report can be localized.
• The paper's size, margins, and orientation can be easily changed.
• The report is easier to edit and maintain.

**Absolute positioning (static layout)**

If you drag and drop report elements on the report page (work area) using the mouse, you can position the element at a specific spot on the virtual grid of a container. This is recommended when you need to match the report design to a pre-printed form. When you drop the element, it snaps to the closest point on the grid. A red dot on the element indicates its attachment point.

An object's absolute position can be measured from the top (for example, 2cm), from the bottom (for example, max−3cm), or relative to height (for example, max*0.4).

As you drag an element, a moving thin black line helps you line it up with other elements on the report if desired. The X and Y properties of the element in the Properties view indicate its location relative to its parent. These are automatically calculated when you drop an element into a container, or move it around. When you move it inside a container, the lines of the container are highlighted in yellow:

![Figure 344: Highlighted container](image)

All elements that are dragged from the Tool Box view or Data View have the floatingBehavior property set to "enclosed", meaning the object will be enclosed in its parent.

- X-Axis and Y-Axis arrows - These indicate the direction of the X-Axis and Y-Axis. On containers, the Y-Axis arrow indicates the filling or layoutDirection of the container. For example, a Stripe lays out its children next to each other left-to-right within the container, and the Y-Axis arrow points to the left. Other containers have the Y-Axis arrow pointing down, as they lay out their children next to each other in a top-to-bottom direction.
- Attachment point - The attachment point at an intersection of the X-Axis and Y-Axis is indicated by a green dot (for relative positioning) or a red dot (for absolute positioning.) If you drag the edge of the element to expand its size, the attachment point remains fixed. You can move the attachment point using the right-click context menu.

Elements on a report have a contextual (right-click) menu of options that allow you to:

- Align elements within a vertical or horizontal container, and move the attachment point on the element.
- Change the width and height of elements.
- Change the focus to a different container or other element.
- Change the object type.
Drag multiple objects
Drag and drop a multiple selection of objects from the Data View to the report page (work area) or the Structure View.

Use the CTRL and SHIFT keys to select the objects, then drag them to the desired location. An object is created for each element selected, following the order in which they were selected.

If you drag to the Design work area using absolute positioning, an additional container is created for each element object. If you have chosen to create a form field object, a horizontal container is used so the elements appear in a line. When you create a column object, a vertical container is used so the elements appear stacked.

If you use relative positioning, or drag to the Structure View, the behavior is the same as if each element was dragged individually.

Related concepts
Absolute or relative positioning on page 782
The position of a report element can be absolute, to enable reports that use pre-printed forms, or relative, adjusting as needed based on the length of the report element as well as its parents and siblings.

Adding data values and captions on page 797
Before you place a data object onto the report design window, click one of the icons on the integrated toolbar to specify whether you wish to drop the item or its title, and whether the object is part of a table.

Choose the right container
Select the right containers to easily manage and organize your reports.

The template for new reports starts with a Page Root in the report structure, which is a Mini Page container expanded to its maximum width and length. Other containers are dropped within the Page Root. Although you could drop all the elements directly on the Page Root, building up the report in blocks of containers allows you to group elements, move the groups, and align the children elements within a parent container.

- A Mini Page is used for the main container of a report page. The default name is PageRoot. The default Layout Direction when you add elements to a Mini Page is top to bottom, down the length of the page. This container propagates: when a report is printed, if a Mini Page fills, a copy is made and the leftover material flows to the copy or copies as needed.
- Use a Layout Node for page headers and footers. A Vertical Box (Layout Node) defines a rectangular area in which the elements are laid out top-to-bottom by default. A Layout Node does not propagate; the contents of headers or footers can not spill into another page. Within the header or footer Layout Node, use Stripe (Mini Page) containers for elements that should be laid out left-to-right across the page. Adding a Stripe to a Layout Node automatically extends the Layout Node across the page.
- Use a Stripe (Mini Page) container for table rows. A Stripe (Mini Page) container is a Mini Page with the Y-Size set to max, so it stretches across the report page. Items added to a Stripe are laid out left-to-right. If the elements within a Stripe exceed the page width, the row is broken into the next line.
- Use a Mini Page for a report page with a different layout, such as a different first page.

Related concepts
Propagating Containers (Mini Pages) on page 861
The Propagating Containers section of the toolbox contains those containers that propagate. If the container fills, a copy is generated and the extra content overflows to the copy.

Simple Containers on page 860
The Simple Containers section of the toolbox contains those containers that do not propagate. These containers are used to group and organize objects on a page.

Related tasks
Add headers and footers to a report on page 792
To add a page header or footer to a report, create a simple container and set the **Section** property.

**Choose other report elements**

After placing a container on a report page, add the report elements that hold the data and determine how the report looks.

**Data View**

Icons at the top of the **Data View** specify the type of object created when you drag and drop a data value. See **Adding data values and captions** on page 797.

**Tool Box view**

From the Tool Box view:

- For additional text - use a **Word Box** or **Word Wrap Box**. The **text** property specifies the content. You can edit the text in the Properties view or by double-clicking on the text in the Report Designer work area.
- For Numeric data - use a **Decimal Format Box**, which makes it possible to parse and format numbers in any locale. The **value** property specifies the number. You can edit the value expression in the Properties view or by double-clicking on the Decimal Format Box in the Report Designer work area. You can define the printed format, including decimal places, by setting the value of the **format** property.
- For HTML pages - use an **HTML Box** on page 862, which displays the image of an HTML document in a report.
- For Images - use an **Image Box**. The **location** property specifies the image to be printed.
- For PDF files - use an **PDF Box**. The **location** property specifies the PDF to be printed, and you can also specify the **page ranges** to display.
- For page numbers - use a **Page Number Box** to automatically display the correct page number for each report page.
- For tables - use a **Table** on page 867 to set up an object that contains columns and rows to display rows of data.
- For business graphs - choose the appropriate **Business Graphs** on page 869 object (chart or pivot table) for the specific type of graph. See **Working with business graphs** on page 821.
- For values such as "Total from previous pages" or "Totals until this point" - use an **InfoNode** and **Reference Box**. These two elements work together to enable this type of content. See **Print a Layout-dependent reference (InfoNodes)** on page 811.

**Modify the sizing policy of containers**

When as report element is in focus, arrow-shaped controls appear on the four sizing knobs located at the center of each side. These controls allow you to view and modify the sizing policy of a container.

Arrows pointing inward indicate a shrinking sizing policy:

- X-Size="min" and Y-Size="min" 
  - X-Size-Adjustment="shrinkToChildren" and Y-Size-Adjustment="shrinkToChildren"

Arrows pointing outward indicate a maximizing policy:

- X-Size="max" and Y-Size="max" 
  - X-Size-Adjustment="expandToParent" and Y-Size-Adjustment="expandToParent"

Click on an arrow to toggle its value.

**Examples**

These images illustrate some common cases.
A Page Root container typically expands in all directions to use up the available space.

Modify an object’s borders, margins, or padding
Any box object on a report design document can have margins, borders, and padding.
Set an object's **specific properties in the Properties View** to change:

- the width of a margin, border, or padding - Margin width properties, Border width properties, Padding width properties
- the style of a border (solid, dashed, double, dotted, groove, ridge, inset, outset) and whether the box will have rounded corners Border style properties
- the color of a border - Border color properties

Borders are **drawn outside the box** and will increase the actual size of the box beyond the value specified in x-Size and y-Size.

When a bordered item is positioned it behaves like a regular element, so that the attachment point appears at the specified position.

**Illustrations**

![Illustration of borders with different styles and widths](image)

**Figure 349: Examples of borders**
Figure 350: Examples of borders

Related concepts
Properties related to margins and borders on page 908
These are the margin and border-related properties for report elements.

Size expressions for bordered boxes
Define the outer bounds of a box by using the X-Size and Y-Size properties.

The X-Size and Y-Size properties specify the inner size of the box. For example, if a box is 3cm wide and has a 1mm thick border on all sides, the box's outer bounds appear to be 3.2cm wide. This conforms to the CSS specification.

You can define the outer bounds of a box instead:

- Determine the X-Size and Y-Size values by subtracting the width of the borders from the desired height and width. For example, if you want a box to be 3cm wide on the outside while having 1mm borders on all sides, calculate the width to be 3cm-2mm=2.8cm wide.
- If you want a box to have the same size as its parent, set both the X-Size and Y-Size properties to the value max. You do not have to subtract the borders, since the system automatically adjusts the value of max in cases where the box has borders. For example, if the box has a 1mm border and is contained in a box that is 3cm high and wide, the outer bounds of the contained box are also 3cm.

Important: For bordered boxes, do not use expressions that contain max as only one of its components, such as max/2. See following section for details.

Size expressions that contain the variable max with other components
Expressions such as max-2cm or max/2 can have unexpected results when you specify the height and width of bordered boxes. Modify these expressions as follows:

1. Take the original formula and replace any occurrence of max with (max+borders+padding+margin), where "borders", "padding", and "margin" denote the width values for each on both sides of the box.
2. Take the resulting formula (which we'll call \( f \)) from step 1, and create the final formula as \( f \)-borders-padding-margin.

For example, the original properties box are:

- **X-Size** = \( \text{max}/2 \)
- **leftBorderWidth** = 2mm
- **leftMargin** = 1mm
- **rightBorderWidth** = 1.5mm
- **rightPadding** = 3mm

X-Size is then changed in the following steps:

1. **X-Size** = \( \frac{\text{max}+2\text{mm}+1\text{mm}+1.5\text{mm}+3\text{mm}}{2} \); this is \( f \) in the explanation.
2. **X-Size** = \( \frac{\text{max}+2\text{mm}+1\text{mm}+1.5\text{mm}+3\text{mm}}{2} - 2\text{mm} - 1\text{mm} - 1.5\text{mm} - 3\text{mm} \), which can be consolidated to \( \frac{(\text{max}-7.5\text{mm})}{2} - 7.5\text{mm} \).

**Align and format numbers**

Use proper containers and properties to align and format numbers.

Use the ToolBox object **Decimal Format Box** as the report element for numbers. It supports different locales and kinds of numbers, including integers, fixed-point numbers, and currency amounts ($123).

The text property specifies the value for the number to be printed. The value of this property may also be edited directly in the report design document by double-clicking the Decimal Format Box. The input cursor will be placed in the document, and the layout of the document is updated on each keystroke.

The format property specifies how the number will print out. The default value for this property is "---,---,---&.&&". You can change this string, using the specified symbols. The - (minus) symbol represent digits (if the number is negative, it will print with a leading - ), and the period represents the decimal point. The & symbol fills with zeros any position that would otherwise be blank. If the actual number displayed requires fewer characters than the format string specifies, numbers are right-aligned and padded on the left with blanks.

The alignment property specifies how the number is aligned, for example, baseline or centered. It controls the x position of this report element in its parent container, unless you have set the x property explicitly.

**Table 209: Formatting examples**

<table>
<thead>
<tr>
<th>Value of text property</th>
<th>Appearance in report</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456.1</td>
<td>123,456.10</td>
</tr>
<tr>
<td>15.24</td>
<td>15.24</td>
</tr>
<tr>
<td>-1600</td>
<td>-1,600.00</td>
</tr>
</tbody>
</table>

**Related concepts**

**Using a page number string** on page 800

For Page Number containers, use the **Text Expression** property to create a page number string, such as the standard "Page N of M".

**Center elements**

Use properties to center an element in its parent.

To center an element in its parent container you can set its properties as described in **Table 210: Centering elements** on page 789.

**Table 210: Centering elements**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>max/2</td>
</tr>
</tbody>
</table>
An X and Y value of max/2 sets the x and y coordinates of the element to the maximum of its parent container divided by 2. An Anchor X and Anchor Y value of 0.5 sets the attachment point to the center of the element.

**Position elements at the bottom right**

Use properties to position an element at the bottom right of its parent.

To position an element at the bottom right of its parent container, you set its properties as described in Table 211: Positioning elements at the bottom right on page 790

**Table 211: Positioning elements at the bottom right**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>max</td>
</tr>
<tr>
<td>y</td>
<td>max</td>
</tr>
<tr>
<td>anchorX</td>
<td>1</td>
</tr>
<tr>
<td>anchorY</td>
<td>1</td>
</tr>
</tbody>
</table>

**Organizing the report structure (the Report Structure view)**

You use containers to make sure that the report elements print correctly.

- The tree structure on page 790
- Triggers on page 791
- Place a trigger within the report structure on page 792
- Add headers and footers to a report on page 792
- Switch child and parent nodes on page 793
- Example report structure on page 794

By combining horizontal boxes, stripes, word boxes, and other elements, in the correct order in the parent container, you control the order in which these elements print. Then you can move the containers in the Structure View to make sure they print in the correct place.

**The tree structure**

The Structure View is a tree containing object and trigger nodes.

**object nodes**

In the Structure View, the object nodes are the containers and objects that are to be printed on the report output.

**trigger nodes**

In the Structure View, the trigger nodes are the triggers (event handlers) that specify when the object nodes are to be printed out.

Change the hierarchy of the objects and triggers in the Structure View by dragging and dropping them within the tree:

- Drag a container or trigger and drop it on a different node; it will become a child of that node.
- Press the ALT key and drag a container or trigger, dropping it on a different node; it will become the parent of that node.
Triggers
The report data is passed from your report program to the report one row at a time, sorted by the criteria specified in your report. The triggers in the report structure specify what should be printed when a change in data occurs.

You may want to print some of the data from each row received; this is generally the report body. Other data and text should only print when a change in a specific data item takes place. For example, you may want to print a total each time the customer_id value changes.

Arrange all the report elements for a single trigger in a parent container. The example report structure illustrates how the triggers appear in the Structure view.

How triggers work
Triggers are defined in the Structure view of the report design document and in the BDL code. For a Genero BDL program, the SQL statement defines the sort criteria.

Each row streamed to the report contains all the data or text specified by the data schema (the data report file). The data schema feeds this information to the report structure.

Triggers and Genero BDL report applications
Trigger nodes are specified by the ORDER EXTERNAL BY statement in the REPORT program block in your Genero BDL code, and indicate the data values by which the data is grouped. The final trigger node is ON EVERY ROW, specifying what is to be printed for each row of data passed to the report. The trigger nodes display in the Report Structure as red bullets:

Figure 351: The ORDER BY clause and report triggers

When the Data Schema changes
When the data schema associated with a report is modified, the Genero Report Designer regenerates the triggers to match the new schema. The Genero Report Designer attempts to update the Report Structure using the minimum number of modifications required to perform the update. Cases of adding new triggers and cases of removing triggers that do not contain document fragments are handled automatically.

When it is necessary to move or remove a trigger that contains a document fragment, a warning displays in the Document Errors view stating that the fragment may require manual correction. These warnings are stored persistently in the report definition (.4rp) file. The warning can only be removed by using the context menu Clear trigger update message (issue is fixed).

The warning message Data schema changed. The document has been updated to match the changes is only displayed when a new trigger is inserted, a trigger is removed, or a trigger is moved.

Related concepts
Adding report data (Data view) on page 796
The Data View specifies the structure of the data record for the report.

**Place a trigger within the report structure**
Organize the report triggers that appear in the report structure after you select the data schema.

You can organize your report structure using drag-and-drop within the Report Structure view. Alternatively, you can use the **Repeat selected items on** contextual menu to easily make a trigger the parent of a document node.

1. Identify the node in the report structure that you wish to be the child of a specific trigger node.
2. Right-click the node and select **Repeat selected items on**.
   The list of triggers appear in a sub-menu.
3. Select the trigger.

The trigger becomes the parent of the currently selected document fragment. The structure of the document tree is preserved.

**Note:** A similar functionality is available by holding the ALT key while dropping a trigger on top of a document node.

**Add headers and footers to a report**
To add a page header or footer to a report, create a simple container and set the **Section** property.

**About this task**
Placement of the header and footer containers take the page size into account. When a report page is printed, any page header prints at the top of the page, followed by the content of the containers associated with trigger nodes, followed by any page footer at the bottom of page, in accordance with the **Page Size** set for the report. This is the basic design of the report page, which is repeated when the report is run for as many pages as are required, based on the data passed to the Report Writer.

1. Open or create a report.
2. Add the header or footer container to a **MiniPage** or **Page Root** of the report.
   You can use any simple container or drawable, but a **Vertical Box (LayoutNode)** is most commonly used.
3. Set the container's **Section** (also called **port**) property to one of the header or footer values.
   For example, a Vertical Box with the **Section** property set to the firstPageHeader section will print as the header on the first page of the report. For a full list of the options, see **Section**. In the parent Container, you cannot have multiple header or footer containers set to the same section.
4. If you use a Vertical Box for the header or footer container, set the Layout Node's **X-size** property to `max`, and its **Y-Size** property to `min`.
5. Ensure that the header and footer nodes are correctly placed in the Report Structure.

**Important:** If the **Section** property is specified, the header or footer must be the elder sibling of any node without the **Section** property specified. The exception is the Last Page Footer, which must come after the other nodes. See Figure 352: Headers and Footers in Structure View on page 793.
6. Build up the header or footer using various containers and report elements.

   For example:
   
   • **Stripes** are useful for report elements that are to be laid out left to right across the page.
   • **Word Boxes** and **Image Boxes** display content within the container in the order in which it should be printed.
   • **Page Number Boxes** provide runtime information about the pages of the report.

   **Note:** The properties **HidePageHeaderOnLastPage** and **HidePageFooterOnLastPage** provide flexibility in the printout.

**Switch child and parent nodes**

In the Report Structure, you have two nodes in a parent-child hierarchy. You can make the child the parent, and the parent the child.

While holding the ALT key, drag the child and drop on to the parent.

This example starts with the WordBox being a child of the Orders_orderid trigger node:

![Figure 353: WordBox is child of Orders_orderid trigger](image)

While holding the ALT key, click on WordBox, then drag and drop it on top of Orders_orderid.

![Figure 354: Orders_orderid trigger is child of WordBox](image)

The two nodes have effectively switched places.
Example report structure
This example demonstrates how you can use the Structure View to examine and modify the structure of your report.

![Structure View and Properties of a sample report](image)

**Figure 355: Structure View and Properties of a sample report**

The Page (Page Root container)
- Page Header: In the example shown, the Page Header is a Layout Node container that specifies what is to be printed as a Page Header. See Add headers and footers to a report on page 792. The example Page Header contains:
  - a Stripe container
  - a Word Box named Report Title
  - a Page Number box
• The example data-related triggers Group userid, Group orderid, and Group linenum do not cause anything to print as they have no child containers.
• The example OnEveryRow trigger has a Stripe containing a Decimal Text Box and a Word Box. These are the data items that will be printed for every data row that was passed to the report.

**Changing a property value (The Properties view)**
Select a report element in the report page (work area) or Structure View to display and edit the property values in the Properties View.

![Properties View](image)

**Figure 356: Properties View**

The values for the properties of a report element can be changed by typing the new value in the Value column. The value may be a literal value, or it may be an expression written using the RTL Expression Language. For example, the appearance can be changed conditionally, by creating an expression to turn the background color red if the value for the form element is greater than 1000. All the properties are assigned a type, and the values entered must be valid for that type.

**Note:** For WordBoxes, WordWrapBoxes, and DecimalFormatBoxes, if the text property is a literal value it may be edited directly in the report design document. Double-clicking on the element selects the text and places the input cursor in the document. The layout of the document is updated on each keystroke.

**Related concepts**

- **Report Element Properties** on page 876
  Each element in a report has associated properties. Values for these properties may be literal, or they may be RTL expressions.

- **Expressions in properties** on page 847
  A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

- **RTL Class Reference** on page 945
Reference information for Report Template Language (RTL) classes.

**Report Design Document metadata**
The Title, Author, Creator, Subject, and Keywords properties of the document root allow you to add metadata for a report.

Select the document root in the Structure View and enter your values in the Properties View.

![Figure 357: Report Design metadata](image)

The metadata is inserted into the final document (PDF, SVG) if the format supports metadata. In the case of SVG, the **title** property is used as a document caption in the Genero Report Viewer.

If your report runs in compatibility mode (having no `4rp` design document), the values can be set by calling the **report API functions for metadata**.

The "Keywords" property does not currently work for "xlsx" format.

**Related concepts**
[Properties for Report Metadata](#) on page 910

Report Metadata properties can be set in the report design document (`.4rp`).

**Adding report data (Data view)**
The Data View specifies the structure of the data record for the report.

The structure of the data record is defined by the input data schema file. The data schema file is extracted from the application source files:

- For applications written in Genero BDL, an `.rdd` file is extracted.
- For arbitrary XML data sources, you can generate an `.xsd` file to describe the data schema. See [Support for arbitrary XML data sources](#) on page 798.
For existing data sources, you can use a report schema transformation.

When a data schema has been selected in the Data View, the available data objects are shown. The data schema is only used during report development. At runtime, the data is sent to the report by the report application.

Within the Data View:

- The Arrows icon allows you to sort the data items alphabetically.
- Values in the Sample Data column display when you preview a report. Double-click a value to edit or replace it.
- The Filter Fields By Name field, located at the bottom of the Data View, allows you to specify filtering criteria for the Data View, where only fields containing the name entered in the box are displayed in the data items.

![Data View](image)

**Figure 358: Data View**

Click the Open Data Report File icon at the top of the Data View to specify the data schema file to populate the Data View.

**Adding data values and captions**

Before you place a data object onto the report design window, click one of the icons on the integrated toolbar to specify whether you wish to drop the item or its title, and whether the object is part of a table.

Dropping the object as a simple report field:

- Form field value object
- Form field title object - Places the caption for the selected object. Use for field labels.

Dropping the object aligned as part of a table (the space allocated for the column will be the larger of the space required for the data or the title, helping to align the title and data in the columns of a table):

- Table column value object
- Table column title object - Places the caption for the selected object. Use for table column headers.
- Table column value object for a column without a title - if you are not going to have a column header, the space allocated for the column is set to the maximum required by the value only.

Allow the Report Designer to determine the type of dropped object.

- Create element based on the document context

  The object type created for a field is determined by the location in the document. Consider dragging a numeric field to two different locations in a document. In the first instance, the object is dropped into the OnEveryRow...
stripe and it becomes a Decimal Format Box. In the second instance, the object is dropped onto a Map Chart and it becomes a chart Item element.

Table 212: Rules governing element creation based on context

<table>
<thead>
<tr>
<th>Element</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITEM (&quot;key&quot; is set if field isn’t numeric, “value” otherwise)</td>
<td>Parent element is a MAPCHART</td>
</tr>
<tr>
<td>CATEGORYITEM (&quot;key&quot; is set if field isn’t numeric, “value” otherwise)</td>
<td>Parent element is a CATEGORYCHART</td>
</tr>
<tr>
<td>Same object as option “Create a table column title object”</td>
<td>Parent element is of class grwTableHeader</td>
</tr>
<tr>
<td>Same object as option “Create a table column value object”</td>
<td>Parent element is of class grwTableRow</td>
</tr>
<tr>
<td>Same object as option “Create a table column value object for a column without title”</td>
<td>Parent element is of class grwHeadlessTableRow</td>
</tr>
<tr>
<td>Same object as option “Create a form fields value object”</td>
<td>If none of the above are applicable.</td>
</tr>
</tbody>
</table>

The data objects are automatically contained in a Word Box if the data type is defined as less than 30 CHAR, and in Wordwrap Boxes if the data type is defined as larger than 30 CHAR. If the data type is Numeric, the data object is contained in a Decimal Format Box.

The text property of the Word Box or Word Wrap Box, or the value property of the Decimal Format Box, specify what will print in the report output. The value property of the Decimal Format Box can be calculated using an Expression. See Using RTL Expressions.

Support for arbitrary XML data sources

Genero Report Writer can produce reports from arbitrary XML input sources.

The xml source is described by an XML Schema. The Open schema file dialog proposes rdd and xsd file formats.

If an xsd file is selected, the designer interprets the file as follows:

- Any XML Attribute is considered a variable.
- Any simple type element with minOccurs=1 and maxOccurs=1 is considered a variable.
- Any complex type elements with minOccurs=0 or maxOccurs>1 produce triggers.

For an example, see the Table.4rp report design document in the Reports sample project, located under the OrderReportXML application node. Open this report to see the OrderData.xsd used as data schema for the report (as shown in the Data View tab).

Encoding null values in the data

The attribute “xsi:nil” (with “xsi” representing the namespace “http://www.w3.org/2001/XMLSchema-instance”) can be used on an empty element to denote a null value. The RTL function isNull() for input variables will return true for such a variable. For example, consider the following document fragment:

```xml
<input xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...<productReference xsi:nil="true"></productReference>
```

The RTL expression productReference.isNull() will yield true for this instance of the variable.

Limitations

Recursive references are not supported. Elements involved in a recursive reference will be ignored by the designer.
Optional variables are not supported. A variable is optional if one of these conditions is true:

- It is an XML simple type element with minOccurs="0".
- It is an XML Attribute without use="required" in the schema definition.

Simple type element with minOccurs="0" are discarded by the designer. No variable is created.

The designer creates variables for optional attributes and issues a warning. An error occurs if the variable is not present at runtime.

The Genero application

When you declare the data of a report design document (.4rp) to use an arbitrary XML data sources, you must use the fgl_report_runFromXML API function in your report. See fgl_report_runFromXML on page 742.

Using page numbers

Use the Page Number drawable to print page numbers on a physical page.

The Page Number drawable (PageNoBox) is a layout container that prints the page number of the physical page. It is frequently part of a page footer. The drawable prints only the page number (a numeric) by default, but you can use the Text Expression property (textExpression) to create a page number string, such as the standard "Page N of M". See Using a page number string.

Use the Page Number container properties Offset (pageNoOffset), Format (pageNoFormat), and Text Alignment (textAlignment) to format the page.

Use the Name (pageName) property if you want to reset the page number each time a specific report trigger fires. Select a Mini Page under the report trigger to use as a basis for the page number count. You can add a new Mini Page for this purpose if necessary.

Figure 359: Page Number Box properties

Setting the length of a page number

Use properties to change the default length of a Page Number container and ensure that enough space is available to display the page number string.

For performance and latency reasons, the Genero Report Engine (GRE) lays out the pages of the report without waiting to know the total number of pages. Instead, when the GRE has acquired all other information for the page, it computes the layout and allocates a default size of four digits for the page number string. This length is sufficient for most needs, but in some cases you might want to change the default:

- If the report exceeds 9999 pages, the default size will not be large enough for the actual text.
- If the report does not exceed 100 pages, you might want to minimize the space it occupies.
For these cases, you can set the allocated size of the Page Number container by setting one or more of the following properties:

1. **X-Size**: Set the length explicitly, for example, 3cm.
2. **Text**: Set to the maximum number of digits you want. For example, set Text to 000 to specify a maximum length of three digits.
3. **Text Expression**: Set the PXXML string expression value. See Using a page number string on page 800.
4. **Code Value Expression**: Set the PXXML string expression value for the bar code.

If multiple sizing properties are configured, Genero Report Writer uses the property with the highest priority (listed above from highest to lowest priority).

If none of these properties are set, Genero Report Writer calculates the length of the Page Number container to contain a maximum of four digits.

**Using a page number string**

For Page Number containers, use the **Text Expression** property to create a page number string, such as the standard "Page N of M".

If you set the **Text Expression** property, it is important to verify that the Page Number container length is long enough to print the full page number text. The size of the string depends in part on the number of page breaks in the report. For example, the string "Page 2 of 5" has a smaller length than "Page 2 of 76".

Because of the unpredictable variation in size, Genero Report Writer defaults to a container size based on a four digit current page and total number of pages ("Page 9999 of 9999"). To override the default behavior you can set the **X-Size** to an explicit size or specify a smaller size with the **Text** property (for example "Page 99 of 99").

The values of the **Name** (pageNumber), **Offset** (pageNoOffset), and **Format** (pageNoFormat) properties are ignored when the **Text Expression** property is set.

These functions can be used to format and access specific page numbers and totals.

- **Class String**: format(Numeric number, Enum format) - formats the number as specified. The value for the format parameter can be ARABIC, LOWERROMAN or UPPERROMAN.
- **Class Numeric**: getPhysicalPageNumber() - gets the current page number of the physical page.
- **Class Numeric**: getTotalNumberOfPhysicalPages() - gets the total number of physical pages.
- **Class Numeric**: getPageNumber(String pageName) - gets the page number of the specified page.
- **Class Numeric**: getTotalNumberOfPages(String pageName) - gets the total number of pages for the specified page.

**Note**: If you use functions getTotalNumberOfPhysicalPages() or getTotalNumberOfPages() and if printing is initiated from the viewer, report pages waiting to be updated with the actual page count will be held back.

**Examples**

This expression computes the string "Page N of M" for the physical pages. The equivalent of the **Offset** property can be achieved by doing arithmetic with the results from the page number functions. In this case, the numbering starts at page 11 since the example formula adds 10 to the value returned from the function getPhysicalPageNumber();

\[
\text{"Page } +\text{format(getPhysicalPageNumber(),ARABIC)}\text{+" of } +\text{format(getTotalNumberOfPhysicalPages(),ARABIC)}
\]

This expression computes the string "Page N of M" for logical pages, providing page numbers for each order within a batch of several orders:

\[
\text{"Page } +\text{format(getPageNumber("pageRoot"),ARABIC)}\text{+" of } +\text{format(getTotalNumberOfPages("pageRoot"),ARABIC)}
\]
Related concepts
Align and format numbers on page 789
Use proper containers and properties to align and format numbers.

Display an invoice page number
Print the number of the invoice page instead of the physical page number.

About this task: A report can contain a batch of invoices with several pages in each invoice.

1. Use the name property of the Mini Page container of the invoice to assign a name to the page.
2. Add a Page Number report object to the page header or footer of this Mini Page container.
3. In the Page Number section of the properties for the Page Number object, set the name property to the name of the Mini Page Container.

Note: Alternatively, you can use a page number string.

The invoice page number is printed on the report. For each invoice, the page count starts again at 1.

Example
The report includes four invoices, sorted by customer.

Table 213: Physical versus invoice page numbers

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Number of pages in invoice</th>
<th>Physical page numbers</th>
<th>Invoice page numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jean Dupont</td>
<td>5</td>
<td>1 to 5</td>
<td>1 to 5</td>
</tr>
<tr>
<td>Pedro Garcia</td>
<td>3</td>
<td>6 to 8</td>
<td>1 to 3</td>
</tr>
<tr>
<td>Steve Miller</td>
<td>2</td>
<td>9 to 10</td>
<td>1 to 2</td>
</tr>
<tr>
<td>Laura Kelly</td>
<td>4</td>
<td>11 to 14</td>
<td>1 to 4</td>
</tr>
</tbody>
</table>

Print group totals and report totals
Print totals on your report to aggregate numerical information.

Define report totals (Genero BDL)
In your Genero BDL report application, define variables for the totals, calculate the values, and output them to the report engine.

1. Open the .4gl file.
2. In the REPORT program block, define variables:

```plaintext
DEFINE
  data store_order_data,
  --Add variables for totals
  store_total, order_total, item_total, report_total DECIMAL(10,2)
```

3. Identify the data columns by which the data is grouped in the ORDER EXTERNAL command:

```plaintext
ORDER EXTERNAL BY
  data.orders.store_num,
  data.orders.order_num
```

4. Calculate the group and report totals, and output them in the FORMAT section.

Tip: Restrict your use of control blocks to FIRST PAGE HEADER, BEFORE GROUP OF, ON EVERY ROW, AFTER GROUP OF, and ON LAST ROW.

```plaintext
FORMAT
  --Add FIRST PAGE HEADER, BEFORE GROUP OFs
FIRST PAGE HEADER
LET report_total = 0

BEFORE GROUP OF data.orders.store_num
LET store_total = 0

BEFORE GROUP OF data.orders.order_num
LET order_total = 0

ON EVERY ROW
   --Add statements to calculate totals
   LET item_total = data.items.price*data.items.quantity
   LET order_total = order_total + item_total
   LET store_total = store_total + item_total
   LET report_total = report_total + item_total
PRINT data.*, order_total, store_total, report_total

Related concepts
Creating a simple report on page 674
This example BDL program (SimpleReport.4gl) uses data from the officestore database to create a report.
There are more complex examples in the Reports project provided as a demo with Genero Report Writer.

Display totals on a report
In the Report Designer, add group and report totals using the Stripe container.

1. Open the report design (.4rp) file.
2. In the Report Design window, define Stripes in the design document, one for each total.
3. Use Tool Box elements to hold the values, for example, Decimal Format Boxes.
4. In the Structure View, drag each Stripe and drop it onto the trigger node for the corresponding group trigger. This will position the Stripe as a child of the trigger node and display it at the end of the group. It is important that the total stripes are the last child element of a report trigger node.

Example
In this example, the order_total Stripe is the last child of the Group order_num, the store_total Stripe is the last child of the Group store_num. The report_total Stripe was dropped onto the Page Root, the main page for the report. This positions the Stripe as a child of the Page Root, and places it at the very bottom of the child list for that trigger.

Figure 360: Example Report Structure
On each change of data, the specified data for every corresponding row will print, and the appropriate Stripe will print after each change of Group. The report_total Stripe will be the last thing to print out on the report. The values of any
data objects in the Stripe are taken from the immediately preceding row of data (the last report line of the container for onEveryRow trigger).

**Print totals at the beginning of a report**
Print aggregate values before the first detail row is printed.

For example, totals can be printed at the beginning for a report.

Two methods exist:

- Use an InfoNode and a Reference Box to hold the totals. See also Print a Layout-dependent reference (InfoNodes) on page 811.
  
  When this feature is used, the output must be delayed until the input has been processed to the point where the variable value is shipped. In the case of a grand total, which is shipped at the end of the report, the entire input must be consumed before the document fragment containing the total is output. If the total number of records is small, the delay is hardly noticeable, for example, when you print the order total before printing up to a few hundred rows relating to the order.

- Use a report schema transformation to shift the total from the bottom to the top of the list.

**Specify different pages**
A report might contain front and back matter, or it might display different content on the reverse side of the page.

**Specify different first and last pages**
Add additional Page Root containers to serve as different first and last pages in a report.

**About this task**
A typical report starts with a single Page Root container, where the streaming data is formatted and displayed. To add additional pages to the report, you add and position additional Page Root containers. For example, you can add an upfront advertisement at the start of a report, or add terms and conditions at the end of a report.

1. In your report design document, add Page Root containers.
2. In the Structure view, place the Page Root containers in the position that you want them to appear in the report, before or after the main Page Root.
3. Label the Page Root containers to distinguish their functions.
   
   For example:
   - first-page - This is a "before" page, to print before the main content.
   - main-report-page - This contains the triggers and containers that make up the body of the report.
   - last-page - This is an "after" page, to print after the main content.
4. Add content to the pages as required.

**Specify a different reverse side page**
Specify content that prints on the reverse side of each report page.

**About this task**
The reverse-side content is contained in a page header or footer.

1. In your report, create a page header or footer under the Page Root.
2. Select the container for the page header or footer and enter the following values in the Properties View:
   - **Section**: evenPageHeader or evenPageFooter
   - **X-size**: max
   - **Y-Size**: max
3. Add other content to the page as required.
4. If you want to print a reverse side after the last page, define an additional reverse side page at the end of the report.

If the report includes four pages and the reverse side ‘B’ is printed between the pages, the print sequence is 1, B, 2, B, 3, B, 4.
If you define an additional reverse side page at the end of the report, the print sequence is 1, B, 2, B, 3, B, 4, B.

**Example**

You want to print a legal notice on the reverse side of each page of your report, including the last page. The page header node is called **TandC**, for "Terms and Conditions". The child node of the page header is a HTML box that brings in the legal text. To include the legal notice at the end of the report, an additional page called **TandC_last** is created to include the same HTML box.

**Figure 361: Reverse side pages in Structure and Properties View**

<table>
<thead>
<tr>
<th>Report Structure</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Order List Group" /></td>
<td><img src="#" alt="Order List Group" /></td>
</tr>
<tr>
<td><img src="#" alt="Page Root" /></td>
<td><img src="#" alt="Page Root" /></td>
</tr>
<tr>
<td><img src="#" alt="Page Header" /></td>
<td><img src="#" alt="Page Header" /></td>
</tr>
<tr>
<td><img src="#" alt="HTMLBox" /></td>
<td><img src="#" alt="HTMLBox" /></td>
</tr>
<tr>
<td><img src="#" alt="Group user id [0; unbounded]" /></td>
<td><img src="#" alt="Group user id [0; unbounded]" /></td>
</tr>
<tr>
<td><img src="#" alt="TandC_last" /></td>
<td><img src="#" alt="TandC_last" /></td>
</tr>
<tr>
<td><img src="#" alt="HTMLBox2" /></td>
<td><img src="#" alt="HTMLBox2" /></td>
</tr>
</tbody>
</table>

**What to do next**

If you use server-side printing, ensure that your printer supports duplex printing by using the `$GREDIR/bin/printerinfo` script. In your Genero BDL source file, ensure that the printer is set up to print on two sides.

```bash
fgl_report_setPrinterSides("two-sided-long-edge")
```

If you use client-side printing, use the Print dialog box to configure the printer. Double-sided printing is not possible if you are printing silently on a client-side printer.
**Start on an odd or even page**

Ensure that part of your report starts on an odd or even page. For example, a batch of invoices are printed in duplex mode, so a new invoice must always start on an odd page.

**About this task**

The following PXML Boolean functions, in the **X-Size** or **Y-Size** properties, reveal whether an object starts on an odd or an even page:

- `oddPhysicalPage()` yields TRUE if, when the size is computed, the total number of physical pages (including the page currently under construction) is odd.
- `evenPhysicalPage()` yields TRUE if, when the size is computed, the total number of physical pages (including the page currently under construction) is even.

**Note:** Objects can spill over to subsequent pages. An object for which `oddPhysicalPage()` yielded TRUE might end on an even page in the final document.

1. In your report design document, before the object you want to start on an odd or even page, create a **Vertical Box** or other object.
2. In the Properties view for the box, set **Y-Size** to:

   ```plaintext
   oddPhysicalPage()?0:max
   ```

**Design labels**

A label is a logical page that is printed several times on a single physical page. For a report application that prints out labels, the report design document (.4rp) is the size of a single label.

**About this task**

A report program programmed to output labels expects the report design document to represent a single label.

To code a Genero report application that creates labels in a report format, see Create labels: the report program (Genero BDL) on page 692.

1. Create a new report.
   Select **File > New > Reports > Empty Report (.4rp)**.
2. In the **Data View**, specify a Data Schema.
   See Adding report data (Data view) on page 796.
3. Set the page size to the size of a single label.
   a) Select **File > Report properties > Paper Settings**.
   b) Set the **Page Size Format** to **Custom**.
   c) Set the paper settings to the size of one label. Adjust margins as needed.
4. In the custom page you've created, design the label as you would design any report, to include adding fields from the Data view.

In this example, the page has a width of 9.90 cm and a height of 4.30 cm. The page contains six WordBox objects. Each WordBox object is populated with data from fields listed in the Data View.
5. In the **Report Structure**, place the report under the appropriate trigger.

   In this example, the label (Page) is positioned under the **orderid** trigger, meaning a new label is printed each time there is a change in **orderid**.

6. **Save your report.**

   For an example, see the `orderlabels.4rp` report in the OrderReport demo application.

**Design address labels**

Design for a label that may contain three to five lines, depending on the data record.

**Before you begin**

Before reading this procedure, you should be familiar with designing a basic label report. See [Design labels](#) on page 805 for more information.

**About this task**

A common label need is printing of address labels, yet the number of lines required for an address can vary, depending on the complexity of the address. In many database tables that store address data, there are several fields for storing address information, such as `addr1`, `addr2`, and so on. When creating an address record, those fields that are not needed for the new address are set to **NULL**.
When an address is printed out, however, those addresses that contain empty fields (addr2 is set to NULL, for example) can cause an issue. No blank line should appear on the label. In addition, we may have information that we want to print after the last non-blank address line included (such as a postal code).

Follow this procedure to answer these address issues.

**Note:** Field names used in this example have been simplified. Use the full field names as they exist in the Data View.

1. **Create your address label report.**
   a) Use a **Vertical Box (Layout Node)** to contain all of the label data.
   b) Add all the lines of the address as children of this node, using dynamic layouting.

   At this point, you have designed a report that prints an address label. If one of the lines is empty, however, a blank line is printed.

2. **Identify which lines may contain empty values.**

3. **For each line that may contain an empty value, set the visibilityCondition to specify that the line not print if the content is blank.**

   For example, if one of the address label lines contains the data value shipaddr2, and this field has the potential of being empty, you could set the visibilityCondition as follows:

   ```java
   shipaddr2.trim().length()>0
   ```

   With the visibilityCondition set properly, the line will not print if it has a length of zero. No blank lines appear within the address.

4. **If you have a set of lines where some may be blank, and you wish to print something at the end of the last non-blank line, you set this up using a conditionality expression in the value property. With this expression, you test to see whether any of the subsequent (or following) lines contains a value. If one or more of the lines contains a value, the current line is printed. If none of the subsequent lines contain a value, then the postcode is appended to the end of the current line and printed.**

   For example, consider an address label containing three lines: `addr1`, `addr2`, and `addr3`. You have an additional field, `postcode`, that you wish to print after the last non-empty line.

   - For the line containing `addr1`, we test and see whether `addr2` and `addr3` are empty by setting the value as follows:
     ```java
     addr3.trim().length()+addr2.trim().length()==0?
     addr1.trim()+postcode.trim():addr1.trim()
     ```

   - For the line containing `addr2`, we test and see whether `addr3` is empty by setting the value as follows:
     ```java
     addr3.trim().length()==0?
     addr2.trim()+postcode.trim():addr2.trim()
     ```

   - For `addr3`, it only prints if it is not empty (assuming the visibilityCondition is set correctly). Therefore, set the value as:
     ```java
     addr3.trim()+postcode.trim()
     ```

   With the value property set properly, the last non-empty line will have the postcode at the end.

**Specify the paper settings of a report**

Paper settings set the paper orientation, the page size format, and the report margins for a report.

Each report defines these paper settings:

**Orientation**

Select **Portrait** or **Landscape** as the layout for the report.

**Units**

Specify whether to use **centimeters** or **inches** when defining the page size and margins for a report.

**Page Size Format**

Set the page size using either **Standard sizes** or **Custom** sizes. With standard, select the desired standard size from the list provided in the combobox. For custom, specify the height and width of the report.

**Margins**

Each page has four margins: top, bottom, left, right.

**Paper Settings order of precedence**

There are three areas which determine the paper settings for a report, listed here in the order of precedence:
1. In the report application using reporting APIs.
2. In the report design document (.4rp).
3. GRW default values.

**Set paper settings using the reporting APIs**

A report application can use the reporting APIs to override the paper settings for an individual report. For Genero BDL applications, see Change paper settings and output format on page 702.

**Set paper settings for a report design document (.4rp)**

The Genero Report Writer preferences specify the initial paper settings for a new report.

To change the paper settings:

1. Open the report in Genero Report Designer.
2. Select **File > Report properties, Paper Settings...** The **Paper Settings** dialog opens.
3. Set the desired orientation, units of measure, page size format, and margin settings for the report.
   
   **Tip:** To set the paper settings to the current values set in the **Paper Settings Configuration** of the local Genero Report Writer, click **Load from Default**.
4. Click **OK** to save your settings and close the **Paper Settings** dialog.

**Set paper setting preferences for GRW**

The **Paper Settings Configuration** page sets the default paper settings for all new reports.

1. Select **Tools > Preferences, Report Writer, Paper Settings**.
2. Set the desired orientation, units of measure, page size format, and margin settings.
   
   **Tip:** To set the paper settings to the installation default values, click **Load from Default**.
3. Click **Apply** to save your preferences.
4. Click **OK** to close the **Preferences** dialog.

Once created, the paper settings for a report are stored as part of the report design document (.4rp). Changes to the **Paper Settings Configuration** have no effect on existing reports.

**Configure the output**

The same report can be output in different formats and to different output devices.

For information about setting the orientation, units, page size, and margins, see Specify the paper settings of a report on page 807.

To select the output format and other options for the current report, select **File > Report Properties > Output Configuration**.
Figure 364: Output Configuration dialog

Options for Output format:

- **SVG** (scalable vector graphics - can be displayed using the Report Viewer feature of GDC)
  - Rendering - Select options to minimize the aliasing distortion
- **PDF** (Acrobat PDF format, can be displayed using PDF viewer)
  - Rendering - Select options to minimize the aliasing distortion or set monochrome mode
  - Page Range - output all pages or enter a range
- **Image** (creates an image, such as .jpg. You can select the image type.)
  - Rendering - Select options to minimize the aliasing distortion or set monochrome mode
  - Page Range - output all pages or enter a range
  - **Image settings** - Select image **Type**, **Resolution**, **prefix for the image** filename

**Note:** The Rendering options for font antialiasing (SVG or PDF documents) will only take effect if the **fidelity** property of report text elements is set to "True".

Functions from the Reporting API can be used in your report program to override the default options at runtime. For example, the function fgl_report_selectdevice provides additional output formats. See the Genero Studio >> Report Writer documentation topic "Report API Functions".

**Other Report Designer tasks**

These procedures help you complete specific report design tasks.

- Change the type of a report element on page 810
Change the type of a report element
Change a report element from one type to another, for example, from a Word Box to a Word Wrap Box.

1. In the report, right-click on the element you want to change.

   **Tip:** To convert the type of multiple report elements at once, Ctrl-click each element before right-clicking.

2. Select **Convert To** and choose the new type from the list.

   The *name* property of the element does not change, unless the name of the old node is of the format [Type] [Number]. A node named WordBox12 would be renamed, for example.

   By default, the new object type has the properties set that it has in common with the old type. For some type conversions, additional properties may be set. For example, when you convert a Decimal Format Box to a Word Box, the **X-Size** and **Y-Size** properties remain the same, but the **Value** property is converted to a string value and assigned to the **Text** property.

Related concepts
**Report Design Elements (The Toolbox)** on page 859
The Toolbox contains report object that can be placed on a report design document.

**Design Documents for pre-printed forms**
Use an Image Box and other report elements to create a pre-printed form with absolute positioning.

A simple drag and drop of the report elements invokes the absolute positioning method, which allows you to place an element in the desired location on the form. To make this easier, add an Image box to the report, containing an image of the form to serve as a background.

![Figure 365: Report with Image](image-url)
The report design window contains a global grid; as you drag a report element onto the design window, black grid lines help you align it with the other elements on the form. The red attachment point indicates the mapping of the element to the global grid.

Once you have dropped an element, you can refine its location by dragging, or you can select an element and use the keyboard arrow keys:

- Pressing an arrow key moves the element incrementally in the corresponding direction
- Shift-arrow key moves the element along the global grid.

Right-click an element to display a menu of additional options.

Related concepts

Absolute or relative positioning on page 782

The position of a report element can be absolute, to enable reports that use pre-printed forms, or relative, adjusting as needed based on the length of the report element as well as its parents and siblings.

Force a page or line break

Force a page or line break by setting the X-size or Y-size property to "rest".

To force a page break, insert a container in the page at the point you want to break.

To force a line break, insert a container in the stripe at the point you want to break.

If the container has a vertical layout direction, set the Y-size property (width) to rest. If the container has a horizontal layout direction, set the X-size property (height) to rest. The container then stretches to fill the remaining space in the appropriate direction.

Print a Layout-dependent reference (InfoNodes)

Use InfoNodes to print a value on the report that depends on the paged stream resulting from the report layout.

For example, a value for "total from previous page" can vary depending on how the page options for a report are set. For a report layout that works with various page sizes, you can use an InfoNode and a Reference Box.

This example illustrates how to print the total price (overalltotal) from a previous page.

In the Genero BDL report program

The REPORT block of the Genero BDL file must calculate the desired value and output it to the report. The following example is from the OrderReport.4gl file in the demo sample programs:

```
ON EVERY ROW
  LET lineitemprice = orderline.lineitem.unitprice * orderline.lineitem.quantity
  LET overalltotal = overalltotal + lineitemprice
  LET ordertotal = ordertotal + lineitemprice
  PRINT orderline.*, lineitemprice, overalltotal, ordertotal
```

The variable overalltotal contains the running total price of the line items on the report.

In the report design document (.4rp)

Use these objects from the Toolbox in your report design:

- InfoNode - place this object in the container for the ON EVERY ROW trigger of your Structure view. This creates an invisible column in your report line containing the value of the InfoNode.

  The Value property of the InfoNode must be a String.

  You can use the fglValue member of the FGLNumericVariable class to convert overalltotal:

```
overalltotal.fglValue
```

This formats the value of overalltotal as a String based on the default format set in the Genero DVM.
You can also use the `format` method of the `Numeric` class to convert to a string and also specify the format, as in this example:

```java
overalltotal.format("-,-,---,--&.&&")
```

- **Reference Box** - place this object in the **Page Header** at the top of the report structure.
  - For the **InfoNode Name** property, enter the name of the InfoNode that you created.
  - For the **Text** property, enter a string used to determine the maximum length of the value in the InfoNode, because the value will not be known at the time the ReferenceBox is positioned. Examples: Enter 000,000.00 as the maximum length for a value from a numeric data type, or MMMM as the maximum length for a value from a CHAR(4) data type.
- **WordBox** - optionally use this object to add some text next to the Reference Box.

A Reference Box points to the immediately previous occurrence of the InfoNode value in the paged stream. Because you placed the Reference Box in a Page Header, it will point to the last occurrence of the `overalltotal` value on the previous page.

**Example**

**Figure 366: Report Structure with reference box and info node** on page 813 shows an example report structure (from the `OrderReport.4rp` report in the Reports demo). **Figure 367: Report showing total in a reference box** on page 814 shows how this report looks when run.
Reference box "ReferenceBox3" is in the Any Page header.

Info Node "overalltotal" is in "On Every Row".

Reference box uses the InfoNode Name property to connect to the Info Node.

Figure 366: Report Structure with reference box and info node
Show or hide a report element

Make report elements conditionally invisible by using the **Visibility Condition** property.

1. In the report, select the element.
2. In the Properties view, edit the **Visibility Condition** property.

   For example, you don’t want the `email` field to display if its value is empty, so you set the visibility condition as follows:

   ```
   email.trim().length()>0
   ```

   If the expression evaluates as TRUE, the field displays. If the expression evaluates as FALSE, the instance of the element is removed from the report, as are all its children. If you are using relative positioning, all sibling elements after this element in the report structure shift accordingly, reclaiming the space that the element would have occupied.

Use hyperlinks in a report

Use the id and href properties to add hyperlink functionality to a report.

The `id` and `href` properties can be specified for text and images in the following containers: `wordBox`, `wordWrapBox`, `decimalFormatBox`, `pageNoBox`, `referenceBox`, `imageBox`, and `htmlBox`.

The **Hyperlink** properties group contains two properties:

- **id**
  - Creates an anchor in the document. Identify a node using a unique id. The unique id can then be used as the target of a hyperlink.

- **href**
  - Defines a hyperlink pointing to a resource on the Internet, local disk, or an anchor inside the document. The href should be defined using the URI syntax. Examples include:

    ```
    http://www.google.com
    
    mailto:santa.claus@northpole.com
    
    file:///C:/animals/images/honey_badger.jpg
    
    #ref
    ```
A hyperlink only functions on report outputs that are interactive. Hyperlinks do not work in reports that are output to Image, Printer or Postscript formats.

Providing a link to a place within the report

By using the hash symbol (#), you can provide a hyperlink from one place to another within the report. This is implemented by using the **href** property of the link-from object and the **id** property of the linked-to object.

For example, create a Word Box called `SeePaymentTermsForDetails` and set its **href** to `#paymentTerms`. Create a Word Wrap Box called `PaymentTerms` and set its **id** to `paymentTerms`.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Value</th>
<th>Properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object</strong></td>
<td></td>
<td><strong>Object</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>SeePaymentTermsForDetails</td>
<td><strong>Name</strong></td>
<td>PaymentTerms</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>WordBox</td>
<td><strong>Type</strong></td>
<td>WordWrapBox</td>
</tr>
<tr>
<td>Visibility Condition</td>
<td></td>
<td>Visibility Condition</td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td></td>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>Comment</td>
<td></td>
<td>Comment</td>
<td></td>
</tr>
<tr>
<td><strong>Hyperlink</strong></td>
<td></td>
<td><strong>Hyperlink</strong></td>
<td></td>
</tr>
<tr>
<td><strong>href</strong></td>
<td>#paymentTerms</td>
<td><strong>href</strong></td>
<td></td>
</tr>
<tr>
<td><strong>id</strong></td>
<td></td>
<td><strong>id</strong></td>
<td>paymentTerms</td>
</tr>
</tbody>
</table>

**Figure 368: Hyperlink objects**

In the report, when the user clicks the `SeePaymentTermsForDetails` hyperlink, they are sent to the `PaymentTerms` object.

**Some tips for legacy report designers**

This table answers some common questions the correlation between Report Designer and traditional 4GL commands in reports:
Table 214: Legacy Report to Genero Report Writer

<table>
<thead>
<tr>
<th>Legacy Report command</th>
<th>Using Genero Report Writer</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEFORE GROUP OF <code>groupname</code></td>
<td>In the Report Structure view, place the Page Root container as a child of the element you want to group by.</td>
</tr>
<tr>
<td>SKIP TO TOP OF PAGE</td>
<td></td>
</tr>
</tbody>
</table>

![Report Structure Diagram](image)

Figure 369: Page Root as child of group
<table>
<thead>
<tr>
<th>Legacy Report command</th>
<th>Using Genero Report Writer</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFTER GROUP OF <em>groupname</em></td>
<td>In the report design document, add a container that will consume the remainder of the page. See also Force a page or line break on page 811.</td>
</tr>
<tr>
<td>SKIP TO TOP OF PAGE</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 370: Container that consumes the page**

There is a GROUP trigger for data control breaks in the report structure. The position and contents of the child containers of the trigger determine what is printed out and when. See Print group totals and report totals on page 801.


<table>
<thead>
<tr>
<th>Report Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Report Structure" /></td>
</tr>
</tbody>
</table>

In the report design document, the Data View tab displays the list of data items in the order in which they are specified in the PRINT statement.
## Legacy Report command

**ON LAST ROW**

Drop a container positioned as the last child of the page root. The contents of the container will print out after the last report row.

### Figure 371: Contents on last row

See [Display totals on a report](#) on page 802.

## Using Genero Report Writer

**NEED \( n \) LINES**

Put all the report elements that need to be kept together in a [Vertical Box Layout Node](#) container. If there is not sufficient room on the page to print all of the elements in the container, the entire container will be printed on the next page.

**PAGE HEADER, PAGE FOOTER**

Avoid using these control breaks, which are triggered by the line count of the BDL report, which does not correspond with the actual page breaks in the report output by Report Writer. Create [page headers and footers](#) in the report design document instead.

### Figure 372: Headers and Footers
Working with tables

A table object displays data in columns and rows.

- Add a table to a report on page 819
- Assign content to a table column on page 819
- Set the triggers for a table in a report on page 820
- Merge cells on page 820
- Add rows or columns on page 820
- Add table headers and footers on page 820
- Change the width of a table on page 821
- Change the width of a column on page 821

Related concepts

Table on page 867

A table object has the ability to display data in columns and rows.

Add a table to a report

Add a simple table to a new report.

About this task

This procedure presumes you are starting with an empty report, but it provides you with the information you need to add a table to any report.

1. Create a new, empty report.
   
   Select File > New > Reports > Empty Report (.4rp) and click OK.
   
   An empty report design document (.4rp) displays.
2. In the Data View, open a schema file.
3. From the Toolbox, add a table to your report design document.
4. For each column, assign the field to display.
   
   See Assign content to a table column on page 819.
5. Set the report trigger.
   
   See Set the triggers for a table in a report on page 820.
6. Save and execute the report.
   
   You will likely have to modify your report application to call your new report.

Assign content to a table column

Associate a field from the Data View with a column header and body element.

About this task

This procedure shows you how to associate a field with a column. However, a column can contain any content, for example, a column can include another table or a chart.

1. Select the Data View tab. Identify the field you wish to use for a column in the table object.
2. Select the Create an element based on the document context icon from the Data View Toolbar.
3. Using relative positioning, hold down the CTRL key and drag the field from the Data View and drop it into the header row for the desired column.
   
   A column header is created. The column title is placed as a WordBox object. The Class property reflects that this is a title for the column, rather than a value.
4. Using relative positioning, drag the field from the Data View and drop it into the body row for the desired column.
   
   An expression is created with the value of the field. The field is placed as a drawable reflective of the data type of the field. The Class property reflects that this display the value (instead of the title).
Set the triggers for a table in a report
Set the triggers to create a table row for each data row streamed to your report.

1. In the Report Structure, find the row element for your table.
2. Right-click the row element.
   The context menu for the row element displays.
3. Select Repeat selected items on > On Every Row.
   When you place the table row under the On Every Row trigger, each data row results in one row added to the table.

Merge cells
Merge one or more columns (cells) within a row into a single cell. You can also revert the merge and convert the merged cell back to its original number of cells.

About this task
You can only merge the cells within a row. You cannot merge cells across rows.

1. Select multiple cells.
   Hold down the Ctrl key and click on each cell you wish to merge. As each cell is selected, it turns blue in color.
2. Right-click and select Merge Cells from the context menu.
3. To reverse the merge, right-click on the merged cell and select Split Cells from the context menu.
   The merged cell is split back into its original columns. Any content of the merged cell is put into the first column.

Related concepts
colspan (Column Span) on page 880
The colspan property sets when two or more cells are merged in a row definition for a report table.

Add rows or columns
Add additional rows and columns. By default, a column has two rows and three columns.

About this task
Do not use the toolbox to add a column or row to a report table object.

1. Right click on a row or column, either in the cell itself or on the control (selection tab) for the column or cell.
   You will be inserting a row or column relative to the row or column you click. If you are inserting a new row, you will be able to add the new row above or below the selected row. If you are inserting a new column, you will be able to add the new column to the left or right of the existing column.
2. If you clicked on a cell instead of a control (selection tab):
   a) From the context menu, select Insert Table Item.
      A second context menu displays with options to insert a row or a column.
   b) Choose the appropriate option to add a row or a column.
3. If you clicked on a column or row control instead of within a table cell, the context menu is specific to the column or row. Select the appropriate option from the context menu.
   A new column or row is added to the existing table. Adding a column does not change the width of the table. The columns are resized to fit the new column.

Add table headers and footers
Add additional headers or footers to a table. By default, a table has a single header (Any Page Header).

About this task
Do not use the Toolbox to add a header or footer to a table.

1. Right click on a cell in a row or column.
   Which cell you click into is not relevant, as you will be specifying the type of header or footer to add.
2. From the context menu, select Insert Table Item.
   A second context menu displays with options.
3. To add a header, select **Insert Header**. You are asked to choose between an any page header, a first page header, an even page header, or an odd page header. Select the appropriate header type.
   - If an option is grayed out, then that specific header has already been added to a report. To provide two rows for a specific header, you do not add two headers of the same type; you add two rows to the specific header type.

4. To add a footer, select **Insert Footer**. You are asked to choose between an any page footer, a first page footer, an even page footer, or an odd page footer. Select the appropriate footer type.
   - If an option is grayed out, then that specific footer has already been added to a report. To provide two rows for a specific footer, you do not add two footers of the same type; you add two rows to the specific footer type.

A new row is added to the existing table for the added header or footer. Headers are displayed at the top of the report design document, while footers are displayed at the bottom.

**Change the width of a table**

Specify the width of a table. If you do not specify a width, the table expands to the width of the parent container.

**About this task**

This procedure assumes that you have not explicitly sized individual columns with an absolute value.

**Note:** You can set the width of a table, but you should not try to set the height of a table. The height is determined by the number of rows created.

1. Select the Table object.
2. Hover your cursor over the right-most border of the last column tab in the report until the resizing arrows appear.
3. Move the mouse until the table is the size you desire.

   The **X-Size** property will change from the default (max) to a number representing the width you have selected.

**Change the width of a column**

A column width can be proportional or fixed. If you do not specify a width, the columns are equal in width, and the width is calculated based on the width of the table itself.

When you specify a value in the Proportional Width property, you are specifying its width in proportion to other columns in the same table. For example, consider a table with three columns: columns A, B and C. Column A has a proportional width setting of 1, column B has a proportional width of 2, and column C has a proportional width of 3. This means that column B is two times as wide as column A, and column C is three times as wide as column A.

When you specify a value as Fix Width, you are giving an absolute size for that column. By default, the number entered refers to points, but you can change the unit of measure by specifying the type of units used. See **Unit Names** on page 978.

**Note:** If you specify both Proportional Width and Fix Width, the Fix Width value is used.

1. Select the table.
   - The controls (selection tabs) appear at the top of each column and to the left of each row.
2. Place the cursor on the right-side border of the selection tab for the column you wish to size. You can then drag the column to make it bigger or smaller.
3. For more precision, you can edit the property directly.
   - To change the column width in proportion to other columns, change the **Proportional Width** property. Using the up and down arrows on the property field increases and decreases in increments of one.
   - To specify a specific width, enter the value in the **Fix Width** property. Use the Reset button to clear the value from the **Proportional Width** property.

**Working with business graphs**

A business graph allows you to represent data visually on the report.

- **Types of business graph** on page 822
- **Creating a graph** on page 824
- **Output charts as tables** on page 835
Types of business graph
All of the business graphs in Genero Report Writer map Numeric values, grouping the data for the desired result.

Map charts
The map chart allows you to create a graph that has one set for values to be mapped, grouped together by a specific key.

The chart is defined by the Map Chart object. The key and value are defined in the Map Chart Item. For example, the pie chart in Figure 373: Pie Chart example on page 822 presents the total revenue (the value) for each customer (the key), based on the OrderReport.rdd file in the Report Writer sample programs.

Figure 373: Pie Chart example
The default map chart is the pie chart. You can use the Draw As property to produce other types, including 3D pie, bar, and ring. Map charts can also be output as tables.

Related tasks
Create a map chart on page 824
Map charts have one key value and one data value. For example, you could map the revenue distribution by customer.

Category charts
The category chart allows you to create a graph that has two keys for each set of values.

The chart is defined by the Category Chart object. The keys and values are defined in the Category Chart Item. For example, the bar chart in Figure 374: 3-D Bar Chart example: Revenue by Customer and Categories on page 823 presents total revenue (the value) by the Customer Name and Product Category (the keys).
Figure 374: 3-D Bar Chart example: Revenue by Customer and Categories

The default category chart is the bar chart. You can use the Draw As property to produce other types, including 3D bar, stacked bar, area, line, spider web, and waterfall. Category charts can also be output as tables.

Note: Spider web charts can also be called radar charts.

Note: If you select Waterfall as the chart type, the value in the last category of the data set should be (redundantly) specified as the sum of the items in the preceding categories - otherwise, the final bar in the chart will be incorrectly plotted. At the present time, the chart can only have one category.

Related tasks
Create a category chart on page 826
Category charts have two key values and one data value. For example, you could map revenues by area and customer.

XY charts
The XY chart allows you to create a graph that maps a series that has two values, as an XY-plot.

The chart is defined by the XY Chart object. The values for the X and Y axis are defined by the XY Chart Item. This chart is used most often for scientific data.
Figure 375: XY Chart example

The default XY chart is the line chart. You can use the Draw As property to produce other types, including area, polar, scatter, step, and time series. XY charts can also be output as tables.

Related tasks
Create an XY chart on page 827

XY charts have two data values. For example, you could map quantity sold against discount percentage.

Creating a graph
Place a business graph inside a report, and then edit the properties for the chart and its items.

Create a map chart
Map charts have one key value and one data value. For example, you could map the revenue distribution by customer.

1. Open a new or existing report in Report Designer.
2. If you are creating a new report, specify the data schema to be used for the report.
3. From the Tool Box view, drag and drop the Map Chart into a container on the report design.

   The Design Window contains a Chart object and its related Item object (shown as a price tag).
4. Select the chart object and enter the values for its properties in the Properties View:
   - Title - caption at the top of the graph
   - Keys Title - caption for the keys
   - Values Title - caption for the values
   - Draw As - the type of chart, for example, Pie or Ring. The default is a pie chart.
   - Sort By - sort alphabetically, numerically, or by input order.
   - Sort Ascending - sort in ascending or descending order.
   - DrawLegend - include or exclude the chart legend.

   Tip: When several similar charts are drawn next to each other, set DrawLegend to False on all charts except one. The charts now appear to be sharing the same legend.
   - DrawLabels - include or exclude the chart labels.

5. Select the chart's price tag, which represents the item object, and enter the following properties:
   - Key - The data item that summarizes the data. For example, in Figure 376: Map chart object in the work area on page 825, the key is the category name. The key must be a String. To use a non-string value, define a custom string for the key, using the data item in an expression.
   - Value - The data item that contains the numbers to be charted. For example, in Figure 376: Map chart object in the work area on page 825, the value is the unit price. The value must be Numeric.
- **Color** - The color of each slice in the map chart. If not specified, the slices display using the Genero default colors. See Coloring a map chart on page 830.

You must modify the report structure to ensure the chart displays as required.

**Related concepts**

Map charts on page 822

The map chart allows you to create a graph that has one set for values to be mapped, grouped together by a specific key.

**Create a category chart**

Category charts have two key values and one data value. For example, you could map revenues by area and customer.

1. Open a new or existing report in Report Designer.
2. If you are creating a new report, specify the data schema to be used for the report.
3. From the Tool Box view, drag and drop the Category Chart into a container on the report design.

   The Design Window contains a Chart object and its related Item object (shown as a price tag).

![Figure 377: Category chart object in the work area](image)

4. Select the chart object and enter the values for its properties in the Properties View:
   - **Title** - caption at the top of the graph
   - **Keys Title** - caption for the keys
   - **Values Title** - caption for the values
   - **Categories Title** - caption for the categories
   - **Draw As** - the type of chart, for example, Bar or Area. The default is a bar chart.

5. Select the chart's price tag, which represents the item object, and enter the following properties:
   - **Key** - The data item that summarizes the data. For example, in Figure 377: Category chart object in the work area on page 826, the key is the category name. The key must be a String.
• **Value** - The data item that contains the numbers to be charted. For example, in Figure 377: Category chart object in the work area on page 826, the value is the unit price. The value must be Numeric.

• **Category Key** - Within each key value, the data that you summarize by. For example, in Figure 377: Category chart object in the work area on page 826, the category key is the sales person's name. The category key must be a String.

To use a non-string value for **Key** or **Category Key**, define a custom string for the key, using the data item in a expression.

You must modify the report structure to ensure the chart displays as required.

**Related concepts**

**Category charts** on page 822

The category chart allows you to create a graph that has two keys for each set of values.

**Create an XY chart**

XY charts have two data values. For example, you could map quantity sold against discount percentage.

1. Open a new or existing report in Report Designer.
2. If you are creating a new report, specify the data schema to be used for the report.
3. From the Tool Box view, drag and drop the **XY Chart** into a container on the report design.
   - The Design Window contains a Chart object and its related Item object (shown as a price tag).
4. Select the chart object and enter the values for its properties in the Properties View:
   - **Title** - caption at the top of the graph
   - **X Axis Title** - caption for the X axis
   - **Y Axis Title** - caption for the Y axis
   - **Draw As** - the type of chart, for example, Scatter or TimeSeries. The default is a line chart.

5. Select the chart's price tag, which represents the item object, and enter the following properties:
   - **Series Title** - Determines the color and legend entry for the data point. The series title must be a String.
   - **X** - What to plot on the X-axis. X must be Numeric.
   - **Y** - What to plot on the Y-axis. Y must be Numeric.

You must modify the report structure to ensure the chart displays as required.

**Related concepts**

XY charts on page 823
The XY chart allows you to create a graph that maps a series that has two values, as an XY-plot.

**Custom keys**

Enter any valid expression for the String value of a key property. This could be a substring or a concatenation of existing strings.

For example, this expression would group the data values based on the first letter of shiplastname:

\[
\text{orderline.orders.shiplastname.substring(0,1)}
\]

**Examples**

This chart uses the last name as the key (trimmed of trailing blanks), as shown in the Properties view. The unit price on each order for each unique last name is rolled up to a total as shown on the chart. There are five unique names:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>{orderline.orders.shiplastname.trim()}</td>
</tr>
<tr>
<td>Value</td>
<td>{orderline.lineitem.unitprice}</td>
</tr>
<tr>
<td>Name</td>
<td>Item2</td>
</tr>
</tbody>
</table>

**Figure 379: Properties panel for the chart Items (where key uses trimmed last name)**

And this chart uses the first letter of the last name as the key, as shown in the Properties View. The unit price on each order for each unique first letter is rolled up to a total as shown on the chart. In this chart, there are only four unique first letters, as two customers have last names beginning with D:

**Figure 380: Chart showing Revenue Distribution by Customers (where key uses trimmed last name)**
Figure 381: Properties panel for the chart Items (where key uses substring)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>orderline.orders.shplastname.substring(0,1)</td>
</tr>
<tr>
<td>Value</td>
<td>{orderline.lineitem.unitprice}</td>
</tr>
<tr>
<td>Name</td>
<td>Item2</td>
</tr>
</tbody>
</table>

Figure 382: Revenue Distribution by Customers (where key uses substring)

Related tasks

Create a map chart on page 824
Map charts have one key value and one data value. For example, you could map the revenue distribution by customer.

Create a category chart on page 826
Category charts have two key values and one data value. For example, you could map revenues by area and customer.

Coloring a map chart
Genero Studio provides default colors for the slices in a map chart. These colors are suitable for most purposes but can be changed if required.

For example, you might want to change the color if:

- You want to keep consistency of color across several charts with similar data.
- The content of the field is associated with a particular color. For example, your chart shows political parties and you want to display the Green Party in green.

The color of the slices is determined by the Color property of the Map Chart Item.
**Example 1: Display by hue**

You want to display your chart in shades of blue. Set the Color property as follows:

```java
Color.BLUE
```

The map chart displays as in Figure 383: Chart showing shades of blue on page 831.

![Figure 383: Chart showing shades of blue](image)

**Example 2: Pick specific colors**

Your chart displays sales for three different types of fruit and you want to use the following colors:

- Bananas: Yellow
- Apples: Green
- Strawberries: Red
- Blueberries: Blue

You could set the color within the RTL code:

```java
item.fruitname.trim()=='Bananas'?Color.YELLOW:item.fruitname.trim()=='Apples'?Color.GREEN:
item.fruitname.trim()=='Strawberries'?Color.RED:item.trim()=='Blueberries'?Color.BLUE:null
```

However, it's more likely that you'd set the RGB values within the database and define the Color property using those values:

```java
Color.fromRGBA(item.color.redValue, item.color.greenValue, item.color.blueValue)
```

The map chart displays as in Figure 384: Chart showing colored slices on page 832.
Related concepts

The Color Class on page 946
The Color class provides methods for specifying the color of an object.

Related tasks

Create a map chart on page 824
Map charts have one key value and one data value. For example, you could map the revenue distribution by customer.

Specifying classic or default style

The business graphs use a "flat design" by default, but you can change this to use the more 3D "classic" design. You can also choose between three different color schemes.

Two drawing styles are available for business graphs:

- **classic** – the classic drawing style used by earlier versions of Genero Studio.
- **default** – the newer drawing style. This is a "flat design" using fewer shadows, 3D effects, and gradients than the classic style.

If no drawing style is specified, the default style is used.

Three color schemes are available for business graphs: JF, FV and D3. JF is the default scheme for the classic drawing style and FV is the default scheme for the default drawing style.

Use the fgl_report_setRenderingHints function with the pxmChartStyle hint to specify the style and color options. The style options are passed as a comma-separated list of styles. For example, the following code selects the original drawing style using the D3 color scheme:

```plaintext
DEFINE renderingHints om.SaxAttributes
LET renderingHints=om.SaxAttributes.create()
CALL renderingHints.addAttribute("pxmChartStyle","classic,D3")
CALL fgl_report_setRenderingHints(renderingHints)
```
Examples

Figure 385: Default drawing style, FV color scheme

Figure 386: Classic drawing style and color scheme

Related concepts
Optimize the rendering process on page 715
You can optimize the rendering process by setting rendering hints.

Modify the report structure
When you add a business graph (map chart, category chart, or XY chart) to your report, the chart and its chart item are added to your report structure. You need to modify this report structure to ensure that your business graph displays correctly.

1. Open a new or existing report in Report Designer. If the business graph does not exist, create it. See Create a map chart, Create a category chart, or Create an XY chart.

   The Structure view shows the position of the chart and the chart items. When a business graph is created, the structure displays as in Figure 387: One chart per each row on page 834. It will create a new page and chart for each data row passed to the report, and therefore needs to be modified.
2. The location of the chart object (map chart, category chart, or XY chart) specifies how many different charts will be created, based on the resulting subsets of the data. Drag and drop the chart object to the desired location in the hierarchy.

**Tip:** If you drop a container or trigger on a different node, it will be a child of that node. Use Alt-drag and it will become the parent of that node.

3. The item object for the chart (map chart item, category chart item, or XY chart item) must a child of the node where the value changes, typically the OnEveryRow trigger node. Drag and drop the chart item object to the node.

4. Save the report.

**Examples**

With **Figure 388: One chart, one page** on page 834, the Structure view results in a page containing one chart; the item node is a child of OnEveryRow:

![Structure View](image)

**Figure 388: One chart, one page**

With **Figure 389: One chart for each unique userid** on page 835, the Structure View results in one chart for every unique user id, since the Map Chart is a child of the user id trigger node and the item node is a child of OnEveryRow:

![Structure View](image)
All chart types now support the drawing of the data as tables via the Draw As property.

The tables drawn are pivot tables that may contain subtotals. This terminology and rules are applied:

- The columns of the tables are either of type **Dimension** (the data items that are used to group values) or of type **Value**.
- **Values** are aggregated or totaled by the **Dimension**.
- **Dimension** columns precede **Value** columns in the table.
- The order of the **Dimensions** specifies the order of the data.
- Subtotals are generated for each **Dimension** except the rightmost.

**Mapping of chart properties**

- **Map chart** - the **Key** property is mapped to a **Dimension** column, and the **Value** property to a **Value** column.
- **Category chart** - the **Category Key** property and the **Key** property are mapped to **Dimension** columns, and the **Value** property to a **Value** column.
- **XY chart** - the **Series Title** property is mapped to a **Dimension** column and the **X** and the **Y** properties are mapped to **Value** columns.

There are three types of tables: Table, Sorted table, and Aggregated table.

**Table**

This option lists all data items in a table. The data needs to be presorted in the order of the dimensions; if this is not the case, the table will contain useless subtotal rows.

If the number of rows in the table is large, then **Table** is the preferred choice since it produces the tabular output row by row while reading the input and does not keep a copy of the table data in memory. In other words, this option does not delay the output until the end of the input has been read. as the Sorted Table and Aggregated Table options do.

**Sorted Table**

This option produces the same output as the Table option, but the data does not need to be presorted. The output is delayed until the last row of table data has been read, and the entire table data is stored in main memory.

**Aggregated Table**

This option draws the same table as the previous two options, but subsequent rows with identical dimensions are drawn only once and the total values are computed. This option always sorts the data, and delays the output until the last row of data has been read. This option is not available for **XY Chart** types.

**Related concepts**

*Working with tables* on page 819
A table object displays data in columns and rows.

**Table** on page 867
A table object has the ability to display data in columns and rows.

**Working with pivot tables**

The pivot table element defines a table element with fixed roles and types for its columns.

- **What are pivot tables?** on page 836
- **Sample pivot table reports** on page 839
- **Create a pivot table** on page 840
- **Pivot table properties** on page 841
- **Add a dimension to a pivot table** on page 841
- **Add a measure to a pivot table** on page 842
- **Arrange your hierarchies** on page 843
- **Pivot table elements and the Structure view** on page 844

**Related concepts**

**Pivot table library** on page 771
The library `libpivot.4gl` provides functions to retrieve information about pivot tables contained in a .4rp file

**What are pivot tables?**

A **pivot table** is a table element with fixed roles and types for its columns, suitable for processing and aggregating multi-dimensional data.

Grouping, sorting, and summarizing can be performed on a pivot table. The results can be displayed in different ways.

A pivot table has two types of columns: **dimensions** and **measures**. A column is either a dimension or a measure. A pivot table has one type of row, a **fact row**. The values in the cells of a row are either **dimension values** or **measures**, depending on the column type.

Data is sorted by the dimension values. A column usually has many rows with identical dimension values. The dimensions can be seen as forming a hierarchy. For this reason, dimension, can also called **hierarchies**.

A measure is aggregated. If the measure is numeric, the aggregation could be an average of the measure values, the sum of the measure values, the maximum or minimum of the measure values, or another mathematical function.

For example, a table has the dimension columns "Country" and "Region". After sorting the data, several rows starting with {"Afghanistan","1 North"...} are at the top, followed by rows starting with {"Afghanistan","3 South"...}, followed by rows starting with {"Albania","1 North"...}. "Country" and "Region" form a hierarchy or tree where a country branch has sub branches for its regions. The innermost dimension contains the "facts" or "values" (the measure columns from the fact rows). In a tree representation, the leaves of the tree are records containing the values for the measure columns.

**Relationship to business graphs**

The pivot table is a generalization of the business graph objects. For example, a CATEGORYCHART is a PIVOTTABLE with two dimensions (The "categoryKey" and "key" attributes in the CATEGORYITEM element) and one measure (the "value" attribute in the CATEGORYITEM element) on which an aggregation is performed for both dimensions. The following table compares the business graph objects to the pivot table.
Table 215: Comparing chart objects and pivot tables

<table>
<thead>
<tr>
<th>Element type</th>
<th>Number of dimensions</th>
<th>Number of measures</th>
<th>Number of aggregation groups</th>
<th>Aggregation functions</th>
<th>Sorting options</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAPCHART</td>
<td>One (specified by the key attribute)</td>
<td>One (specified by the value attribute)</td>
<td>One (values with the same key value are summarized)</td>
<td>Summarizing</td>
<td>By key, value, and input order</td>
</tr>
<tr>
<td>CATEGORY CHART</td>
<td>Two (specified by the key and categoryKey attributes)</td>
<td>One (specified by the value attribute)</td>
<td>One (values with the same key + categoryKey value combination are summarized)</td>
<td>Summarizing</td>
<td>By keys, value, and input order</td>
</tr>
<tr>
<td>XYCHART</td>
<td>None</td>
<td>Two (specified by the x and y attributes)</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>PIVOTTABLE</td>
<td>N (specified by HIERARCHY elements)</td>
<td>N (specified by MEASURE elements)</td>
<td>N (Aggregation can be performed on all dimensions)</td>
<td>Summarizing and others (such as count, average, maximum, minimum, and so on)</td>
<td>Input order and any combination of measures</td>
</tr>
</tbody>
</table>

Related concepts

**Pivot Table Hierarchy Value** on page 873
The HIERARCHY elements represent dimensions. An element represents both the declarative aspects of the column as well as the data value which is typically defined as an RTL expression.

**Pivot Table Measure** on page 874
A MEASURE element represents both the declarative aspects of the column as well as the data value.

**Working with business graphs** on page 821
A business graph allows you to represent data visually on the report.

**Data types in the pivot table**
The pivot table makes use of the set data types Column Selector and OrderSpecifier.

These set data types are defined:

**Column Selector**
The column selector is a comma-separated list of positive integers numbering 0 for the first column and n-1 for the last column in a table of n columns.

Example: Given a table with four columns, a column selection of "0,3,1" selects the first, the last and the second column.

**Order Specifier**
The order specifier is a comma-separated list of positive integers numbering 0 for the first column and n-1 for the last column in a table of n columns. Positive values specify ascending, negative values descending order.

Example: Given a table with four columns, a order specifier of ".-0,3" specified an ascending order on the first column and a descending order on the last column.
There is a difference between not setting a value and specifying an empty value. The empty value always means the empty set. Not setting a value may mean selecting all or nothing, depending on the context.

**Runtime configurability**

The number of possible visualizations for a single pivot table data model is huge. Instead of providing a separate file for each variant, you can define RTL expressions to create the different variants at runtime.

Consider a table with the dimensions “Country”, “Salesperson” and “Year” and the measure “Turnover”.

The list of possible views:

- Total Turnover by Country
- Total Turnover by Country and Salesperson
- Total Turnover by Country, Salesperson and Year
- Total Turnover by Country and Year
- Total Turnover by Country, Year and Salesperson
- Total Turnover by Salesperson
- Total Turnover by Salesperson and Year
- Total Turnover by Salesperson, Year and Country
- Total Turnover by Salesperson and Country
- Total Turnover by Salesperson, Country and Year
- Total Turnover by Year
- Total Turnover by Year and Country
- Total Turnover by Year, Country and Salesperson
- Total Turnover by Year and Salesperson
- Total Turnover by Year, Salesperson and Country

Adding one more dimension multiplies the number of variations by more than four.

If we add two more measures “Margin” and “Cost”, the number of variations is multiplied by two as we can now view “Total Margin by ..” and “Total Cost by ..” for all existing variations.

If we compute not only the total but also the average and the maximum, the number is again multiplied by two, as we can now view “Average Turnover by ..” and “Maximum Turnover by ..”

If we sort the output by some measure and display only the top n items, the number of options is again multiplied by the number of measures and n, since we can produce view such as “Top 3 Selling Countries” or “Top 5 Salespersons regarding margin”.

The number of variants could be multiplied by the 20+ drawing options (specified by the drawAs property).

Instead of having to provide a separate 4RP file for each variant, the implementation of pivot tables allows the creation of pivot table models containing a larger amount of dimensions and measures which will likely never be displayed as a whole. From the static setup, one can then select dimensions and measures for display via selection properties. By defining RTL expression for these properties, one can create the different variants described at runtime.

**Performance considerations**

A pivot table report is capable of handling large amounts of data without exhausting memory, as long as some constraints are met.

If tabular output is selected and other constraints are met, output is produced without delay and memory consumption is nearly constant. The processing time is proportional to the input length; for very large data sets it is advisable to aggregate the data in the database.

**Processing should be latency free**

A chart displays on a single page. As such, it displays only after all data has been processed.

When outputting a table, the output can span multiple pages. Data can be output during processing, a page can be returned well before all data is processed. Yet selecting this visualization type alone does not ensure latency free
processing; the data must be pre-sorted (See the `hierarchiesInputOrder` property). If the data is partially sorted, there can be periods of delay while the processor waits for the end of a block of data that needs to be sorted.

**Pre-sorting data reduces memory consumption**

Sorting is done in memory. Very large reports should therefore be run on (partially) pre-sorted data (See the `hierarchiesInputOrder` property). Output sorting is also done in memory (See the `outputOrder` property) and should be used with equal care. Suppressing the display of the fact rows (See the `displayFactRows` property) can significantly reduce memory consumption.

**Not sending duplicate values reduces processing time**

In the case that data is pre-sorted (see the `hierarchiesInputOrder` property), an optional, more compact form of data representation can be chosen that allows omitting dimension values that did not change from one row to another, thereby improving performance.

For example, after shipping the first fact row (“Afghanistan”,“1 North”...) all subsequent rows that contain measure for north Afghanistan need not ship these two dimensions anymore. When the first row of the next block (“Afghanistan”,“3 South”...) is reached only the value “3 South” needs to be reported once on the first row of the block. See Pivot Table Hierarchy Value on page 873.

**Sample pivot table reports**

Two sample pivot table reports are provided with the installation of Genero Studio. One report shows a static pivot table, while the other shows a dynamic pivot table.

The reports are located in the `Reports.4pw` project. This project can be found in the `My Genero Files/samples/Reports` directory. When you open the project, you can find the pivot table reports under `OrderReport > Designs`.

**Static pivot table sample report**

The report name is `StaticPivotTable.4rp`.

This sample report produces a table of customer data, grouped by customers and orders. The input is presorted. The dimension columns, the userid and orderid, are populated accordingly.

**Dynamic pivot table sample report**

The report name is `DynamicPivotTable.4rp`.

When this report is selected, a second dialog opens. From this dialog, you select the dimensions and measures included in the report, along with how to sort the measures.

The form `pivotdialog.4fd` defines the dialog. The module `pivotdialog.4gl` contains the Genero code driving the dialog, making use of the pivot table library `libpivot.4gl`.

A new record of type “PivotControlBlock” is shipped in the FIRST PAGE HEADER of the report. The structure of the record:

```plaintext
TYPE PivotControlBlock RECORD
  title STRING,
  drawAs STRING,
  dimensionsDisplaySelection STRING,
  measuresDisplaySelection STRING,
  outputOrder STRING,
  topN INTEGER,
  displayFactRows BOOLEAN,
  displayRecurringValues BOOLEAN,
  computeAggregatesOnInnermostDimension BOOLEAN,
  computeTotal BOOLEAN,
  computeCount BOOLEAN,
  computeDistinctCount BOOLEAN,
END RECORD
```
computeAverage BOOLEAN,  
computeMinimum BOOLEAN,  
computeMaximum BOOLEAN  
END RECORD

The definition of the record is located in the file globals.4gl.

The record is populated by a call to the dialog function promptForPivotDialogIfAny (from pivotdialog.4gl) in OrderReport.4gl in the sample application. This is the code fragment:

```plaintext
CALL promptForPivotDialogIfAny(filename) RETURNING retval, controlBlock.*
IF NOT retval THEN
    RETURN NULL
END IF
IF NOT fgl_report_loadCurrentSettings(filename) THEN
    EXIT PROGRAM
END IF
```

The function inspects the 4rp file. If it finds exactly one dynamically configurable pivot table, it will prompt the user to configure it (see pivotdialog.4gl for details).

The last step lies in DynamicPivotTable.4rp where the pivot table properties are defined as RTL expressions that initialize from the field values in control record. For example, the title property is initialized to "controlBlock.title".

![Figure 390: Properties of the Pivot Table element](image)

This figure displays the values of the properties for the pivot table element.

Create a pivot table
Create a pivot table in a report, add dimensions and measures, and set additional properties as needed.

1. Create a new, empty report.
2. In the Data View, associate your data schema.
3. From the Tool Box, drag and drop a Pivot Table to the report structure under the Page Root.
4. Add dimensions as Hierarchy elements. See Add a dimension to a pivot table on page 841.
5. Add measures under the Fact node. See Add a measure to a pivot table on page 842.
6. Arrange the dimensions and measures in the Structure View. See Arrange your hierarchies on page 843.
7. Set additional properties as needed for the Pivot Table. See Pivot table properties on page 841.
8. Save the report.

Examples are provided in the Reports project (Reports.4pw) as part of the samples directory. The reports are StaticPivotTable.4rp and DynamicPivotTable.4rp.

Note: You can create a dynamic pivot table by calling the Pivot library APIs.

**Pivot table properties**
The properties define the type of output and the display of the pivot table.

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

In addition to the attributes available for LAYOUTNODE, the PIVOTTABLE element includes the following properties:

- **Title** - The title of the pivot table.
- **Draw As**

  The **drawAs** property specifies the type of output that is rendered from the data. Depending on the type selected and the number of available dimensions, the rendering is delegated to the map chart, category chart, XY chart or table element. If the number of selected dimensions outnumbers the respective number in the selected visualization, the exceeding dimensions and measures are ignored. The values are assigned from left to right. For example, if a pivot table with 4 dimensions and 3 measures is drawn as a category chart with only 2 dimensions and one measure, then the chart will be drawn using the first two dimensions and the first measure from the pivot table's columns. Selecting Table causes the output to be drawn in tabular form, displaying all selected columns of the pivot table.

  Valid values for **drawAs** for Pivot Table: Area | Bar | Bar3D | Line | Line3D | Pie | Pie3D | Polar | Ring | Scatter | SpiderWeb | StackedArea | StackedBar | Step | StepArea | Table | TimeSeries | Waterfall | XYArea | XYStackedArea | XYLine. The default is a Table.

  - **Compute aggregates on the innermost dimension** - Whether to compute aggregates on the innermost dimension.
  - **Hierarchies input order** - The order by which the data is presorted.
  - **Display selection** - Which of the declared dimensions or measures to display.

    Not specifying a value is equivalent to selecting all declared dimensions. For example, given a table with three dimensions, not specifying this attribute is the equivalent of specifying a value of "0,1,2".

    - **Display recurring dimensions** - Whether recurring dimension values in the same column of table output should be displayed.
  - **Range Upper Bound** and **Range Lower Bound** - The highest and lowest values on the Y-axis.

**Add a dimension to a pivot table**
The dimension of a pivot table specifies the columns that the data is sorted by and the values against which aggregation occurs. Dimensions are represented by the Pivot Table Hierarchy Value.

When you add a Pivot Table, a single Pivot Table Hierarchy Value is added as a child. You can add more dimensions by dragging the Pivot Table Hierarchy Value object from the Tool Box and dropping it into the Pivot Table.

1. In the Report Structure, select the Pivot Table Hierarchy Value.
2. In the **Value** property, enter the column name for the dimension.

3. If you want to use a Numeric dimension, select the **Numeric Column** check box. If this check box is not selected, the dimension is a String.

4. Select the desired aggregation types, such as **Compute Count** or **Compute Average. Compute Totals** is selected by default.

5. In the **Title** property, enter a title for the dimension.

### Add a measure to a pivot table

A measure is used to aggregate numeric values in a pivot table.

### About this task

In the **Report Structure view**, measures are placed under facts. When you add a Pivot Table, a single **Pivot Table Fact** and single **Pivot Table Measure** are added as children. You can add more Facts and Measures by dragging the objects from the Tool Box and dropping them into the Report Structure.

1. In the **Report Structure**, select the **Pivot Table Measure**.
Figure 392: Measures in the Report Structure

You can now edit the Measure properties in the Properties View.

2. In the Value property, enter the column name for the measure.

3. If you want to use a Numeric measure, select the Numeric Column check box and enter a value in the Format field. If this check box is not selected, the measure is a String and you do not need to specify a format.

4. In the Title property, enter a title for the measure.

Arrange your hierarchies
Pivot table dimensions can be viewed as forming a hierarchy in the report structure. Arrange the hierarchies so that the values are triggered at the desired point in the data stream.

In order to minimize the volume of the data stream, the HIERARCHIES can be shipped sparsely, in the sense that not all hierarchy values need to be shipped for every row. Comparing the hierarchy values of two consecutive rows from right to left starting with the innermost dimension, any value can be omitted that is the same in both rows until reaching the first dissimilar column.

This example shows two equivalent pivot tables. The first table uses a flat representation while the second ships the values using a more space efficient method. It ships the value for “userid” only on changes of “userid” by placing the value in the corresponding trigger. The value remains the same for all orders of that customer. The same is done for “orderid” which changes its value for every order but remains the same for all items within the order. The arrows in the diagrams indicates the location of the hierarchies.
The HIERARCHY elements represent dimensions. An element represents both the declarative aspects of the column as well as the data value which is typically defined as an RTL expression.

**Pivot table elements and the Structure view**

The pivot table elements (pivot table, hierarchy, fact, and measure) must be organized within the Report Structure view.

A pivot table is constructed from four elements.

- The PIVOTTABLE element represents the table itself and all other elements are contained in it.
- The columns of the table are described by HIERARCHY elements (in case of a dimension column) or MEASURE elements (in case of a value column).
• MEASURE element are grouped in FACT elements.

A static pivot table uses predefined dimensions and measures when creating the report, whereas a dynamic pivot table determines the dimensions and measures at runtime.

**Figure 395: Pivot table elements in the Structure View**

This image shows a table with six dimensions (HIERARCHY elements) and four measures (MEASURE elements).

**Column definition**

This figure shows a pivot table definition on the left and a possible rendering of the table on the right.

- Dimensions and measures define the columns of the table.
- Not all defined dimensions and measures were selected for display.
- The title of the table and the selection of the dimensions is defined in the pivot table element.
- The title of the columns are defined in the dimension and measure elements.
- The selection of the measures is defined in the fact element.
Figure 396: Report columns

Row definition

The entity of dimension declaration followed by one fact element forms a row of a table. Typically one row is defined. It is placed in a trigger to get repeated for each input record.

Figure 397: Report row placed in a single trigger

It is allowed, but highly unusual, to specify the rows literally. All rows have to have exactly the same structure (number of dimensions and measures, types, and so on).
Figure 398: title

Related concepts

Organizing the report structure (the Report Structure view) on page 790
You use containers to make sure that the report elements print correctly.

Expressions in properties

A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

- RTL expressions overview on page 848
- Using the expression language on page 849
  - Operators
  - Conditional Expressions
  - Operands
  - 4GL Variables
  - Examples
- Using RTL classes on page 852
  - Classes
  - Members
  - Examples
- Using the PXML expression language on page 854
  - Units of Measure
  - Variables
• Functions
• Substituting 4GL variables for constants on page 855
• Expression examples on page 855

The RTL Expression language used in Report Writer closely follows the Java™ syntax for expressions and evaluation semantics.

• Arithmetic formulas can be used.
• Conditional expressions allow you to express IF/ELSE statements.
• Genero BDL variables can be used
• Functions from the Reporting API can be used.

Press the fx button to open the Expression Editing Window.

Related concepts
Changing a property value (The Properties view) on page 795
Select a report element in the report page (work area) or Structure View to display and edit the property values in the Properties View.

RTL Class Reference on page 945
Reference information for Report Template Language (RTL) classes.

Dimensions on page 977
The dimensions for units that are used in reports.

RTL expressions overview
RTL Expressions use methods and variables to define runtime values for the properties of a report element.

To define an object, values are specified for the object's properties. The value for a property of a report item can be a literal value, or it can be derived from an expression that is written using the RTL Expressions language. RTL Expressions allow you to define runtime values for any property of a report item, except for those properties that display a specific set of valid values in a dropdown listbox.

Click the Value field to enter an expression in a field of the Properties View. To enter longer values, or obtain hints while typing the entry, click the fx button of the desired property to open the RTL Expression Editor:

![Edit Expression dialog](image)

**Figure 399: Edit Expression dialog**

Press CTRL+SPACE and a Code Completion box appears, containing a list of the valid choices based on the context.
**RTL Expressions**

RTL Expressions:

- **are typed** - an expression is composed of items of different types and the expression returns a particular type. The properties in the Properties View are also typed. *Any expression entered as a value for a property must return the specified type.* For example, the text property of a WordBox has a type of String. So, any RTL expression for the text property is expected to yield a String.

- **closely follow the Java™ syntax** for expressions. The main difference is that the "new" keyword is not supported; it is not possible to create and subclass objects. RTL Expressions loosely follow the Java™ evaluation semantics (operator precedence, evaluation order, and so on).

- **can use 4GL variables.** The variable is converted to a specific type within the expression (See Conversion table).

RTL Classes provide member functions (methods) and member variables that you can use in your expressions. There is a class for each type of a report property.

**Important:** Numeric data types are limited to 15 significant digits. To avoid problems with totals in reports, do not calculate the value of an item that is part of a total using an RTL expression. Perform any calculations in the BDL program.

**Related concepts**

- **Report Element Properties** on page 876
  Each element in a report has associated properties. Values for these properties may be literal, or they may be RTL expressions.

- **Using the expression language** on page 849
  An expression is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

- **Using RTL classes** on page 852
  The Report Template Language (RTL) expressions are typed; that is, the expression must return a particular class.

**Using the expression language**

An expression is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

The RTL Expression language follows the Java™ syntax for expressions and evaluation semantics. Expressions can include these components:

- **Operators** on page 850
- **Conditional Expressions** on page 850
- **Operands** on page 851
- **4GL Variables** on page 851
**Operators**

**Table 216: RTL Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>Arithmetic: Modulus</td>
<td>x % 2</td>
<td>8</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>x * y</td>
<td>7</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>x / y</td>
<td>7</td>
</tr>
<tr>
<td>+</td>
<td>Addition</td>
<td>x + y</td>
<td>7</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>x - y</td>
<td>6</td>
</tr>
<tr>
<td>+</td>
<td>Concatenation</td>
<td>string + string</td>
<td>5</td>
</tr>
<tr>
<td>&lt;</td>
<td>Relational/Boolean: Less than</td>
<td>numeric &lt; 100</td>
<td>4</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less then or equal to</td>
<td>numeric &lt;= 100</td>
<td>4</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td>numeric &gt; 100</td>
<td>4</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
<td>numeric &gt;= 100</td>
<td>4</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
<td>numeric == 100</td>
<td>4</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
<td>numeric &lt;&gt; 100</td>
<td>4</td>
</tr>
<tr>
<td>!</td>
<td>Logical inverse (NOT)</td>
<td>!(x = y )</td>
<td>3</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Logical intersection (AND)</td>
<td>expr1 &amp;&amp; expr2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Logical union (OR)</td>
</tr>
</tbody>
</table>

The first column in the table describes the precedence order of the operators, listed highest to lowest. For example, the % modulus operator has a higher precedence than the * operator. Parentheses can be used to overwrite the precedence of operators.

**Conditional Expressions**

Conditional expressions allow you to express IF/ELSE statements.
Syntax:

```
Boolean-expression?expression-1:expression-2
```

The `?` operator indicates that this expression is conditional; the return value is dependent on the result of the Boolean expression. If the Boolean expression is TRUE, the first expression is the return value; otherwise, the second expression is the return value.

You can use the `null` keyword in the ternary conditional operator. The “if then” and “if else” operands can be either expressions or the keyword `null`. A property whose RTL expression yields “null” is not set. This is useful in cases where a property should be set only when a certain condition is met. Consider the case where the background color of a WORDBOX should be set to red when a variable value `x` drops below a value of 10. The expression for this would be:

```
x<10?Color.RED:null
```

**Operands**

Operands include:

- Literal values
- Other expressions
- 4GL Variables
- RTL Class Members
  - Objects
  - Methods (returning a single value)

A literal value for a string in an expression should be delimited by double quotes: “Test”.

**4GL Variables**

The data types of 4GL variables are taken into account when constructing expressions. For every 4GL variable an object is created that is either an instance of a FGLNumericVariable or an FGLString Variable. These objects hold the value of the 4GL variable, and at the same time they contain a member variable `value` which also contains the value. For this reason, it is legal to write "order_line.itemprice" in your expression as a shortcut for "order_line.itemprice.value". Both types of objects have these specific member variables defined:

- `value` - value of the 4GL variable
- `caption` - the title of the field
- `name` - the name of the variable
- `type` - the RTL type of the variable
- `isoValue` - the locale and formatting-independent representation of the value of the variable

The conversion table lists 4GL data types and the type into which they are converted within an RTL expression:

**Table 217: 4GL data types and the type into which they are converted within an RTL expression**

<table>
<thead>
<tr>
<th>4GL type</th>
<th>Corresponding RTL type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR, VARCHAR, STRING and TEXT</td>
<td>FGLStringVariable</td>
</tr>
<tr>
<td>DATE, DATETIME and INTERVAL</td>
<td>FGLNumericVariable</td>
</tr>
<tr>
<td>INTEGER, SMALLINT, FLOAT, SMALLFLOAT, DECIMAL and MONEY</td>
<td>FGLNumericVariable, limited to 15 significant digits.</td>
</tr>
<tr>
<td></td>
<td>The value of a number larger than 15 digits will be truncated, and the resulting number is rounded. For example, 12345678901234567 will be rounded to 123456789012346.</td>
</tr>
</tbody>
</table>
**Important:** To avoid problems with inaccurate totals on a report due to rounding, do not perform RTL arithmetic on either the individual values or the total value; calculate the value of the item in the BDL program instead, before passing the value to the report.

**Examples**

For the purpose of these examples, `order_line` has been replaced with `order`.

1. To add 10% to the `itemprice`:  
   ```  
   order.itemprice*1.10  
   ```
   The data item `order_line.itemprice` is converted to a Numeric type, so we can use the Numeric operators. In order to display the result of a Numeric expression in a WordBox, we must convert the result to a String. See Example 1 in the Using RTL Classes section.

2. Let's add 10% to the `item price` conditionally, depending on the value:  
   ```  
   order.itemprice<100?
   order.itemprice*1.10:order.itemprice  
   ```
   The condition in this Boolean expression tests whether the `itemprice` is greater than 100; if so, the value returned is 110% of the `itemprice`; otherwise, the value returned is simply the `itemprice`.

3. To set the font of a report item to italic when the 4GL variable `order_line.lineitemprice` exceeds $20, we must create an expression for the `fontItalic` property:  
   ```  
   order.lineitemprice>20  
   ```
   The property `fontItalic` is of type boolean, so any RTL expression that we use for that property must return a boolean value (TRUE/FALSE). Any of the relational operators yields a boolean, so the type of the returned value of this expression is a boolean (The expression will return TRUE if the `lineitemprice` exceeds 20).

**Note:** A numeric value by itself is not a boolean value as it is in some programming languages.

**Using RTL classes**

The Report Template Language (RTL) expressions are typed; that is, the expression must return a particular class. RTL expressions do not contain the primitive data types "byte", "short", "int", "long", "float", "double", "boolean" and "char". Instead, everything is expressed as objects. All methods are member functions. There are no global functions.

**Basic Object Classes**

There are object classes for each type of the report item properties. See the Properties documentation to identify the type of a specific property.

The basic object class types for properties are:

- **String** - contains methods used for all string operations
- **Numeric** - contains methods used for all numeric operations; the class has the precision of a double and the arithmetic operators are defined for objects of this type.
- **Boolean** - contains methods used for all logical operations
- **Color** - contains methods and static member variables related to color.
- **Enum** - a set consisting of a class for each property of this type; each class contains static member variables that provide a list of valid values for the corresponding property.
- **Date** - a class representing a Date value; contains methods for parsing and formatting strings.

**Members**

**Instance Member Methods**

Instance Member methods are called on an object instance. You can get an object instance by referencing a 4GL variable or by calling a method on another object. You can also use a literal value as an object instance. When you invoke the method, it is prefixed with the object instance name and the "." character.
Examples of instance methods are expressions like "order_line.customer_name.trim()". This is valid because the 4GL variable order_line.customer_name has a CHAR data type, which is converted within the RTL Expression to an object of the type String. And, the method trim() is a member function of a String object.

Methods always yield objects, so it is also legal to call methods on the return value of a method.

**Static Member Methods**

Static Member methods do not require an object instance. When you invoke the method, it is prefixed with the classname and the '.' character.

**Static Member Objects**

Static Member objects are member variables that do not require an object instance. The objects are prefixed with the classname and the '.' character. Examples of static member objects are expressions like "Color.RED" or "Numeric.PI".

**Examples**

1. This example concatenates the first and last names of a customer, using the trim method of the String class and the + operator:

   ```java
   order_line.shipfirstname.trim() + " " + order_line.shiplastname.trim()
   ```

   This expression prints the first name (trimmed of trailing blanks), a string consisting of a single space, and the last name (trimmed). Use double quotes instead of single quotes to delimit strings.

2. Parenthesis can be used to change the order of operations. For example:

   ```java
   (order_line.unitprice+10).toString()
   ```

   Parentheses are used to force the addition to be done prior to the conversion to a String.

3. This conditional expression used in the color property of the order_line.unitprice WordBox will change the color to red if the value is less than 20:

   ```java
   order_line.unitprice<20?Color.RED:Color.BLACK
   ```

   This expression specifies that the return value when the boolean expression is TRUE is the static member variable RED of the Color class, otherwise the return value is the static member variable BLACK.

4. It is legal to call methods on the return value of a method. For example, this is a valid expression:

   ```java
   order_line.customer_name.trim().toUpperCase().substring(1)
   ```

   In this example, the object order_line.customer_name is a BDL CHAR variable; this variable is assigned to the String type. The String method trim() is called first, returning the String object a. The method toUpperCase() is called for the object a, returning object b which will be in upper case. Finally, the method substring() is called for the object b, returning object c. If the customer_name is "Springs", the resulting object c is the string "PRINGS".

   There are many additional examples of expressions in the properties of report elements defined in the 4rp programs that are part of the GRWDemo project.
Using the PXML expression language
Genero Report Writer provides the PXML Expression language to define the value of a property that is of the PXML (dimension) type.

Tip: The type of each property is listed in the Properties page of the Report Writer documentation.

A PXML expression always yields a Numeric value. The value is expressed in units of measurement. It is legal, for example, for the value to be 10in. If no unit is specified, the unit is presumed to be points. When you specify a value in units, it is converted internally to its equivalent value in points.

Units of Measure
The most commonly used units are:

- point|pt
- pica|pc
- inch|in
- cm
- mm

For additional explanations and examples of the units that can be used, see Dimensions on page 977.

Variables
These variables can be used in any PXML expression to define the layout dynamically:

- max - the maximum extent of the current parent box
- min - the minimum extent of the current parent box
- rest - the remainder of the current parent box

Note: If the parent object is a propagating container and the child object does not fit in the remaining space for the parent object, the rest variable for the Y-Size property yields the same value as max (the child expands to the maximum extent of the parent). This forces the parent object to propagate and avoids overfullness.

For example, to center an element in its parent container you can use the max variable for these properties:

Table 218: Centering an element

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>max/2</td>
</tr>
<tr>
<td>y</td>
<td>max/2</td>
</tr>
<tr>
<td>anchorx</td>
<td>0.5</td>
</tr>
<tr>
<td>anchory</td>
<td>0.5</td>
</tr>
</tbody>
</table>

To force a page break after the object, use the rest variable in the Y-Size or X-Size properties.

Functions
The PXML Expression language has a 4GL-like syntax. The most commonly used functions are:

- max(valueA, valueB) - this is a function, not the variable listed in Variables on page 854!
- min(valueA, valueB)
- length(value)
- width(value)

For example, this expression uses the functions max and width:

```
max(10cm, width("HELLO"))
```
In this example, the report engine first calculates the width of the string "HELLO", taking the current font metrics into account. It then determines which is larger (10cm or the calculated width of "HELLO") and returns the larger value.

**Substituting 4GL variables for constants**
You may want your expression to depend on a data variable rather than on a constant string, such as "HELLO".

For this, we use RTL expressions embedded in curly braces.

**Note:** We are now mixing two languages. The content within the braces is RTL, the content outside the braces is a PXML expression.

The rules for embedding RTL in PXML are:

- wherever a numeric constant is allowed in PXML, you can insert an **RTL numeric expression** (enclosed in curly braces) instead.
- wherever a string constant is allowed in PXML, you can insert an **RTL string expression** (enclosed in curly braces) instead.

For example, consider this expression:

```
max(10cm, width("HELLO"))
```

This PXML expression contains one numeric constant ("10") and one string constant ("HELLO"). These constants can be replaced by data variables, enclosed in curly braces:

```
max({orderline.orders.shipcity.trim().length()}cm, width("{orderline.orders.shipcity}"))
```

**Tip:** It is good practice to use the `trim()` function to remove extra white space from a string expression.

For this expression to be legal, the variable `order_line.titlewidth` has to be of type Numeric, and the variable `order_line.title` has to be of type String.

**Note:** You cannot construct a dynamic expression where the function names (such as `max` or `width`) or the unit names (such as `cm`) are dynamic.

**Expression examples**
Examples of common expressions used by report designers.

**Related concepts**
- Using a page number string on page 800
  - For Page Number containers, use the **Text Expression** property to create a page number string, such as the standard "Page N of M".
- Custom keys on page 829
  - Enter any valid expression for the String value of a key property. This could be a substring or a concatenation of existing strings.

**Check a field for a value**
Check whether a field contains a value, is empty, or is null and act accordingly.

For this example, the data source includes the field `orderline.order.contact`, and the field is a STRING. The field either contains a value, is an empty string, or is null.

To test whether it contains a value, you write an expression:

```
orderline.order.contact.trim().length()>0
```

This expression will evaluate to either TRUE or FALSE.
To explicitly test for an empty string, there are two options:

```java
orderline.order.contact.trim().length() == 0
```

```java
orderline.order.contact.isEmpty() == TRUE
```

To explicitly test for null:

```java
orderline.order.contact.isNull() == TRUE
```

**Use in the Visibility Condition property**

Expressions that evaluate to TRUE or FALSE can be used in the Visibility Condition (visibilityCondition) property. If the expression evaluates as TRUE, then the instance of the element will appear in the report. If the expression evaluates as FALSE, the instance of the element (to include all its children, i.e. the entire element tree) is removed from the report. If you are using relative positioning, all sibling elements after this element in the report structure shift accordingly, reclaiming the space that the element would have occupied.

**Use in the Text property**

You can use these expressions when defining the Text (text) property. The text property specifies the text to be drawn.

```java
orderline.orders.contact.trim().length() > 0 ? orderline.orders.contact : ""
```

In this expression, the expression evaluates to TRUE when the length of the trimmed field is greater than zero and the field value is printed (to include any leading or trailing spaces). If the length is not greater than zero, the field is identified as not having a value and an empty string is printed; the vertical allocated space for that field remains in the report.

An alternate expression could simply be:

```java
orderline.orders.contact.trim()
```

**Tip:** When you use a character type with a fixed length for a field (such as CHAR (N)), you typically need to add .trim() or .trimRight() to remove trailing spaces. You can avoid this by using the STRING data type. With the STRING data type, the value is not padded with trailing spaces unless trailing spaces are explicitly set.

```java
DEFINE field1 CHAR(5),
DEFINE field2, field3 STRING
LET field1 = "ABC"   -- you end up with "ABC 
LET field2 = "ABC"   -- you end up with "ABC"
LET field3 = "ABC " -- you end up with "ABC ", as explicitly specified
```

**Change the display based on a condition**

Change the way your report displays depending on a condition in the RTL expression.

A condition in RTL uses the following syntax:

```java
Boolean-expression?expression-1:expression-2
```

For this example, the data source includes the field `lineitemprice`, and the field is a NUMERIC.

The report includes a Decimal Format Box that displays the line item price. You want the line item price to display red if below $100, and black if $100 or above. For the Color property, you enter the following expression:

```java
lineitemprice < 100 ? Color.RED : Color.BLACK
```
The Rating column for this report shows an Image Box displaying a "thumbs up" icon. You want this image to turn upside down if the line item price is below $100, turn to the right if the line item price is between $100 and $500, and display unturned if the line item price is $1000 or above. For the Layout Direction property, you enter the following expression:

```java
lineitemprice<100?LayoutDirection.UpsideDown:lineitemprice<1000?
LayoutDirection.TurnRight:LayoutDirection.Unturned
```

The report displays as in the following figure.

<table>
<thead>
<tr>
<th>Stock report</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Store #</strong></td>
</tr>
<tr>
<td>101</td>
</tr>
<tr>
<td>101</td>
</tr>
<tr>
<td>101</td>
</tr>
<tr>
<td>101</td>
</tr>
<tr>
<td>101</td>
</tr>
<tr>
<td>101</td>
</tr>
<tr>
<td>101</td>
</tr>
<tr>
<td>101</td>
</tr>
<tr>
<td>101</td>
</tr>
</tbody>
</table>

**Figure 401: Stock Report**

**Display an image based on a variable**

Check the value of a variable and display the image specified by the variable.

For this example, all images are contained in `/images/database`, which is in the same directory as the report design document. The data source includes the field `orderline.product.prodpic`, and the field is a STRING.

In the Image Box, for the Location property, you enter the following expression:

```java
"/images/database/"+orderline.product.prodpic.trim()
```
This expression evaluates to a STRING value that specifies the image to be displayed.

**Size a column based on data**

Base the size of a column on the size of the translated text displayed in it.

For this example, you want the **Price** column to be wide enough to fit the column heading and the data displayed in it. You also want to translate the column heading into the system language.

To translate the heading, include the `translate()` function in the **Text** property, as in the following expression:

```
"Price".translate()
```

To make the column wide enough to contain the translated text and data, set the **X-Size** property to the following expression:

```
max(width("{"Price".translate()}"),width("-0,000,000.00"))
```

**Code elements:**

- `max()` determines that **X-Size** is the larger of the two comma-separated numbers.
- `width("{"Price".translate()}")` is the width of the translated label. The string "Price".translate() is an RTL expression within a PXML expression, and therefore must be within curly braces.
- `width("-0,000,000.00")` is the maximum width of the data within the column.

The column width will be the maximum width of the data displayed in it or the localized column title, whichever is wider.

**Format a number within a string**

Include a formatted number in a text string.

If you want to display a formatted number on its own, you use a **Decimal Format Box**. However, if you want to include additional text with the number, you must use a **Word Box** and format the number within the string.

For this example, you set the **Text** property to the following expression:

```
"Order "+orderline.orders.orderid.format("---#")+" from "+orderline.orders.orderdate.trim()
```

The `format()` method converts the Numeric value to a String representation. This prints text on the report in the format "Order X from Date", for example, "Order 5 from 2014/06/24".

### Debugging your Report Design Document

Tips to help you debug issues you may have with your report design.

**Using a Background Color**

To check for overlap of an object on a report design, or to simply visually see where an object falls on your report, set the **Background Color** property of the object.

![Figure 402: Setting the background color](image-url)
Using GREDEBUG environment variable

Set GREDEBUG to check overfull boxes. Warning messages regarding any overfull boxes are written to standard output.

![Overfull Box message](image)

Figure 403: Overfull Box message

Report Designer Reference

Reference information for Report Designer.

- Report Design Elements (The Toolbox) on page 859
- Report Element Properties on page 876
- Bar Codes on page 911
- RTL Class Reference on page 945
- Dimensions on page 977
- Report Writer preferences on page 982

Report Design Elements (The Toolbox)

The Toolbox contains report object that can be placed on a report design document.

- Simple Containers
  - Horizontal Box (Layout Node)
  - Vertical Box (Layout Node)
- Propagating Containers
  - Horizontal Box (Mini Page)
  - Vertical Box (Mini Page)
  - Page Root (MiniPage)
  - Stripe (MiniPage))
- Drawables
  - Word Box
  - Word Wrap Box
  - HTML Box on page 862
  - Decimal Format Box on page 862
  - Page Number (PageNoBox) on page 864
  - Image Box on page 865
  - Table on page 867
- Business Graphs
  - Map Chart on page 869
  - Map Chart Item on page 870
  - Category Chart on page 870
  - Category Chart Item on page 871
  - XY Chart on page 871
  - XY Chart Item on page 871
  - Pivot Table on page 872
  - Pivot Table Hierarchy Value on page 873
The Tool Box view on page 781
The Tool Box view provides report elements to place on a report design document.

Related concepts
The Tool Box view on page 781

Simple Containers
The Simple Containers section of the toolbox contains those containers that do not propagate. These containers are used to group and organize objects on a page.

Horizontal Box, Vertical Box (Layout Node)

Container: A Layout Node is a rectangular area in the Report Design page. A Layout Node does not propagate; the content is not allowed to overflow the container. As a result, a Layout Node can be used for content that should be kept together on the page.

The Horizontal Box is a Layout Node with the layout direction property set lefttoright; elements in the box are laid out across the page. The Vertical Box is a Layout Node with the layout direction property set topbottom; elements in the box are laid out down the page.

Properties
Select the object on the Report Design page to display its properties in the Properties View.

You can change the object's default appearance by setting the values of these properties:

name, color, bgColor, fontName, fontSize, X-SizeAdjustment, Y-SizeAdjustment, fontBold, fontItalic

Some specific properties allow you to define borders, margins, and padding for the boxes:

Margin width properties, Border width properties, Padding width properties, Border style properties, Border color properties

The x-Size and y-Size properties specify the INNER size of the box. Adding borders, for example, increases the overall size.

Layout Nodes can be used as a container for content that must print at a specific part of the MiniPage, a page header, for example, by setting its section property.

The clip property can be used to clip the object box and its content along the sides. This property applies to all layout nodes.

Other properties have values that are derived from the type and position on the page, adjust automatically if the object is moved or re-sized, and need not be changed manually:

type, baselineType, layoutDirection, swapX, alignment, scaleX, scaleY, x, y, anchorX, anchorY, floatingBehavior
Propagating Containers (Mini Pages)
The Propagating Containers section of the toolbox contains those containers that propagate. If the container fills, a copy is generated and the extra content overflows to the copy.

Mini Page
Container: Mini Page. This container formats the report page into lines and columns.
A Mini Page is a propagating box. The boxes can handle unknown amounts of material; if the box is full, a copy is made and the leftover material flows to the copy or copies, as needed. A MiniPage cannot be used as a container for a page header, page footer, or separator.

Page Root (Mini Page)
Container: Mini Page. Page Root is the recommended base container when you start creating a report. It is a Mini Page with the height and width properties set to maximum. The container propagates; if it is full, a copy is generated and the extra content overflows to the copy.

Stripe (MiniPage)
Container: Mini Page. This container has the y-size property set to "max". This container is recommended for content that stretches horizontally across the report page (lines in a report, for example.) The container propagates; if it is full, a copy is generated and the extra content overflows to the copy.

Horizontal Box, Vertical Box (Mini Page)
Container: Mini Page. These containers have their orientation (layoutDirection property) set to display the box content horizontally (lefttoright) or vertically (toptobottom). The containers propagate; if a container is full, a copy is generated and the extra content overflows to the copy.

Properties
Select the object on the Report Design page to display its properties in the Properties View.
These properties are specific to Mini Page:
Hide PageHeader OnLastPage, Hide PageFooter OnLastPage
Additional properties are inherited from Propagating Box and Layout Node. The property floatingBehavior allows you to specify whether the parent container resizes itself so that this Mini Page object is enclosed in the parent.

Drawables
The Drawables section of the toolbox contains a variety of objects that can hold static or dynamic values, such as text, numbers, HTML snippets, page numbers, images, and tables.
Word Box and Word Wrap Box
The Word Box and Word Wrap Box are layout objects for the display of text.

Word Box
Word Box (WordBox type) is a layout container, found in the Drawable group in the Tool Box view.
Use this object for a specified chunk of text, which uses the current font.
These properties are specific to Word Box:
trimText, underline, strikethrough, fidelity, localizeText

Word Wrap Box
Word Wrap Box (WordWrapBox type) is a layout container, found in the Drawable group in the Tool Box view.
This object is like a WordBox with paragraphs of uniform text.
These properties are specific to Word Wrap Box:
trimText, indent, fidelity, localizeText

**Properties**
Select the object on the Report Design page to display its properties in the Properties View.
The text property specifies the string to be displayed in the WordBox or Word Wrap Box.
You can set the textAlignment property for a Word Box or Word Wrap Box to left, right, or center. The alignment does not influence page break positions even if the indent property is set to some value. For Word Wrap Boxes, the textAlignment property can also be set to justified.
The localizeText property enables the localization of text content in Word Boxes and Word Wrap Boxes.
Additional properties are inherited from Layout Node.

**Tip:**
- Don't set the Y-Size (height) property on a Word Wrap Box, because the element should typically grow in accordance with its content. If you set a fixed Y-Size, you prevent that automatic enlargement.
- Use relative positioning for Word Boxes and Word Wrap boxes. Relative positioning means the text adapts to changes in font and page setting, as well as to translation.
- A Word Wrap Box can span pages. When the available vertical space for a Word Wrap Box is not sufficient to display the entire text, the box propagates the exceeding content to additional boxes, with the same behavior as a propagating Mini Page.

**Entering text and line breaks**
The text value can be edited directly in the report design document. Double-click the object to place the input cursor in the box, and type your text. The Text property and the layout are updated on each keystroke.

**Note:** Line breaks display in the Text property as a newline character, \n.
You can also enter an RTL expression in the Text property. To force a line break in an RTL expression, use (10).toChar(), for example:

```
orderline.account.addr1+(10).toChar()+orderline.account.city
```

**Decimal Format Box**
The Decimal Format Box is a layout object for the display of decimal numbers.
Decimal Format Box (DecimalFormatBox type) is a layout container, found in the Drawable group in the Tool Box view.
Use a Decimal Format Box for decimal numbers. It has features that make it possible to parse and format numbers in any locale, including support for Western, Arabic, and Indic digits. It supports different kinds of numbers, including integers (123), fixed-point numbers (123.4), scientific notation (1.23E4), percentages (12%), and currency amounts ($123). All of these can be localized. The value of the number is limited to 15 significant digits.

**Properties**
Select the object on the Report Design page to display its properties in the Properties View.

**format**
The value of the text property can also be edited directly in the report design document; double-click the object and the input cursor is placed in the text. The layout is updated on each keystroke.

**HTML Box**
An HTML Box displays an image of an HTML document in the report.
HTML Box (HTMLBox type) is a layout container, found in the Drawable group in the Tool Box view.
Note: The content of an HTML box cannot span pages.

Properties
Select the object on the Report Design page to display its properties in the Properties View.
Use the Location property to specify the file name and path of the HTML document.

Embedding HTML
To embed a document, use a URL type that allows encoding the data in the body of the URL text. The full syntax of data URLs is:
\[
data: [<MIME-type>] [; charset=<encoding>] [; base64], <data>
\]
See data URI scheme (Wikipedia link) for a complete description of the concept and the syntax.
For our purposes, it is sufficient to support a simplified subset that omits the charset and assumes that characters are encoded in UTF-8. Image data is always encoded in base64 (Wikipedia link) while other data such as HTML content is typically “Percent encoded” (Wikipedia link).
For HTML content, a URL has the form “data:text/html,<data>". To embed HTML, use the data protocol syntax in the Location property of the HTMLBOX element:
\[
data:text/html, followed by the percent encoded data of the html document
\]
To automatically construct this URL, click the ... button for the Location property. Choose the file and select the Embed in document check box:

![Figure 404: Embed in document checkbox](image)

Populating HTML content from text variables
Data in text variables is typically input into a database using a TextEdit form field, with the textFormat style attribute set to html. See the Presentation Styles topic in the Genero Business Development Language User Guide for more details.
Enter the code in the Expression Editor for the Location property of the HTML Box, including the name of the text variable.
Figure 405: RTL Expression

To encode the data using percent encoding, use the function `String.urlEncode()`.

**Page Number (PageNoBox)**

A Page Number element is a layout object for the display of page numbers.

Page Number (PageNoBox type) is a layout container, found in the Drawable group in the Tool Box view.

In order to provide for virtual pages (multiple logical pages on one physical page) a `pageName` property can identify the logical page.

Specific functions allow you to calculate an expression for the page number, such as `Page N of M`, using the property `textExpression`. If a value is provided for `textExpression`, the `pageName`, the `pageNoOffset`, and `pageNoFormat` properties are ignored.

If values for either the `textExpression` property or the `text` property are not set, a default length is calculated. See Using page numbers on page 799.

**Properties**

Select the object on the Report Design page to display its properties in the Properties View.

These properties are specific to PageNoBox:

- `pageName`, `pageNoOffset`, `pageNoFormat`, `textAlignment`, `textExpression`

Additional properties are inherited from Word Box.

**PDF Box**

A PDF Box displays one or more pages from a PDF document in the report.

PDF Box (PDFBox type) is a layout container, found in the Drawables group in the Tool Box view. The PDF Box propagates if the PDF has more than one page.

**Properties**

Select the object on the Report Design page to display its properties in the Properties View.

The `Location` property specifies the file name and path of the PDF document.

The `Password` property is used for password-protected documents.

The `Page Ranges` property specifies the pages to include.

The `cropRightWidth`, `cropBottomWidth`, `cropLeftWidth`, and `cropTopWidth` properties specify the amount to be cropped from the margins.
**Embedding PDFs**

To embed a document, use a URL type that allows encoding the data in the body of the URL text. The full syntax of data URLs is:

```
data:[<MIME-type>][;charset=<encoding>][;base64],<data>
```

See data URI scheme (Wikipedia link) for a complete description of the concept and the syntax.

For our purposes, it is sufficient to support a simplified subset that omits the charset and assumes that characters are encoded in UTF-8. Image data is always encoded in base64 (Wikipedia link) while other data such as HTML content is typically “Percent encoded” (Wikipedia link).

For PDF content, a URL has the form “data:application/pdf;base64,<data>. To embed PDFs, use the data protocol syntax in the Location property of the PDFBOX element:

```
data:application/pdf;base64, follo```

To automatically construct this URL, click the ... button for the Location property. Choose the file and select the Embed in document check box:

![Image of Embed in document checkbox]

**Figure 406: Embed in document checkbox**

**Displaying PDFs depending on a variable**

In this example, all PDFs are stored in the directory C:/My Docs/pdfs/, and the PDF name is determined by the field orderline.product.pdf. Enter the following in the Expression Editor for the Location property:

```
"file:///C:/My Docs/pdfs/"+orderline.product.pdf.trim()
```

**Related concepts**

- **location (Location)** on page 888
  The location property specifies the location of a document, such as an image, HTML file, or PDF.

- **Image Box**
  An Image Box displays images in GIF, JPEG, PNG, BMP, WBMP, PDF, and SVG formats.

  Image Box (ImageBox type) is a layout container, found in the Drawable group in the Tool Box view.

  The images are cached on a per document basis. The default resolution is the current screen resolution.

  **Note:** The Image Box renders only the first page of a PDF document. If you want to embed the entire PDF, use the PDF Box.
Properties

Select the object on the Report Design page to display its properties in the Properties View.

These properties are specific to Image Box:

- **Location**, **Intended Resolution**, **Fill**

Use the **Location** property to specify the location of the image to be rendered.

If you want to preserve the aspect ratio of an image, set the value of *either* length (y-size) or width (x-size), allowing Report Writer to calculate the corresponding value. If you set both properties, the resulting image appears distorted.

**Note:** Beginning with version 2.4x, the **tile** property is replaced by a new property, **fill**. If you open a document containing the tile property, it is automatically replaced by **fill**.

Embedding images

To embed a document, use a URL type that allows encoding the data in the body of the URL text. The full syntax of data URLs is:

```
data: [<MIME-type>] [;charset=<encoding>] [;base64], <data>
```

See [data URI scheme](https://en.wikipedia.org/wiki/Data_URI_scheme) for a complete description of the concept and the syntax.

For our purposes, it is sufficient to support a simplified subset that omits the `charset` and assumes that characters are encoded in UTF-8. Image data is always encoded in base64 (Wikipedia link) while other data such as HTML content is typically “Percent encoded” (Wikipedia link).

For images, the simplified URL has the form "data:/text:xxx;base64,<data>" where `xxx` is one of “png”, “jpg”, “gif” or “bmp”. For an image, the syntax is:

```
data:image/png;base64, followed by the base 64 encoded bitmap data of the image
```

To automatically construct this URL, click the ... button for the **Location** property. Choose the file and select the **Embed in document** check box:

![Figure 407: Embed in document checkbox](image)

**Figure 407: Embed in document checkbox**

Populating images from blob variables

If you drag a BYTE variable from the Data View onto the Report Design, an Image Box object is created and the **Location** property is filled with this formula:

```
data:image/jpg;base64,"+imageblob
```

where `imageblob` is the name of the blob

You can also enter the blob variable name using the Expression Editor for the **Location** property:
Figure 408: Entering blob variable name in RTL Expression editor

Note: Since a blob variable may contain images of various types, the implementation ignores the image type declared in the formula and looks at the encoded data to determine the image type. This formula also works for blobs of png type.

SVG images

Using SVG images instead of bitmap images can substantially reduce the document size. When providing the SVG content from a 4GL variable, use the mime type image/svg so that the url looks something like data:image/svg,... when read from a string variable and data:image/svg;base64,... when read from a BLOB. The currently supported SVG version is "1.2 Tiny" (See [http://www.w3.org/TR/SVGTiny12](http://www.w3.org/TR/SVGTiny12)).

Fallback image

A fallback image is displayed if the requested image is not found. To specify a fallback image, set the GRE_DEFAULT_IMAGE_URL environment variable to the image URL. The image URL can be a relative URL, which resolves relative to the location of the form design (.rpt) file.

Related concepts

Display an image based on a variable on page 857

Check the value of a variable and display the image specified by the variable.

Table

A table object has the ability to display data in columns and rows.

Table (Table type) is a layout container, found in the Drawable group in the Tool Box view.

When you drag a table onto a report, it creates a table with a default of two rows and three columns. Of the two rows, one is a header row (Any Page Header) and one is a body row (Body). You can add and remove rows and columns, size the table or its components, merge columns, define its borders and padding and much more.

Table Structure

In the Report Structure, you can view the table structure.
Figure 409: Table Object as viewed in the Report Structure

The top-level element is the Table element. It contains three child elements, which make up the parts of the table. These elements are the column definitions, the head, and the body.

The column definitions (or coldefs) define the basic properties for the columns in this table object. These properties include settings for padding, width, and alignment.

The head section contains the heading rows.

The body section contains the body rows.

**Table Properties**

Properties specific to the table involve rules, borders, and padding.

A *rule* refers to a line that separates two rows or two columns. Rule-related properties include Rule, Rule Color, Horizontal Rule, and Vertical Rule.

The *border* refers to the border around the table. Border-related properties include Border, Border Color, Top Border, Left Border, Bottom Border, and Right Border.

*Padding* refers to the space between a cell boundary and the value contained within. Padding-related properties include Padding, Horizontal Padding, and Vertical Padding.

**Column Properties**

Column properties are specific to the column selected. Any column property set overrides the same property set for the table.

You can set the padding for the cells of a column. Padding-related properties include Padding, Horizontal Padding, and Vertical Padding.

You can set the width of a column using Proportional Width or Fix Width.

You can set the alignment of a value within the cells of a column with the Horizontal Alignment and Vertical Alignment properties.
Cell Properties

Cell properties are specific to the cell selected. Any cell property set overrides the same property set for the table or the column.

You can set the padding for the cells of a column. Padding-related properties include Padding, Horizontal Padding, and Vertical Padding.

You can set the alignment of a value within the cells of a column with the Horizontal Alignment and Vertical Alignment properties.

You can merge cells by setting the Column Span property.

Demos

The Reports demo project includes two report design documents showing reports that include a table object.

- The TableDemo.4rp shows a report design document with a simple table containing five columns and two rows. One row is the Any Page Header row, while the other row is the Body row.
- The GroupedTableDemo.4rp shows a report design document where the table is more complex, with several header and body rows. For each header row, the section property specifies whether it is the First Page Header, Any Page Header, and so on. Each body row is created with a purpose - to show a row of data, to show the sum of a group of rows, and so on. Some of the cells in the summary rows span columns. Triggers are used to determine when each of the body rows is output to the table in the report.

Related concepts

Working with tables on page 819
A table object displays data in columns and rows.

Business Graphs

The Business Graphs section of the toolbox contains a variety of chart objects (map charts, category charts, XY charts) and pivot table objects.

Map Chart
The MapChart layout object allows you to create a graph that has one set for values to be mapped, grouped together by a specific key.

Map Chart (Mapchart type) is a layout container, found in the Business Graphs group in the Tool Box view.

The MapChart layout object is defined by the MapChartDrawAs class.

The MAPCHART element defines the header for an abstract map dataset that can be used for creating a variety of one dimensional graphs such as pie charts. The map items are defined using the ITEM element. The resulting chart is drawn automatically. See Working with Business Graphs for additional information.

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

title, valuesTitle, keysTitle, drawAs, fidelity, drawLegend, drawLabels

Valid values for drawAs for Map Chart: Bar|Bar3D|Pie|Pie3D|Ring|Table|SortedTable|AggregatedTable. The default is a Pie.

The fidelity property applies only if the chart is drawn as a table (drawAs="Table").

The drawLegend and drawLabels properties have been added to customize the appearance of the plots. The option to remove the legend is useful when many charts are drawn next to each other in a document; the option can be used to make the charts share a single legend by specifying the legend only on one of the charts.

Related concepts

Map Chart Item on page 870
A Map Chart Item defines the data value items for a Map Chart.

Map charts on page 822
The map chart allows you to create a graph that has one set for values to be mapped, grouped together by a specific key.

Related tasks
Create a map chart on page 824
Map charts have one key value and one data value. For example, you could map the revenue distribution by customer.

Map Chart Item
A Map Chart Item defines the data value items for a Map Chart.

Map Chart Item is found in the Business Graphs group in the Tool Box view.

A Map Chart Item defines the data value items for a Map Chart.

Properties
Select the object on the Report Design page to display its properties in the Properties View.

key, value, name, color

The data is grouped by the key property, which must be a String. The chart displays the total of the value property, which must be numeric, for each key.

The name of the report item is displayed in the Structure view.

The color property gives each slice a specific color. When a color is specified for a particular key in one chart, then the same color will be used for that key in other charts too, unless specified otherwise. If different colors are specified for the same key, the most recent value is used. If the same color is specified for a number of different keys, only one of these keys will be painted with the specified value; the other slices will be painted with interpolated values. Charts may use gradients, shading, or translucency with the colors specified.

Category Chart
A Category Chart defines the header for an abstract category dataset that can be used for creating a variety of two dimensional charts.

Category Chart (Categorychart type) is found in the Business Graphs group in the Tool Box view.

The CATEGORYCHART element defines the header for an abstract category dataset that can be used for creating a variety of two dimensional charts such as category charts. The categories are defined by the CATEGORY element and its "key" attribute, which has to be unique within a CATEGORYCHART. Within a CATEGORY, CATEGORYITEMS define the values within the category. Within one category, the "key" values of individual CATEGORYITEM elements has to be unique. The resulting chart is drawn automatically. See Working with Business Graphs for additional information.

Properties
Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

title, valuesTitle, keysTitle, drawAs, fidelity

Valid values for drawAs for Category Chart: Area|Bar|Bar3D|Line|Line3D|SpiderWeb|StackedBar|StackedArea| Waterfall|Table|SortedTable|AggregatedTable. The default is a Bar.

Note: If you select Waterfall as the chart type, the value in the last category of the data set should be (redundantly) specified as the sum of the items in the preceding categories - otherwise, the final bar in the chart will be incorrectly plotted. At the present time, the chart can only have one category.

The fidelity property applies only if the chart is drawn as a table (drawAs="Table").

Related concepts
Category Chart Item on page 871
A Category Chart Item defines the data value items for a Category chart.

**Category charts** on page 822
The category chart allows you to create a graph that has two keys for each set of values.

**Related tasks**

- **Create a category chart** on page 826
Category charts have two key values and one data value. For example, you could map revenues by area and customer.

**Category Chart Item**
A Category Chart Item defines the data value items for a Category chart.

Category Chart Item is found in the Business Graphs group in the Tool Box view.

A Category Chart Item defines the data value items for a **Category Chart**.

**Properties**
Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

- **category Key**, **key**, **value**, **name**

  The categories for the data are defined by the `categoryKey`. The data in each category is grouped by the `key` property, which must be a String. The chart displays the total of the `value` property for each category divided into groups by key. The `categoryKey` and `key` must be a String, while the `value` must be Numeric.

**XY Chart**
An XY Chart defines the header for an abstract XY dataset that can be used for creating a variety of XY-Plots such as line or scatter plots.

XY Chart (XyChart type) is found in the Business Graphs group in the Tool Box view.

The XYCHART element defines the header for an abstract XY dataset that can be used for creating a variety of XY-Plots such as line or scatter plots. The XY data is defined by XYITEM elements. The resulting plot is drawn automatically. See **Working with Business Graphs** for additional information.

**Properties**
- **title**, **xAxisTitle**, **yAxisTitle**, **drawAs**, **fidelity**

  Valid values for `drawAs` for XY Chart: Area|Line|Polar|Scatter|StackedArea|Step|StepArea|TimeSeries|Table|SortedTable. The default is Line.

  The `fidelity` property applies only if the chart is drawn as a table (drawAs="Table").

**Related concepts**

- **XY Chart Item** on page 871
An XY Chart Item defines the data value items for an XY Chart.

**XY charts** on page 823
The XY chart allows you to create a graph that maps a series that has two values, as an XY-plot.

**Related tasks**

- **Create an XY chart** on page 827
XY charts have two data values. For example, you could map quantity sold against discount percentage.

**XY Chart Item**
An XY Chart Item defines the data value items for an XY Chart.

XY Chart Item is found in the Business Graphs group in the Tool Box view.

It defines the data value items for an **XY Chart**.
Properties
Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

seriesTitle, x, y, name
The seriesTitle defines the caption for the series of values being charted. There can be more than one series.
The x property defined the values for the x axis, while the y property defines the values for the y axis. Both x and y must be Numeric.

Pivot Table
The PIVOTTABLE element is the enclosing element of an abstract pivot dataset that can be used for creating a variety of multidimensional outputs such as tables and charts.

Pivot Table is found in the Business Graphs group in the Tool Box view.
The resulting data is drawn into a box defined by X-Size and Y-Size. If these are not defined, the view expands to whatever space it can claim in the parent.
The PIVOTTABLE element is part of the Business Graphs group in the Report Designer Tool Box.
Depending on the visualization type (specified by the drawAs property), the output may span several pages. For these visualization types (such at Tables), the enclosing containers should support propagation.

Properties
Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

In addition to the attributes available for LAYOUTNODE, the PIVOTTABLE element has the following properties:

- title
- drawAs

The drawAs property specifies the type of output that is rendered from the data. Depending on the type selected and the number of available dimensions, the rendering is delegated to the map chart, category chart, XY chart or table element. If the number of selected dimensions outnumbers the respective number in the selected visualization, the exceeding dimensions and measures are ignored. The values are assigned from left to right. For example, if a pivot table with 4 dimensions and 3 measures is drawn as a category chart with only 2 dimensions and one measure, then the chart will be drawn using the first two dimensions and the first measure from the pivot table's columns. Selecting Table causes the output to be drawn in tabular form, displaying all selected columns of the pivot table.

Valid values for drawAs for Pivot Table: Area | Bar | Bar3D | Line | Line3D | Pie | Pie3D | Polar | Ring | Scatter | SpiderWeb | StackedArea | StackedBar | Step | StepArea | Table | TimeSeries | Waterfall | XYArea | XYStackedArea | XYLine. The default is a Table.

- fidelity

The fidelity property applies only if the chart is drawn as a table (drawAs="Table").

- computeAggregateInnermostDimension
- hierarchiesInputOrder
- displaySelection
- displayRecurringDimensions

Related concepts
Working with pivot tables on page 836
The pivot table element defines a table element with fixed roles and types for its columns.

**Pivot Table Hierarchy Value**
The HIERARCHY elements represent dimensions. An element represents both the declarative aspects of the column as well as the data value which is typically defined as an RTL expression.

Pivot Table Hierarchy Value is found in the Business Graphs group in the Tool Box view.

HierARCHy elements are child elements of PIVOTTABLE and need to be located before a FACT element containing the measures of the row.

The HIERARCHY element is part of the Business Graphs group in the Report Designer Tool Box.

**Properties**
Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

- isNumeric
- title (default is the empty string)
- format (default ---,---,---,--&&)
- value
- enumValues
- computeTotal
- computeCount
- computeDistinctCount
- computeAverage
- computeMinimum
- computeMaximum

**Related concepts**
- Working with pivot tables on page 836
The pivot table element defines a table element with fixed roles and types for its columns.

- Arrange your hierarchies on page 843
Pivot table dimensions can be viewed as forming a hierarchy in the report structure. Arrange the hierarchies so that the values are triggered at the desired point in the data stream.

**Pivot Table Fact**
Together with the HIERARCHY elements that can precede it, the FACT element defines a table row.

Pivot Table Fact is found in the Business Graphs group in the Tool Box view.

**Properties**
Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

- outputOrder
- displaySelection
- displayFactRows (default true)
- topN

**Related concepts**
- Working with pivot tables on page 836
The pivot table element defines a table element with fixed roles and types for its columns.

*Pivot Table Measure*
A MEASURE element represents both the declarative aspects of the column as well as the data value. Pivot Table Measure is found in the Business Graphs group in the Tool Box view. The data value is typically defined as an RTL expression. MEASURE elements are child elements of FACT elements. The MEASURE element is part of the Business Graphs group in the Report Designer Tool Box.

**Properties**
Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

- **isNumeric**
- **title** (default is the empty string)
- **format** (default ---,---,---,--&.&&)
- **value**

**Related concepts**
- [Working with pivot tables](#)
- [Add a measure to a pivot table](#)

**References**
The References section of the toolbox provides two objects: a Reference Box and an Info Node. These two objects typically work together in a report design.

*Reference Box*
A Reference Box allows you to create layout-dependent text output, such as "Total from previous pages: num". Reference Box (ReferenceBox type) is a layout container, found in the References group in the Tool Box view.

Because the space to be allocated may not be known until the report is run, make sure there is enough space available to display any possible text. Use the **text** property to provide an example, based on the underlying data type of the InfoNode object. This is only used to determine the maximum space to set aside. For example:

- Data types that are numeric - "000,000,000.00"
- Data types that are strings, for example CHAR(8) - "MMMMMMMMM"

**Properties**
Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

This object works in conjunction with an Info Node object, and its specific properties reference the Info Node:

- **InfoNode Name** - the name of the InfoNode to be referenced
- **default** - text to be displayed if the reference cannot be resolved. The default string is "-".

Additional properties are inherited from PageNo Box, Word Box.

**Related concepts**
- [Print a Layout-dependent reference (InfoNodes)](#)
Use InfoNodes to print a value on the report that depends on the paged stream resulting from the report layout.

**Info Node**
The Info Node helps resolve some layout-dependent problems by enabling the use of references.

Info Node (InfoNode type) is a layout container, found in the References group in the Tool Box view.

This object helps resolve some layout-dependent problems by enabling the use of references. This node is invisible and does not consume space in the layout. This is the object referenced by a Reference Box, to print layout specific information, such as a total from a previous page, for example.

**Properties**
The value is stored for querying by a Reference Box. The value is of type String. If the data type of the field being referenced does not correspond to a String, the value must be converted.

**Related concepts**
- Print a Layout-dependent reference (InfoNodes) on page 811
- Use InfoNodes to print a value on the report that depends on the paged stream resulting from the report layout.

**Bar Code Boxes**
The Bar Codes section of the toolbox contains the Bar Code Boxes for the various types of bar codes, such as UPC-A and Code-39.

**Bar Code Box**
The Bar Code Box (BarCodeBox type) is a layout container, found in the Bar Codes group in the Tool Box view. Use this object for bar codes, for example:

![Sample Bar Code](image)

**Figure 410: Sample Bar Code**
The currently supported types are listed in the topic Bar Code type listing on page 911. For licensing reasons, it may be necessary for the user to supply the fonts required to draw the text for some types of bar code.

Bar codes are drawn in nominal sizes. By setting the scaleX and scaleY properties it is possible to draw larger or smaller versions. It is further possible to force a particular width and/or length but specifying the desired extend value.

Specific functions are available to allow you to calculate an expression for the page number such as Page N of M, using the property codeValueExpression.

**Properties**
Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

These properties are specific to Bar Code Box:
- codeType, fidelity, noText, codeValue, check, noDigits, noCheckDigits, thinToThickRelation, thinToGapRelation, controlCharacters, codeValueExpression
Additional properties are inherited from Layout Node, and specific bar codes may have unique properties.

**Related concepts**

Properties related to bar codes on page 902
These are the properties for bar code elements.

**Related reference**

Bar Code type listing on page 911
A table containing all bar codes that are supported by Genero Studio.

**Report Element Properties**

Each element in a report has associated properties. Values for these properties may be literal, or they may be RTL expressions.

If expressions are used, the resulting value must be of the specified data type of the property.

- General Properties on page 876
- Properties related to margins and borders on page 908
- Properties related to charts on page 897
- Properties related to bar codes on page 902
- Properties for Report Metadata on page 910

**Related concepts**

Expressions in properties on page 847
A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

Changing a property value (The Properties view) on page 795
Select a report element in the report page (work area) or Structure View to display and edit the property values in the Properties View.

**General Properties**

These are the general properties of a report element.

- alignment (Alignment) on page 878
- anchorX (Anchor X) on page 878
- anchorY (Anchor Y) on page 878
- baselineType (Baseline Type) on page 878
- bBorder (Bottom Border) on page 879
- bgColor (Background Color) on page 879
- border (Border) on page 879
- class (Class) on page 879
- clip (Clip) on page 880
- color (Color) on page 880
- colspan (Column Span) on page 880
- computeAverage (Compute Average) on page 880
- computeCount (Compute Count) on page 881
- computeDistinctCount (Compute Distinct Count) on page 881
- computeMaximum (Compute Maximum) on page 881
- computeMinimum (Compute Minimum) on page 882
- computeTotal (Compute Totals) on page 882
- enumValues (Enum Values) on page 882
- fidelity (Text Fidelity) on page 883
- fill (Fill) on page 883
- floatingBehavior (Floating Behavior) on page 883
- fontBold (Bold) on page 883
- `fontItalic (Italic)` on page 884
- `fontName (Name)` on page 884
- `fontSize (Size)` on page 884
- `format (Format)` on page 884
- `fWidth (Fix Width)` on page 884
- `hAlign (Horizontal Alignment)` on page 885
- `hidePageHeaderOnLastPage (Hide Page Header On Last Page)` on page 885
- `hidePageFooterOnLastPage (Hide Page Footer On Last Page)` on page 885
- `href (href)` on page 885
- `hPadding (Horizontal Padding)` on page 885
- `hRule (Horizontal Rule)` on page 886
- `id (id)` on page 886
- `indent (Indent)` on page 886
- `intendedResolution (Intended Resolution)` on page 886
- `isNumeric (Numeric Column)` on page 887
- `layoutDirection (Layout Direction)` on page 887
- `localizeText (Localize Text)` on page 888
- `location (Location)` on page 888
- `name (Name)` on page 889
- `padding (Padding)` on page 889
- `pageNumber (Name)` on page 889
- `pageNoOffset (Offset)` on page 889
- `pageNoFormat (Format)` on page 889
- `pageRanges (Page Ranges)` on page 889
- `password (Password)` on page 889
- `pWidth (Proportional Width)` on page 890
- `rBorder (Right Border)` on page 890
- `referenceDefault (Default)` on page 890
- `referenceName (InfoNode Name)` on page 890
- `rule (Rule)` on page 890
- `ruleColor (Rule Color)` on page 890
- `section (Section)` on page 891
- `scaleX (Scale X)` on page 892
- `scaleY (Scale Y)` on page 892
- `splitOversizedItem (Split Oversized Items)` on page 892
- `strikethrough (Strikethrough)` on page 892
- `swapX (Swap X)` on page 892
- `text (Text)` on page 892
- `textAlignment (Text Alignment)` on page 893
- `textExpression (Text Expression)` on page 893
- `tile - replaced by fill (Fill)` on page 883
- `title (Title)` on page 893
- `trimText (Trim Text)` on page 893
- `objectType (Type)` on page 894
- `underline (Underline)` on page 894
- `URL (Location)` on page 894
- `vAlign (Vertical Alignment)` on page 894
- `value (Value)` on page 894
- `visibilityCondition (Visibility Condition)` on page 895
- `vPadding (Vertical Padding)` on page 895
alignment (Alignment)
The alignment property controls alignment of a report element.

In the Properties view, this property is the **Alignment** property in the **Geometry** category.

Controls the x position of this report element in its parent container, unless you have set the x property explicitly.

Type: **Enum**, the alignment choices are:

- **none** - there is no adjustment.
- **near** - shortcut for \( x = 0 \); that is, aligns closest to the origin of x within the parent container
- **far** - shortcut for \( x = \text{max} \), \( \text{anchorX} = 1 \); that is, aligns the most remotely from the origin of x within the parent container
- **center** - shortcut for \( x = \text{max}/2 \), \( \text{anchorX} = 0.5 \), centered in the parent container
- **baseline** - uses baseline alignment

The default value is **none**.

**Related concepts**

**Placing elements on the report page** on page 782

When placing elements on a report page, you determine whether the positioning is absolute or relative by setting element properties that determine how the element acts as the report changes.

anchorX (Anchor X)
The anchorX property shifts the attachment point for self-adjusting nodes.

In the Properties view, this property is the **Anchor X** property in the **Geometry** category.

This property is relevant only if the property x is set. Shifts the attachment point for self-adjusting nodes between the point nearest to the parent's coordinate system's origin (value=0.0) and the most remote point (value=1.0). For nodes that are adjusted by their parent this attribute has no effect. A value of 0.5, for example, sets the attachment point to the center of the node.

Type: **PXML**, point value. The default value is 0.

anchorY (Anchor Y)
The anchorY property shifts the attachment point for self-adjusting nodes.

In the Properties view, this property is the **Anchor Y** property in the **Geometry** category.

This property is relevant only if the property y is set. Shifts the attachment point for self-adjusting nodes between the point nearest to the parent's coordinate system's origin (value=0.0) and the most remote point (value=1.0). For nodes that are placed by their parent this attribute has no effect. A value of 0.5, for example, sets the attachment point to the center of the node.

Type: **PXML**, point value. The default value is 0.

baselineType (Baseline Type)
The baselineType property specifies which baseline of a report element should be linked to which baseline of a preceding element.

In the Properties view, this property is the **Baseline Type** property in the **Layout** category.
Provides additional information for baseline alignment, to specify which baseline of this report element should be linked to which baseline of a preceding element that also has the property alignment set to baseline. This property is relevant only if alignment for the report element is set to baseline.

Type Enum, choices are:

- **leftleft** - the report element and its preceding element will be aligned along their left baselines
- **leftright** - the left baseline of the report element will be aligned with the right baseline of the preceding element
- **rightleft** - the right baseline of the report element will be aligned with the left baseline of the preceding element
- **rightright** - the report element and its preceding element will be aligned along their right baselines

**bBorder (Bottom Border)**
The bBorder property sets the weight of the bottom border of a table object.

In the Properties view, this property is the Bottom Border property in the Table category.

The bBorder property overrides the more general border (Border) on page 879 property for a table object.

Type: PXML, point value.

Default value: None.

**bgColor (Background Color)**
The bgColor property sets the background color for a node.

In the Properties view, this property is the Background Color property in the Color category.

The value is not inherited from the parent node.

Type: Color; valid colors are selected from the Edit Expression color palette. The default value is 'no background color': transparent.

**border (Border)**
The border property sets the weight of the border of a table object.

In the Properties view, this property is the Border property in the Table category.

Type: PXML, point value.

Default value: 1

**class (Class)**
The class property specifies one or more classes for a report element.

In the Properties view, this property is the Class property in the Object category. The Class property is available for all PXML nodes.

The following Class values are used by the Genero Report Writer (see Adding data values and captions on page 797 for further information):

- grwTableHeader
- grwTableRow
- grwHeadlessTableRow

Other Class values with a grw prefix might be used in future versions of Genero Studio.

Otherwise, the value of Class has no effect, but can be used for categorization.

Type String (space separated list of identifiers).

**Related concepts**

Using the PXML expression language on page 854
Genero Report Writer provides the PXML Expression language to define the value of a property that is of the PXML (dimension) type.

clip (Clip)
The clip property specifies whether to clip a report object and its content along the sides.
In the Properties view, this property is the Clip property in the Layout category.
The clip property applies to all layout nodes.

Figure 411: Clipped field

Type Boolean: The default value is false.

color (Color)
The color property sets the paint color for a node and all its children, or for a slice in a map chart.
In the Properties view, this property is the Color property in the Color category.
On Map Charts this property is used to assign each slice of the chart a specific color.
Type: Color, valid colors are selected from the Edit Expressions color palette. The default value is inherited from the parent node. The root node has the color black.

colspan (Column Span)
The colspan property sets when two or more cells are merged in a row definition for a report table.
In the Properties view, this property is the Column Span property in the Table category.
Type: whole number. The number indicates how many cells are included in the merge, starting with zero. If no cells are merged, the colspan would be zero; if two columns are merged, the colspan value is 1, meaning it spans one additional column; if three columns are merged, the colspan value is 2, meaning it spans two additional columns; and so on.
Default value: none.

comment (Comment)
The comment property allows report designers to add additional information as they design the report objects.
In the Properties view, this property is the Comment property in the Object category.
Comments are listed as information lines in the Document Error window, which you can click to select the associated node.
Type String.

computeAverage (Compute Average)
The computeAverage property specifies whether the average (mean) should be computed for a dimension.
In the Properties view, this property is the Compute Average property in the Value category.
Considerations regarding chart drawing and output sorting: When selecting a chart visualization (specified by drawAs) that displays aggregated values, it is necessary that aggregation is performed on the dimensions required by the chart. Similarly, output sorting requires an aggregation function to be defined for all dimensions by which these will be sorted. In the case that more then one aggregation option is selected, the processor will pick the aggregate option that is highest up in the priority list:

1. computeTotal
2. computeAverage
3. computeMaximum
4. computeMinimum
5. computeCount
6. **computeDistinctCount**

Default is FALSE.

Type: **Boolean**

**computeCount (Compute Count)**

The `computeCount` property specifies whether the number of fact rows should be computed for a dimension.

In the Properties view, this property is the *Compute Count* property in the *Value* category.

Considerations regarding chart drawing and output sorting: When selecting a chart visualization (specified by `drawAs`) that displays aggregated values, it is necessary that aggregation is performed on the dimensions required by the chart. Similarly, output sorting requires an aggregation function to be defined for all dimensions by which these will be sorted. In the case that more then one aggregation option is selected, the processor will pick the aggregate option that is highest up in the priority list:

1. `computeTotal`
2. `computeAverage`
3. `computeMaximum`
4. `computeMinimum`
5. `computeCount`
6. `computeDistinctCount`

Default is FALSE.

Type: **Boolean**

**computeDistinctCount (Compute Distinct Count)**

The `computeDistinctCount` property specifies whether or not the number of sub elements should be computed for a dimension.

In the Properties view, this property is the *Compute Distinct Count* property in the *Value* category.

Sub elements are either sub dimensions or, in the case of the innermost dimension, fact rows.

Considerations regarding chart drawing and output sorting: When selecting a chart visualization (specified by `drawAs`) that displays aggregated values, it is necessary that aggregation is performed on the dimensions required by the chart. Similarly, output sorting requires an aggregation function to be defined for all dimensions by which these will be sorted. In the case that more then one aggregation option is selected, the processor will pick the aggregate option that is highest up in the priority list:

1. `computeTotal`
2. `computeAverage`
3. `computeMaximum`
4. `computeMinimum`
5. `computeCount`
6. `computeDistinctCount`

Default is FALSE.

Type: **Boolean**

**computeMaximum (Compute Maximum)**

The `computeMaximum` property specifies whether the maximum value should be computed for a dimension.

In the Properties view, this property is the *Compute Maximum* property in the *Value* category.

Considerations regarding chart drawing and output sorting: When selecting a chart visualization (specified by `drawAs`) that displays aggregated values, it is necessary that aggregation is performed on the dimensions required by the chart. Similarly, output sorting requires an aggregation function to be defined for all dimensions by which these will be sorted. In the case that more then one aggregation option is selected, the processor will pick the aggregate option that is highest up in the priority list:
1. computeTotal
2. computeAverage
3. computeMaximum
4. computeMinimum
5. computeCount
6. computeDistinctCount

Default is FALSE.

Type: Boolean

computeMinimum (Compute Minimum)
The computeMinimum property specifies whether the minimum value should be computed for a dimension.

In the Properties view, this property is the **Compute Minimum** property in the **Value** category.

Considerations regarding chart drawing and output sorting: When selecting a chart visualization (specified by drawAs) that displays aggregated values, it is necessary that aggregation is performed on the dimensions required by the chart. Similarly, output sorting requires an aggregation function to be defined for all dimensions by which these will be sorted. In the case that more than one aggregation option is selected, the processor will pick the aggregate option that is highest up in the priority list:

1. computeTotal
2. computeAverage
3. computeMaximum
4. computeMinimum
5. computeCount
6. computeDistinctCount

Default is FALSE.

Type: Boolean

computeTotal (Compute Totals)
The computeTotal property specifies whether or not totals should be computed for a dimension.

In the Properties view, this property is the **Compute Totals** property in the **Value** category.

Considerations regarding chart drawing and output sorting: When selecting a chart visualization (specified by drawAs) that displays aggregated values, it is necessary that aggregation is performed on the dimensions required by the chart. Similarly, output sorting requires an aggregation function to be defined for all dimensions by which these will be sorted. In the case that more than one aggregation option is selected, the processor will pick the aggregate option that is highest up in the priority list:

1. computeTotal
2. computeAverage
3. computeMaximum
4. computeMinimum
5. computeCount
6. computeDistinctCount

Default: TRUE. Unlike the other aggregation options, totals are computed by default.

Type: Boolean

enumValues (Enum Values)
The enumValues property specifies an optional list of strings that represent ordinal values.

In the Properties view, this property is the **Enum Values** property in the **Value** category.

This attribute is applicable for numeric dimensions only whose value is limited to a range of whole numbers representing a set of symbols.
Consider a dimension containing the values 0 through 11, representing the months of the year (0=Jan, 1=Feb, ..., 11=Dec). For this example, you could set the enumValues = "Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sept,Oct,Nov,Dec". When the dimensions are sorted (inputOrder and displaySelection differ), it can make a visual difference whether the column is declared a numeric enumeration or as a string column containing the literal value and not its ordinal value. The difference comes from the sorting. In the first case (a numeric enumeration), the month will be displayed in the order "Jan, Feb, ..., Dec". In the second case, the alphabetic order of the month names would result in the order "Apr, Aug, Dec, ...".

Type: List of quotable strings

**fidelity (Text Fidelity)**
The fidelity property controls the way text is output.

In the Properties view, this property is the **Text Fidelity** property in the **Font** category.

When set, this property ensures that the preview and printout are 100% the same. In some cases this is necessary when the operating system font definitions deviate from the built-in font definitions in the printer. This flag then instructs the output routine not to use the printer font.

**Note:** If your report uses Asian fonts, it is recommended that you set the fidelity property to **true**.

Type: **Boolean**. The default value is **false**.

**fill (Fill)**
The fill property specifies how an image fills an area.

In the Properties view, this property is the **Fill** property in the **Image** category.

Replaces the previous property **tile**. When a document containing the tile property is opened in version 2.4x or higher, it is automatically replaced with this property.

This property is relevant when both x-Size and y-Size are set so that the outer image bounds are defined.

Type: **Enum**. The value can be one of:

- **completely** - The images is resized to fill the specified area. The aspect ratio is not respected.
- **preserveAspectRatio** - The images is resized to fit the specified area. The aspect ratio is respected. Setting the attributes X-Size_Adjustment and Y-Size_Adjustment to a value of "shrinkToChildren" will shrink the final bounds of the box so that it has the same size as the image.
- **tile** - The image is painted as tiles into the specified area.
- **clip** - The image is painted without scaling, and it is clipped at the edges of the specified area if it is too large to fit.

**floatingBehavior (Floating Behavior)**
The floatingBehavior property controls the sizing behavior of the parent box when the node floats (sets x or y).

This property is not relevant if x or y are not set.

In the Properties view, this property is the **Floating Behavior** property in the **Layout** category.

The valid values are;

- **enclosed** - will make the parent size itself so that this object will be enclosed in the parent. All objects dragged from the toolbox or Data View have this property set to **enclosed**. Note that the node can still float outside of the parent using negative values for x or y. A
- **free** - will make the parent ignore this node during sizing, allowing it to float outside of its bounds.

Type: **Enum**. Valid values are **free** or **enclosed**.

**fontBold (Bold)**
The fontBold property sets a **bold** font style for this node and for all children of this node.

In the Properties view, this property is the **Bold** property in the **Font** category.

The root node font style is **plain**.

Type: **Boolean**. The default value is inherited from the parent.
fontItalic (Italic)
The fontItalic property sets an italic font style for this node and for all children of this node.

In the Properties view, this property is the **Italic** property in the **Font** category.

The root node font style is **plain**.

Type: **Boolean**. The default value is inherited from parent.

fontName (Name)
The fontName property sets the font face name for a node and for all children of the node.

In the Properties view, this property is the **Name** property in the **Font** category.

Type: **Enum**, platform-dependent. The default value is inherited from the parent. The root node has a platform-dependent Sans Serif' font.

**Related concepts**
Configure fonts on page 657

To output well-formatted reports, verify that the necessary fonts are available.

fontSize (Size)
The fontSize property sets the font size in points for a node and for all children of the node.

In the Properties view, this property is the **Size** property in the **Font** category.

During expression evaluation the variable **fontSize** is available, which contains the inherited font size. This variable can be used for relative font sizing. For example, the expression **fontSize*1.2** makes the current font 20% larger than the parent font.

Type: **Numeric**. The default value (not the expression) is inherited from the parent. The root node has a font size of 12 point.

format (Format)
The format property controls the output of a numeric display.

In the Properties view, this property is the **Format** property in the **Value** category.

For DECIMAL data types, format-string consists of pound signs (#) that represent digits and a decimal point. For example, “###.##” produces three places to the left of the decimal point and exactly two to the right.

See the BDL format attribute for additional information and format characters.

Type: **String**.

**Related concepts**
Align and format numbers on page 789

Use proper containers and properties to align and format numbers.

fWidth (Fix Width)
The fWidth property sets the fixed width of a column.

In the Properties view, this property is the **Fix Width** property in the **Table** category.

When you specify a value as Fix Width, you are giving an absolute size for that column. By default, the number entered refers to points, but you can change the unit of measure by specifying the type of units used. See Unit Names on page 978.

Type: **PXML**, point value.

Default value: None.

**Related tasks**
Change the width of a column on page 821
A column width can be proportional or fixed. If you do not specify a width, the columns are equal in width, and the width is calculated based on the width of the table itself.

**hAlign (Horizontal Alignment)**
The hAlign property defines the horizontal alignment of a value in its cell for all cells in a column of a report table. In the Properties view, this property is the **Horizontal Alignment** property in the Table category.

Type: Enum, the alignment choices are:
- **left** - Left-justify in the column.
- **right** - Right-justify in the column.
- **center** - Center in the column
- **baseline** - uses baseline alignment

The default value is **left**.

**hidePageHeaderOnLastPage (Hide Page Header On Last Page)**
The hidePageHeaderOnLastPage property suppresses the drawing of beforeFirst, firstPageHeader, evenPageHeader and oddPageHeader material on the last page.

In the Properties view, this property is the **Hide Page Header On Last Page** property in the Mini Page category.

Child elements querying for available space will, however, be given space values that are reduced by the size of the header as if it were drawn. Type: Boolean. The default value is **false**.

**Related tasks**
- Add headers and footers to a report on page 792
  To add a page header or footer to a report, create a simple container and set the **Section** property.

**hidePageFooterOnLastPage (Hide Page Footer On Last Page)**
The hidePageFooterOnLastPage property suppresses the drawing of afterLast, firstPageTail, evenPageTail and oddPageTail material on the last page.

In the Properties view, this property is the **Hide Page Footer On Last Page** property in the Mini Page category.

Child elements querying for available space will, however, be given space values that are reduced by the size of the footer as if it were drawn. Type: Boolean. The default value is **false**.

**Related tasks**
- Add headers and footers to a report on page 792
  To add a page header or footer to a report, create a simple container and set the **Section** property.

**href (href)**
The href property defines a hyperlink pointing to any resource on the Internet, local disk, or to any anchor inside the report document.

In the Properties view, this property is the **href** property in the Hyperlink category.

See **id** for additional information about creating anchors. The **href** should be defined using the URI syntax.

Type: String.

**Related concepts**
- Use hyperlinks in a report on page 814
  Use the id and href properties to add hyperlink functionality to a report.

**hPadding (Horizontal Padding)**
The hPadding property sets the width of the horizontal padding for a column in a table object.

In the Properties view, this property is the **Horizontal Padding** property in the Table category.

The **hPadding** property overrides the value set for the **padding (Padding)** on page 889 property when setting the vertical padding for a table in a report.
Type: PXML, point value.
Default value: None.

**hRule (Horizontal Rule)**
The hRule property controls the width of the horizontal rule lines for a table. Horizontal rule lines separate rows.

In the Properties view, this property is the **Horizontal Rule** property in the **Table** category.
The property hRule overrides the **rule (Rule)** on page 890 property.

Type: PXML, point value.
Default value: none.

**id (id)**
The id property creates an anchor in the report document.

In the Properties view, this property is the **id** property in the **Hyperlink** category.
Nodes can be identified with a unique id and then used as the target of an href hyperlink.

Type: String.

**Related concepts**
Use hyperlinks in a report on page 814
Use the id and href properties to add hyperlink functionality to a report.

**indent (Indent)**
The indent property specifies the indentation value for the paragraph.

In the Properties view, this property is the **Indent** property in the **Text** category.
The value may be negative.

Type: PXML. The default value is 0.

**intendedResolution (Intended Resolution)**
The intendedResolution property controls the mapping of pixels to device pixels.

In the Properties view, this property is the **Intended Resolution** property in the **Image** category.
The intended resolution is set in Dots Per Inch (DPI).
This property is relevant only if neither X-Size nor Y-Size are set, or if fill is set.
The default is the current screen resolution.

**Usage**
The width of the image is calculated using this formula:

\[
\text{widthOfImageInPixels/intendedResolutionInDotsPerInch}
\]

The height of the image is calculated using this formula:

\[
\text{heightOfImageInPixels/intendedResolutionInDotsPerInch}
\]

Setting the **Intended Resolution** avoids the problems with raster image scaling, such as the degradation in quality when the image is enlarged. On medium resolution devices, it is advisable that each pixel in the image is mapped to a pixel on the output device. For example, to print an image approximately two-inches wide on a 300 DPI printer, the image should be approximately 600 pixels wide and **Intended Resolution** set to 300. For higher resolution devices, an exact correspondence might result in a huge image, so it might be better to set the **Intended Resolution** to a value which is a fraction of the device resolution. For example, for a 1200 DPI printer, set and **Intended Resolution** to 300 DPI, and then each pixel is mapped to 16 device pixels.
**isNumeric (Numeric Column)**
The isNumeric property specifies whether a column is numeric.

Type: **Boolean**.

If TRUE, the column for the pivot table is numeric. If FALSE, the column for the pivot table is a string.

Numeric dimensions may additionally specify enumeration values (See enumValues).

**layoutDirection (Layout Direction)**
The layoutDirection property controls the direction in which child elements are laid out, which is also the direction of the Y-axis.

In the Properties view, this property is the **Layout Direction** property in the **Orientation** category.

Choices are:
- topToBottom
- leftToRight
- bottomToTop
- rightToLeft
- unturned
- turnRight
- upsideDown
- turnLeft
- inherit - the element inherits the orientation of its parent
- swapped - if the swapX property is also set to swapped, the element inherits the swapped orientation of its parent.

These values depend on the language of the system where GRE is running; this allows you to have the layoutDirection follow the custom of the language.

- horizontalNatural - follows the custom of the system language
- horizontalUnnatural - reverses the natural order of the system language
- verticalNatural - follows the custom of the system language
- verticalUnnatural - reverses the natural order of the system language*

For example, to have a horizontal layout that is leftToRight when the language is European, but rightToLeft for Arabic, select horizontalNatural. Select horizontalUnnatural to reverse the natural order of the language in the horizontal layout.

Type: **Enum**. The default value is **topToBottom**. By default the positive X-axis extends 90 degrees right (clockwise) of the positive Y-axis.

**Layout direction and the graphical report designer**
The parent container of the currently focused item is highlighted by a dashed, slowly moving yellow border. The border moves in the layout direction and is open at the far side of the layout direction forming a “U” shape.

![Office Supplies](image)

*Figure 412: A top-to-bottom layout direction*
In this example, the layout direction is top-to-bottom (as illustrated by the arrow). The box is not closed at the bottom.

**lBorder (Left Border)**

The lBorder property sets the weight of the left border of a table object.

In the Properties view, this property is the **Left Border** property in the **Table** category.

The lBorder property overrides the more general **border (Border)** on page 879 property for a table object.

Type: PXML, point value.

Default value: None.

**localizeText (Localize Text)**

The localizeText property indicates whether a localized string exists for this value.

In the Properties view, this property is the **Localize Text** property in the **Text** category.

Checking the box enables the localization of text contents in **Word Boxes and Word Wrap Boxes**.

Type: **Boolean**. Valid choices are True, False. The default value is False.

**Related concepts**

**Using localized strings** on page 716

Use localized strings to change the text at runtime depending on the territory or site where the application is running.

**location (Location)**

The location property specifies the location of a document, such as an image, HTML file, or PDF.

In the Properties view, this property is the **Location** property in the **Image**, **Html**, or **Pdf** category.

Location values are URLs supporting the protocols "http", "file", and "data", for example:

```
```

You can specify the URL using:

- The Browser window. Click the ... button and select the document.
  
  **Note**: The path appears to be an absolute value, but this is true only if the project remains in the same location. If you copy the project elsewhere, this file URI is recalculated.

- The Expression Editor. Use an RTL expression to provide the document name. The file path can be absolute or relative, and it can use variables. For example: `.images/database/"+orderline.product.prodpic.trim()`

Variables (RTL expressions) can be used if the file will change during processing, such as when the image file name is stored in the database and the value can change for each row processed. This is demonstrated in the "OrderReport.4rp" where the "ImageBox2" element has the expression "/images/database/"+orderline.product.prodpic.trim()"

Type: **String**.

**Related concepts**

**Image Box** on page 865

An Image Box displays images in GIF, JPEG, PNG, BMP, WBMP, PDF, and SVG formats.

**HTML Box** on page 862

An HTML Box displays an image of an HTML document in the report.

**PDF Box** on page 864
A PDF Box displays one or more pages from a PDF document in the report.

**name (Name)**
The name property assigns a name to the node for debugging purposes.
In the Properties view, this property is the **Name** property in the **Object** category.
The name must be unique in the document.
Type: **String**. The default value is "".

**padding (Padding)**
The padding property sets the width of all of an object's padding.
In the Properties view, this property is the **Padding** property in the **Table** category.
Can be overridden by specific padding properties.
Type: **PXML**, point value.
Default value: None.

**pageName (Name)**
The pageName property specifies the name of a parent node.
In the Properties view, this property is the **Name** property in the **Page Number** category.
Type: **String**. No default value is set.

**pageNoOffset (Offset)**
The pageNoOffset property specifies the offset added to the current page number.
In the Properties view, this property is the **Offset** property in the **Page Number** category.
When set to 100, for example, the first page will be number 101.
Type: **Numeric**. The default value is **0**.

**pageNoFormat (Format)**
The pageNoFormat property sets the number format type.
In the Properties view, this property is the **Format** property in the **Page Number** category.
Type: **Enum**. Valid values are Arabic, Lowerroman, Upperroman. The default value is **Arabic**.

**pageRanges (Page Ranges)**
The pages ranges property specifies the pages to include.
In the Properties view, this property is the **Page ranges** property in the **PDF** category.
To display all pages within a range, use a hyphen, for example, 3–5.
To display all pages from a particular page onward, use a trailing hyphen, for example, 10–.
To specify number of pages from the end, use a dollar sign. For example, 1–$2 includes all pages except the last two pages.
Separate page numbers or page ranges using commas, for example, 1, 3, 7–10, 12–.
Type: **String**.

**Related concepts**

**PDF Box** on page 864
A PDF Box displays one or more pages from a PDF document in the report.

**password (Password)**
The password property controls access to the document.
In the Properties view, this property is the **Password** property in the **PDF** category.
Type: String.

**Related concepts**
PDF Box on page 864
A PDF Box displays one or more pages from a PDF document in the report.

*pWidth (Proportional Width)*
The `pWidth` property sets the proportional width of a column.

In the Properties view, this property is the *Proportional Width* property in the **Table** category.

When you specify a value in the Proportional Width property, you are specifying its width in proportion to other columns in the same table. For example, consider a table with three columns: columns A, B and C. Column A has a proportional width setting of 1, column B has a proportional width of 2, and column C has a proportional width of 3. This means that column B is two times as wide as column A, and column C is three times as wide as column A.

Type: PXML, point value.
Default value: None.

**Related tasks**
Change the width of a column on page 821
A column width can be proportional or fixed. If you do not specify a width, the columns are equal in width, and the width is calculated based on the width of the table itself.

*rBorder (Right Border)*
The `rBorder` property sets the weight of the right border of a table object.

In the Properties view, this property is the *Right Border* property in the **Table** category.

The `rBorder` property overrides the more general `border (Border)` on page 879 property for a table object.

Type: PXML, point value.
Default value: None.

*referenceDefault (Default)*
The `referenceDefault` property specifies the text value to be displayed when the reference cannot be resolved.

In the Properties view, this property is the *Default* property in the **Reference** category.

Type: String. The default value is "-".

*referenceName (InfoNode Name)*
The `referenceName` property is the name of the Info Node referenced.

In the Properties view, this property is the *InfoNode* property in the **Reference** category.

The name of the InfoNode that is referenced.

Type: String. Mandatory; there is no default value.

*rule (Rule)*
The `rule` property sets the weight of the line between two rows or two columns in a table object.

In the Properties view, this property is the *Rule* property in the **Table** category.

The properties `hRule (Horizontal Rule)` on page 886 and `vRule (Vertical Rule)` on page 895 can override the default value set by the `rule` property for a table object.

Type: PXML, point value.
Default value: 1

*ruleColor (Rule Color)*
The `ruleColor` property controls the color of the rule for a table.

In the Properties view, this property is the *Rule Color* property in the **Table** category.
Type: Color.
Default value: none.

*section (Section)*
The section property controls the layout of content within a parent MiniPage.

In the Properties view, this property is the **Section** property in the **Layout** category.

This attribute of a Layout Node object specifies that the content of the node should print on its parent MiniPage at the location specified by this property.

Type: Enum.

When you select a MiniPage for a page header or footer, for example, you specify where the container should be printed on its parent MiniPage by setting the section property. The report output prints any headers and footers that you have set, based on priorities of each of these values:

- **firstPageHeader**
  - The page header to be printed on the first page only. If this section is defined, subsequent page headers begin printing on the second page.
- **anyPageHeader**
  - The page header for every page, unless separate odd and even page headers are defined.
- **oddPageHeader**
  - Specific page header to be printed on odd pages. Has precedence over anyPageHeader.
- **evenPageHeader**
  - Specific page header to be printed on even pages. Has precedence over anyPageHeader.
- **firstPageFooter**
  - The page footer that prints on the first page only. If this section is defined, subsequent page footers begin printing on the second page.
- **anyPageFooter**
  - The page footer for every page, unless separate odd and even page footers are defined.
- **oddPageFooter**
  - Specific page footer to be printed on odd pages. Has precedence over anyPageFooter.
- **evenPageFooter**
  - Specific page footer to be printed on even pages. Has precedence over anyPageFooter.
- **lastPageFooter**
  - A page footer that prints on the last page only. For the last page, this node takes priority over oddPageFooter, evenPageFooter, or anyPageFooter nodes. A node with this section value set must be located as the last in the sibling list.
- **itemSeparator**
  - Prints between each sibling element, as long as there is more than one element.

If you have an anyPageHeader and a firstPageHeader, for example, the anyPageHeader content prints only on the second and subsequent pages.

**Important:** If you set the section property for any node, the node may not be preceded in its sibling list in the Report Structure tree by any nodes that do not have this property set.

A layout node with a defined section attribute is also known as a **named port**. A layout node without a defined section attribute is also known as a **primary port**.

**Related tasks**
- **Add headers and footers to a report** on page 792
To add a page header or footer to a report, create a simple container and set the Section property.

scaleX (Scale X)
The scaleX property applies the specified scale in this x-direction.

In the Properties view, this property is the Scale X property in the Geometry category.
The scale affects everything contained in this node (children, children-children, etc.). Scales are cumulative.
Type: PXML. The default value is 1.0.

scaleY (Scale Y)
The scaleY property applies a specified scale in the y-direction.

In the Properties view, this property is the Scale Y property in the Geometry category.
The scale affects everything contained in this node (children, children-children, etc.). Scales are cumulative.
Type: PXML. The default value is 1.0.

splitOversizedItem (Split Oversized Items)
The splitOversizedItem property defines the behavior for when a single item exceeds the space in layout direction.

In the Properties view, this property is the Split Oversized Items property in the Mini Page category.
The Split Oversized Items property has no effect if the Y property is set (that is, the element is self-placing).

When set to TRUE, this value allows the splitting of large non-propagating items (such as HTMLBOX, WORDWRAPBOX or IMAGEBOX) into chunks using preferable breakpoints (if available). Preferable breakpoints refer to whitespace or between table rows.
Otherwise the box becomes overfull.

Note: The splitting of a large item is a costly operation. The item that is split is kept in memory until the last split has been performed. The item that is split should not exceed a few pages. If possible, consider using a PDF Box element, which arranges the content across pages without the need for Split Oversized Items.
Type: Boolean.

strikethrough (Strikethrough)
The strikethrough property specifies strikethrough for the text.

In the Properties view, this property is the Strikethrough property in the Font category.
Type: Boolean. The default value is false.

swapX (Swap X)
The swapX property reverses the direction of the X-axis.

In the Properties view, this property is the Swap X property in the Orientation category.
By default the positive X-axis extends 90 degrees right (clockwise) of the positive Y-axis; if, for example, the Y-Axis points to north, the X axis will point eastward. Setting this value reverses the direction so that (in the example) the X-Axis would point to the west which is 90 degrees to the left (counter clockwise) of the Y-Axis.
Type: Boolean. The default value is false.

text (Text)
The text property specifies the text to be drawn.

In the Properties view, this property is the Text property in the Text category.
Occurrences of the newline character "\n" within the string cause line breaks.
Type: String. The default value is "".

For Word Boxes, Word Wrap Boxes, and Decimal Format Boxes, the value of this property may be edited directly in the report design document instead. Double-click the object and the input cursor will be placed in the text. The layout of the document is updated on each keystroke.
For Page Number Boxes, Bar Code Boxes, and Reference Boxes, the value of the text property specifies the maximum width.

**textAlignment (Text Alignment)**
The textAlignment property controls the horizontal alignment of text.

In the Properties view, this property is the Text Alignment property in the Text category.

Type: Enum. Values are left, center, right, justified. The default value is left.

**textExpression (Text Expression)**
The textExpression property is a PXML string expression value to calculate a page number string.

In the Properties view, this property is the Text Expression property in the Text category.

This is a property of the PAGENOBOX, which expects a PXML string expression value to calculate a page number string. When set, the value of the property overrides all other formatting relevant properties of the PAGENOBOX, although the text property is still used to set the length.

Type: PXML.

See functions and examples in Using a page number string on page 800.

If a value for this property, or for the X-Size or text properties, is not set, a default length is used in calculating the page number string.

**title (Title)**
The title property specifies the title of a report, output, or column.

Type: String.

**Report titles**
For a report, specifies the metadata for the title of the report. In the case of SVG, the title property is used as a document caption in Genero Report Viewer.

In the Properties view, this property is the Title property in the Metadata category.

See also Properties for Report Metadata on page 910.

**Pivot table**
For a pivot table, specifies the title of the output. If and where this text is rendered depends on the selected visualization type (specified by drawAs).

**Pivot table dimension and measure titles**
For a pivot table hierarchy and measure, specifies the title of the column. If and where this text is rendered depends on the selected visualization type (specified by drawAs).

**tBorder (Top Border)**
The tBorder property sets the weight of the top border of a table object.

In the Properties view, this property is the Top Border property in the Table category.

The tBorder property overrides the more general border (Border) on page 879 property for a table object.

Type: PXML, point value.

Default value: None.

**trimText (Trim Text)**
The trimText property controls the trimming of spaces.

In the Properties view, this property is the Trim Text property in the Text category.

Controls the trimming of spaces of the value of the text attribute.
Type: **Enum**. Values are both, compress, left, right. The default value is not set.

*Transform transparently (transformTransparently)*

When the **Transform transparently** property is set on a parent, its children map their orientation based on the parent's parent orientation rather than the parent.

Absolute orientation specifications (such as setting **Layout Direction** as "leftToRight") are internally mapped to relative graphics context operations like "turn-left-by-90-degrees", depending on the orientation of the parent.

When **Transform transparently** is set on a parent, its children map their commands based on the parent's parent orientation, rather than on the parent.

This is useful for turning the entire document or part of it.

In the Properties view, this property is the **Transform transparently** property in the **Orientation** category.

Type: **Boolean**.

**Related concepts**

*layoutDirection (Layout Direction)* on page 887

The layoutDirection property controls the direction in which child elements are laid out, which is also the direction of the Y-axis.

*swapX (Swap X)* on page 892

The swapX property reverses the direction of the X-axis.

*objectType (Type)*

The objectType property specifies the type of the report element.

In the Properties view, this property is the **Type** property in the **Object** category.

This property is automatically set when you drop a specific element on the page.

*underline (Underline)*

The underline property specifies that the text is underlined.

In the Properties view, this property is the **Underline** property in the **Font** category.

Type: **Boolean**. The default value is **false**.

*URL (Location)*

The URL property specifies the loading location or the name of the image to display.

In the Properties view, this property is the **Location** property in the **Image** category.

Type: **String**. A value is mandatory; there is no default value.

*vAlign (Vertical Alignment)*

The vAlign property defines the vertical alignment of a value in its cell for all cells in a column of a report table.

In the Properties view, this property is the **Vertical Alignment** property in the **Table** category.

Type: **Enum**, the alignment choices are:

- **left** - Left-justify in the column.
- **right** - Right-justify in the column.
- **center** - Center in the column
- **baseline** - uses baseline alignment

The default value is **top**.

*value (Value)*

The value property specifies the value of the item.

In the Properties view, this property is the **Value** property in the **Miscellaneous** or the **Items** category.

Type: **Numeric**.
**value (Value) – pivot table**
The value property specifies the value of the pivot table hierarchy or value.

In the Properties view, this property is the **Value** property in the **Value** category.

Type: Can be a String or Float depending on the declared **data type**. In case of a numeric column, this value is converted to a float value. The entered value will fail if:

- The value is not set.
- The column is declared as numeric and the value cannot be parsed as a float point value.

**Related concepts**

*Working with pivot tables* on page 836
The pivot table element defines a table element with fixed roles and types for its columns.

**visibilityCondition (Visibility Condition)**
The visibilityCondition property indicates whether the object is visible or hidden.

In the Properties view, this property is the **Visibility Condition** property in the **Object** category.

Type: **Boolean**. The default is TRUE.

**Related tasks**

*Show or hide a report element* on page 814
Make report elements conditionally invisible by using the **Visibility Condition** property.

**vPadding (Vertical Padding)**
The vPadding property sets the width of the vertical padding for a column in a table object.

In the Properties view, this property is the **Vertical Padding** property in the **Table** category.

The vPadding property overrides the value set for the **padding (Padding)** on page 889 property when setting the vertical padding for a table in a report.

Type: **PXML**, point value.

Default value: None.

**vRule (Vertical Rule)**
The vRule property controls the width of the vertical rule lines for a table. Vertical rule lines separate columns.

In the Properties view, this property is the **Vertical Rule** property in the **Table** category.

The property vRule overrides the **rule (Rule)** on page 890 property.

Type: **PXML**, point value.

Default value: none.

**x (X)**
The x property specifies the x-value of a X/Y coordinate pair defined by the element.

In the Properties view, this property is the **X** property in the **Geometry** category.

The x value is an offset in the X-Size direction of the parent.

Type: **Numeric**. The default value is calculated during placing by the parent.

**X-Size (X-Size)**
The X-Size property gives the box a fixed dimension.

In the Properties view, this property is the **X-Size** property in the **Geometry** category.

**Note:** To preserve the aspect ratio of an image, set the value of either **Y-Size** or X-Size only, and allow Report Writer to calculate the corresponding value. If you set both properties, the resulting image appears distorted.

Type: **Numeric**. The default value is calculated after the node has completed child alignment. The value is set to the smallest possible value that encloses all children without clipping any of them.
Tip: If you want a box to have the same width as its parent, set **X-Size** to \textit{max}. See \textit{Modify the sizing policy of containers} on page 785.

Tip: The **X-Size** property can include the \texttt{oddPhysicalPage()} or \texttt{evenPhysicalPage()} functions. See \textit{Start on an odd or even page} on page 805.

**X-Size Adjustment (X-Size Adjustment)**
The **X-Size Adjustment** property specifies how the adjustment to the **X-Size** is to be made.

In the Properties view, this property is the **X-Size Adjustment** property in the \textit{Geometry} category.

A value of \texttt{shrinkToChildren} shrinks the **X-Size** as much as possible without clipping any of the children. A value of \texttt{expandToParent} causes the box to stretch as much as possible without intersecting the borders of a parent or sibling.

For objects that draw output with a defined size (Word Boxes, BarCode Boxes, Image Boxes where the tile property is set to FALSE), the value \texttt{shrinkToChildren} does not shrink the object below this size even if all children are smaller or there are no children. Self-placing children are not considered.

Type: \textit{Enum}. Choices are \texttt{shrinkToChildren}, \texttt{expandToParent}.

\textbf{y (Y)}
The \textit{y} property specifies the \( y \)-value of a \( X/Y \) coordinate pair defined by the element.

In the Properties view, this property is the \textit{Y} property in the \textit{Geometry} category.

Changing the value adjusts the node at the specified coordinate. The coordinate value is an offset in the **Y-Size** direction of the parent.

Type: \textit{Numeric}. The default value is calculated during placing by the parent.

**Y-Size (Y-Size)**
The **Y-Size** property gives the box a fixed dimension.

In the Properties view, this property is the **Y-Size** property in the \textit{Geometry} category.

\textbf{Note:} To preserve the aspect ratio of an image, set the value of either Y-Size or X-Size, allowing Report Writer to calculate the corresponding value. If you set both properties, the resulting image appears distorted.

Do not set a value for this property in \textit{WordWrapBoxes}, because the element should typically grow based on its content.

\textbf{Note:} If the parent object is a propagating container and the child object does not fit in the remaining space for the parent object, the \texttt{rest} variable for the **Y-Size** property yields the same value as \texttt{max} (the child expands to the maximum extent of the parent). This forces the parent object to propagate and avoids overfullness.

Type: \textit{Numeric}. The default value is calculated after the node has completed its child alignment. The value is set to the smallest possible value that encloses all children without clipping any of them.

\textbf{Tip:} If you want a box to have the same height as its parent, set **Y-Size** to \textit{max}. See \textit{Modify the sizing policy of containers} on page 785.

\textbf{Tip:} To ensure that part of the report starts on an odd page, set **Y-Size** to \texttt{oddPhysicalPage()}?0: \textit{max}. See \textit{Start on an odd or even page} on page 805.

**Y-Size Adjustment (Y-Size Adjustment)**
The **Y-Size Adjustment** property specifies how the adjustment to the **Y-Size** is to be made.

In the Properties view, this property is the **Y-Size Adjustment** property in the \textit{Geometry} category.

A value of \texttt{shrinkToChildren} shrinks the length of **Y-Size** as much as possible without clipping any of the children. A value of \texttt{expandToParent} causes it to stretch as much as possible without intersecting the borders of a parent or sibling.

For objects that draw output with a defined size (Word Boxes, BarCode Boxes, Image Boxes where the tile property is set to FALSE), the value \texttt{shrinkToChildren} does not shrink the object below this size even if all children are smaller or there are no children. Self-placing children are not considered.

Type: \textit{Enum}. Choices are \texttt{shrinkToChildren}, \texttt{expandToParent}.
Properties related to charts
These are the properties for charts and chart items, including map charts, category charts, XY charts, and pivot tables.

- `categoryKey` (Category Key) on page 897
- `categoryTitle` (Categories Title) on page 897
- `color` (Color) on page 897
- `computeAggregatesInnermostDimension` (Compute aggregates on the innermost dimension) on page 898
- `displayFactRows` (Display Fact Rows) on page 898
- `displayRecurringDimensions` (Display Recurring Dimension Values) on page 898
- `displaySelection` (Display Selection) on page 898
- `drawAs` (Draw As) on page 899
- `drawLabels` (Draw Labels) on page 900
- `drawLegend` (Draw Legend) on page 900
- `hierarchiesInputOrder` (Hierarchies input order) on page 900
- `key` (Key) on page 900
- `keysTitle` (Keys Title) on page 900
- `outputOrder` (Output Order) on page 900
- `seriesTitle` (Series Title) on page 901
- `sortAscending` (Sort Ascending) on page 901
- `sortBy` (Sort By) on page 902
- `topN` (Top N) on page 902
- `valuesTitle` (Values Title) on page 902
- `xAxisTitle` (xAxisTitle) on page 902
- `yAxisTitle` (yAxisTitle) on page 902

Related concepts
- Working with business graphs on page 821
  A business graph allows you to represent data visually on the report.

Business Graphs on page 869
The Business Graphs section of the toolbox contains a variety of chart objects (map charts, category charts, XY charts) and pivot table objects.

`categoryKey` (Category Key)
The `categoryKey` property specifies the key of a category in a category chart.

In the Properties view, this property is the **Category Key** property in the **Items** category.

Specifies the key of a category in a **Category Chart**. Must be unique within a chart.

Type: **String**. The default is a blank String.

`categoryTitle` (Categories Title)
The `categoryTitle` property specifies the title for the categories axis in a category chart.

In the Properties view, this property is the **Categories Title** property in the **Chart** category.

Specifies the title for the categories axis in a **Category Chart**.

Type: **String**. The default is a blank String.

`color` (Color) -- Map Charts
The `color` property sets the paint color for the slices in a map chart.

On the **Map Chart Item**, this property is used to assign each slice of the chart a specific color. If Color is not specified, the chart uses the default Genero Studio colors.

Type: **Color**, valid colors are selected from the Edit Expressions color palette.
**computeAggregatesInnermostDimension (Compute aggregates on the innermost dimension)**
The `computeAggregatesInnermostDimension` property specifies whether to compute aggregates on the innermost dimension.

In the Properties view, this property is the **Compute aggregates on the innermost dimension** property in the **Chart** category.

Default is TRUE.

Type: **Boolean**

**displayFactRows (Display Fact Rows)**
The `displayFactRows` property specifies whether or not fact rows (the individual unaggregated data items) display.

In the Properties view, this property is the **Display Fact Rows** property in the **Chart** category.

This is applicable only if the selected output visualization (specified by `drawAs`) is capable of drawing individual rows. This is currently the case for the "Table" visualization type only.

Type: **Boolean**

**displayRecurringDimensions (Display Recurring Dimension Values)**
The `displayRecurringDimensions` property specifies whether recurring dimension values in the same column of table output should be displayed.

In the Properties view, this property is the **Display Recurring Dimension Values** property in the **Chart** category.

By default, cells with recurring values are left empty.

Type: **Boolean**

**displaySelection (Display Selection)**
The `displaySelection` property selects which of the declared dimensions or measures to display.

In the Properties view, this property is the **Display Selection** property in the **Chart** category.

For example, given a table with 4 dimensions, specifying a value of "3,2,0" selects the last, the second last and the first column for display.

Depending on the visualization type (set by the `drawAs` property), it is possible that not all selected dimensions will display.

Not specifying a value is equivalent to selecting all declared dimensions. For example, given a table with three dimensions, not specifying this attribute is the equivalent of specifying a value of "0,1,2".

For dimensions, specifying an empty set will display the measures only and the grand total line.

For measures, specifying an empty set will display the dimensions, their aggregates and the grand total line.

Type: **Column selector**

**domainLowerBound (Domain Lower Bound)**
The `domainLowerBound` property specifies the lowest value on the X-axis in an XY chart.

The property is not relevant to polar charts.

In the Properties view, this property is the **Domain Lower Bound** property in the **Chart** category.

Type: **Numeric**

**Related concepts**
- `domainUpperBound (Domain Upper Bound)` on page 899
  The `domainUpperBound` property specifies the highest value on the X-axis in an XY chart.
- `rangeUpperBound (Range Upper Bound)` on page 901
  The `rangeUpperBound` property specifies the highest value on the Y-axis in a business graph or a pivot table visualized as a graph.
- `rangeLowerBound (Range Lower Bound)` on page 901
The rangeLowerBound property specifies the lowest value on the Y-axis in a business graph or a pivot table visualized as a graph.

**XY charts on page 823**
The XY chart allows you to create a graph that maps a series that has two values, as an XY-plot.

*domainUpperBound (Domain Upper Bound)*
The domainUpperBound property specifies the highest value on the X-axis in an XY chart.

The property is not relevant to polar charts.

In the Properties view, this property is the Domain Upper Bound property in the Chart category.

Type: Numeric.

**Related concepts**

*domainLowerBound (Domain Lower Bound) on page 898*
The domainLowerBound property specifies the lowest value on the X-axis in an XY chart.

*rangeUpperBound (Range Upper Bound) on page 901*
The rangeUpperBound property specifies the highest value on the Y-axis in a business graph or a pivot table visualized as a graph.

*rangeLowerBound (Range Lower Bound) on page 901*
The rangeLowerBound property specifies the lowest value on the Y-axis in a business graph or a pivot table visualized as a graph.

**XY charts on page 823**
The XY chart allows you to create a graph that maps a series that has two values, as an XY-plot.

*drawAs (Draw As)*
The drawAs property specifies the type of chart rendered from the data.

In the Properties view, this property is the Draw As property in the Chart category.

This property also allows you to specify that the chart displays as a table.

Type: Enum.

Valid values for drawAs for Map Chart: Bar|Bar3D|Pie|Pie3D|Ring|Table|SortedTable|AggregatedTable. The default is a Pie.

Valid values for drawAs for Category Chart: Area|Bar|Bar3D|Line|Line3D|SpiderWeb|StackedBar|StackedArea|Waterfall|Table|SortedTable|AggregatedTable. The default is a Bar.

**Note:** If you select Waterfall as the chart type, the value in the last category of the data set should be (redundantly) specified as the sum of the items in the preceding categories - otherwise, the final bar in the chart will be incorrectly plotted. At the present time, the chart can only have one category.

Valid values for drawAs for XY Chart: Area|Line|Polar|Scatter|StackedArea|Step|StepArea|TimeSeries|Table|SortedTable. The default is Line.

Valid values for drawAs for Pivot Table: Area | Bar | Bar3D | Line | Line3D | Pie | Pie3D | Polar | Ring | Scatter | SpiderWeb | StackedArea | StackedBar | Step | StepArea | Table | TimeSeries | Waterfall | XYArea | XYStackedArea | XYLine. The default is a Table.

**Related concepts**

*Working with business graphs on page 821*
A business graph allows you to represent data visually on the report.

*Output charts as tables on page 835*
All chart types now support the drawing of the data as tables via the Draw As property.

*Business Graphs on page 869*
The Business Graphs section of the toolbox contains a variety of chart objects (map charts, category charts, XY charts) and pivot table objects.

**drawLabels (Draw Labels)**
The drawLabels property controls whether labels are drawn for a map chart.

In the Properties view, this property is the Draw Labels property in the Chart category.

Controls whether the Labels are drawn for a MapChart.

Type: Boolean. The default value is true.

**drawLegend (Draw Legend)**
The drawLegend property controls whether a legend is drawn for a map chart.

In the Properties view, this property is the Draw Legend property in the Chart category.

Controls whether the Legend is drawn for a MapChart. The option to remove the legend is useful when many charts are drawn next to each other in a document. You can make the charts share a single legend by specifying the legend only on one of the charts.

Type: Boolean. The default value is true.

**hierarchiesInputOrder (Hierarchies input order)**
The hierarchiesInputOrder property specifies the order by which the data is presorted.

In the Properties view, this property is the Hierarchies input order property in the Chart category.

If nothing is specified, the data is assumed to be presorted in the declaration order of the dimensions. This is the default. For example, given a table with three dimensions, not specifying this attribute is the equivalent of specifying a value of "0,1,2".

Specifying an empty set indicates that the data will arrive unsorted. Large data amounts should be at least partially presorted.

Specifying a wrong input order can cause runtime errors and yield incorrect results.

Type: Order specifier

**key (Key)**
The key property specifies the key of the item in a chart.

In the Properties view, this property is the Key property in the Items category.

Within a category chart, specifies the key of an item within a category; must be unique.

Type: String.

**keysTitle (Keys Title)**
The keysTitle property specifies the title of the keys Axis (usually the y Axis) of a Business Chart.

In the Properties view, this property is the Keys Title property in the Chart category.

Type: String. The default is a blank String.

**outputOrder (Output Order)**
The outputOrder property specifies the order by which the data should be presented.

In the Properties view, this property is the Output Order property in the Chart category.

Not specifying a value or specifying the empty set will output the data in the order it was received.

Consider a pivot table with the dimensions "country" and "region" and the measure "turnover". Specifying an output order of "-1" creates an output in which the country with the highest turnover is listed first. Within that country the region with the highest turnover is listed first and within each country the individual fact rows are ordered in descending order by the turnover value.

**Note:** Specifying an output order may cause large latency and memory consumption on large input data.
When an output order is specified, it is possible to specify a cutoff value called **topN** that limits the number of items displayed. Continuing with our example, a cutoff value of 2 would limit the output to the two top countries; within each country the two top regions; and within each region to the two highest fact rows.

**Note:** Specifying an output order currently limits the number of displayable aggregations to one per dimension. If more are specified, one is picked following a priority list. See **computeTotal**.

Type: **Order specifier**

**rangeLowerBound (Range Lower Bound)**
The `rangeLowerBound` property specifies the lowest value on the Y-axis in a business graph or a pivot table visualized as a graph.

The property is only applicable to visualizations with a Y-Axis, which depends on the value of the **drawAs** property.

In the Properties view, this property is the **Range Lower Bound** property in the **Chart** category.

Type: **Numeric**.

**Related concepts**

**rangeUpperBound (Range Upper Bound)** on page 901
The `rangeUpperBound` property specifies the highest value on the Y-axis in a business graph or a pivot table visualized as a graph.

**Working with business graphs** on page 821
A business graph allows you to represent data visually on the report.

**Working with pivot tables** on page 836
The pivot table element defines a table element with fixed roles and types for its columns.

**rangeUpperBound (Range Upper Bound)**
The `rangeUpperBound` property specifies the highest value on the Y-axis in a business graph or a pivot table visualized as a graph.

The property is only applicable to visualizations with a Y-Axis, which depends on the value of the **drawAs** property.

In the Properties view, this property is the **Range Upper Bound** property in the **Chart** category.

Type: **Numeric**.

**Related concepts**

**rangeLowerBound (Range Lower Bound)** on page 901
The `rangeLowerBound` property specifies the lowest value on the Y-axis in a business graph or a pivot table visualized as a graph.

**Working with business graphs** on page 821
A business graph allows you to represent data visually on the report.

**Working with pivot tables** on page 836
The pivot table element defines a table element with fixed roles and types for its columns.

**seriesTitle (Series Title)**
The `seriesTitle` property specifies the title of the series in an XY Chart.

In the Properties view, this property is the **Series Title** property in the **Items** category.

Type: **String**.

**sortAscending (Sort Ascending)**
The `sortAscending` property sorts the values in ascending order.

In the Properties view, this property is the **Sort Ascending** property in the **Chart** category.

Set this value to false to reverse the display order specified by the **sortBy** property.

Type: **Boolean**.
**Related concepts**

**sortBy (Sort By) on page 902**

The sortBy property specifies the order in which the items of a chart are displayed.

**sortBy (Sort By)**

The sortBy property specifies the order in which the items of a chart are displayed.

In the Properties view, this property is the **Sort By** property in the **Chart** category.

Valid values are:

- **Key** - Sort alphabetically by the key value (MapChart) or by the category/key value (CategoryChart).
- **Value** - Sort by the numeric value.
- **InputOrder** - Preserve the order in which the items were defined. If more than one value is received for a particular key value (MapChart) or category/key value combination (CategoryChart), the first value received defines the order.

By setting the property sortAscending to false a reverse sorting for each of these options can be obtained.

Type: Enumeration.

**topN (Top N)**

The topN property specifies the number of records to display.

In the Properties view, this property is the **Top N** property in the **Chart** category.

Only valid when outputOrder is specified. The specified value limits the number of distinct dimension values displayed for each dimension and the number of fact rows displayed for the innermost dimension to the specified number.

Type: Integer

**valuesTitle (Values Title)**

The valuesTitle property specifies the title of the values axis of a business chart.

In the Properties view, this property is the **Values Title** property in the **Chart** category.

Type: String.

**xAxisTitle (xAxisTitle)**

The xAxisTitle property specifies the title of the x Axis of a Business Chart.

In the Properties view of an XY chart, this property is the **xAxisTitle** property in the **Chart** category.

Type: String.

**yAxisTitle (yAxisTitle)**

The yAxisTitle property specifies the title of the y Axis of a Business Chart.

In the Properties view of an XY chart, this property is the **yAxisTitle** property in the **Chart** category.

Type: String.

**Properties related to bar codes**

These are the properties for bar code elements.

- **check (Check)** on page 903
- **codeType (Code Type)** on page 903
- **codeValue (Code Value)** on page 903
- **codeValueExpression (Code Value Expression)** on page 903
- **controlCharacters (Control Characters)** on page 904
- **dataSymbolsPerLine (Data Symbols per Line)** on page 904
- **encoding (Encoding)** on page 904
- **errorCorrectionDegree (Error Correction Degree)** on page 905
- **noCheckDigits (Number Check Digits)** on page 906
- **noDigits (Number Digits)** on page 906
- **noText (Hide Text)** on page 906
- **preferRectangularSymbols (Prefer Rectangular Symbols)** on page 906
- **rawCodeValue (Raw Code Value)** on page 906
- **smartParse (Smart Parse)** on page 907
- **thinToGapRelation (Thin To Gap Relation)** on page 907
- **thinToThickRelation (Thin To Thick Relation)** on page 907

**Related concepts**

**Bar Codes** on page 911
The report element container for a Bar Code is a Bar Code Box. This flow object draws bar codes.

**Bar Code Boxes** on page 875
The Bar Codes section of the toolbox contains the Bar Code Boxes for the various types of bar codes, such as UPC-A and Code-39.

**check (Check)**
The check property checks the checksum character.

In the Properties view, this property is the **Check** property in the **Bar Code** category.

When set, the checksum character of the specified code value is checked for correctness.

Type: **Boolean**. The default value is **true**.

**codeType (Code Type)**
The codeType property specifies the type of bar code.

In the Properties view, this property is the **Code Type** property in the **Bar Code** category.

This is mandatory, a default value is not set. Not all codeTypes are relevant to all Bar Codes.

Type: **Enum**.

**Related concepts**

**Bar Codes** on page 911
The report element container for a Bar Code is a Bar Code Box. This flow object draws bar codes.

**codeValue (Code Value)**
The codeValue property specifies the code value of a bar code.

In the Properties view, this property is the **Code Value** property in the **Bar Code** category.

The character set and semantic constraints depend on the code type selected.

Type: **String**.

**Related concepts**

**Bar Codes** on page 911
The report element container for a Bar Code is a Bar Code Box. This flow object draws bar codes.

**codeValueExpression (Code Value Expression)**
The codeValueExpression property is a PXML string expression value to calculate a page number string, depending on a bar code.

Some enveloping machines use bar codes to collate report pages and place them into envelopes. The code value expression typically encodes the page number and the total number of pages.

In the Properties view, this property is the **Code Value Expression** property in the **Bar Code** category.

This is a property of the **Bar Code Box**, which expects a PXML string expression value to calculate a page number string. When set, the value of the property overrides the **codeValue** property. The value of the **codeValue** property is still used for measuring the required space, but if it is not set, the default size is computed from the expression; see **calculating the page number string**.
Type: PXML.

The code expression works similarly to the text expression. See functions and examples in Using a page number string on page 800.

**controlCharacters (Control Characters)**

The controlCharacters configures which characters to use for textual printout of control characters in Code 93 and Code 93 extended.

In the Properties view, this property is the Control Characters property in the Bar Code category.

Code 93 defines four control characters "!?\|" which are represented per default by the unicode characters "circled dash" '#' (229d), "circled asterisk operator" '#' (229b), "circled division slash" '#' (2298) and "circled plus" '#' (2295). Depending on the font used, it might be desirable to used different characters than the default characters.

Type: String. Default value: "⊝⊛⊘⊕"

**dataSymbolsPerLine (Data Symbols per Line)**

The dataSymbolsPerLine property specifies the number of data symbols per line.

In the Properties view, this property is the Data Symbols per Line property in the Bar Code category.

This property is unique to the pdf-417 on page 939 bar code type.

Type: Integer value.

The value must be an integer between 1 and 30. Low values cause more narrow printout with more lines. The number of lines is not allowed to exceed 90. It should be noted, that the overall required image space usually grows with lower values because there is a constant amount of organizational information which is added with each additional line. This is not generally the case, since lines have to be filled with padding so that specially with small amounts of data a larger value may actually create a larger image. If the value is not specified, the system computes a value that minimizes image space.

Fails if: Value cannot be parsed as a integer value. Value is not in the range 1...30.

Default value: A value that minimizes the overall image size.

**encoding (Encoding)**

The encoding property sets the encoding for non-ASCII characters in the code value.

In the Properties view, this property is the Encoding property in the Bar Code category.

This property is used by the pdf-417 and qr-code bar code types.

Type: Encoding

**PDF-417**

Run "java CharsetInfo" for a list of character set encodings available on a particular platform. Valid example values are 'ISO-8859-15' or 'IBM437'.

Fails if:
- Value is not a valid host name
- Socket connection cannot be established

Default value: not set (the lower 8 bits of the unicode values are encoded)

**QR-Code**

In the QR-Code bar code, the following encodings can be set:

<table>
<thead>
<tr>
<th>Encoding Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO-8859-1</td>
<td>Use this setting if all characters in codeValue are from this code page. Some scanners or scanner apps interpret</td>
</tr>
<tr>
<td>Encoding Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Bytes</td>
<td>Use this setting to set the byte values. Literal characters in codeValue are mapped to ISO-8859-1 byte representation and characters not representable in XML documents can be escaped by a backslash ('') character followed by a 3 digit octal literal. The backslash character itself can be escaped by a sequence of two backslash characters or its octal representation \134 in ISO-8859-1.</td>
</tr>
<tr>
<td>UTF-8</td>
<td>This is the default value and should be used unless the two other options from above are applicable. If a scanner fails to interpret the characters correctly and all character in codeValue are available from a different encoding listed by CharsetInfo, then this encoding should be tried next.</td>
</tr>
<tr>
<td>Any encoding listed by CharSetInfo (e.g. &quot;Shift_JIS&quot;, &quot;Big5&quot; or &quot;ISO-8859-8&quot;)</td>
<td>If all of the option above are not applicable this option should be used. For encodings for which a ECI code exists, the code will be embedded allowing the scanner to change the interpretation accordingly.</td>
</tr>
<tr>
<td>Any encoding listed by CharSetInfo prefixes with the string &quot;RAW-&quot; (e.g. &quot;RAW-Shift_JIS&quot;, &quot;RAW-Big5&quot; or &quot;RAW-UTF-8&quot;)</td>
<td>Some scanners do not recognize ECI codes and expect a specific encoding (other than ISO-8859-1). For this case, this setting should be used.</td>
</tr>
</tbody>
</table>

Fails if: Value is not a valid encoding name.

Default value: UTF-8

`errorCorrectionDegree (Error Correction Degree)`
The `errorCorrectionDegree` property specifies the error correction degree.

In the Properties view, this property is the **Error Correction Degree** property in the **Bar Code** category.

This property is used by the `pdf-417` and `qr-code` bar code types.

Type: Integer value.

**PDF-417**

Valid values are in the range 0...8. Higher values make the image more robust.

Fails if: Value cannot be parsed as an integer value. Value is not in the range 0...8.

Default value: A value that proportional to the data size.

**QR Code**

Valid values are in the range of 0 - 3. Higher values make the image more robust (ISO 18004:2006, 6.5.1 defines: 0=7%, 1=15%, 2=25% and 3=30%).

Fails if: Value cannot be parsed as an integer value. Value is not in the range of 0 - 3.

Default value: 3
**noCheckDigits (Number Check Digits)**
The noCheckDigits controls the expected number of check digits for codeValue for codes of type "code-11-matrix" and "code-93".
In the Properties view, this property is the Number Check Digits property in the Bar Code category.
Type: Numeric.

**noDigits (Number Digits)**
The noDigits property controls the expected number of digits for codeValue for a code type that allows a variable number of digits.
In the Properties view, this property is the Number Digits property in the Bar Code category.
Specifically these are the code types:
- code-2-5-industrial
- code-2-5-inverted
- code-2-5-IATA
- code-2-5-interleaved
- code-2-5-matrix
- code-2-5-datalogic
- code-BCD-matrix
- code-11-matrix
- code-39
- code-39-extended
- codabar 18
- codabar 2
Type: Numeric.

**noText (Hide Text)**
The noText property specifies whether to suppress text output.
In the Properties view, this property is the Hide Text property in the Bar Code category.
Type: Boolean. The default value is false.

**preferRectangularSymbols (Prefer Rectangular Symbols)**
The preferRectangularSymbols property enables rectangular symbols.
In the Properties view, this property is the Prefer Rectangular Symbols property in the Bar Code category.
The data-matrix bar code is usually quadratic, and any code value can be represented by a quadratic symbol. If you are concerned about running out of space in the vertical of the page, you might prefer a symbol that is wider than it is high. This property produces a rectangular shaped symbol if the encoded data does not exceed 49 code words. Check the box to enable rectangular symbols.
Type: Boolean. Valid choices are True, False. The default value is False.
Fails if the value cannot be parsed as a Boolean value.

**rawCodeValue (Raw Code Value)**
The rawCodeValue property specifies the code value at a lower level than codeValue.
In the Properties view, this property is the Raw Code Value property in the Bar Code category.
This property is unique to the pdf-417 on page 939 bar code type.
Type: A comma-separated list of integers in the range 0...899.
This attribute can be used instead of codeValue to specify the code value at a lower level giving more control on the encoded data.
Fails if: Encoding for non-ASCII characters in the code value.
Default value: not set

**smartParse (Smart Parse)**
The smartParse property controls the parsing of Bar Code Boxes.

In the Properties view, this property is the **Smart Parse** property in the **Bar Code** category.

Specifies that the **codeValue** property for Bar Code Boxes is parsed in "smart" mode. By default it is parsed in raw mode.

In "smart" mode the **codeValue** is interpreted literally. An attempt is made to map the characters contained in the string to one or more codes choosing the shortest possible representation. Control characters cannot be displayed in this mode. Currently the functionality is only available for code-39-extended and code-128.

Type: **Boolean**. The default value is **false**.

**thinToGapRelation (Thin To Gap Relation)**
The thinToGapRelation property controls the ratio of thin bars to the gaps between individual digits.

In the Properties view, this property is the **Thin To Gap Relation** property in the **Bar Code** category.

The value of a gap is calculated by the formula GAPWIDTH=THINBARWIDTH/thinToGapRelation. This parameter applies only to the following code types:

- code-2-5-industrial
- code-2-5-inverted
- code-2-5-IATA
- code-2-5-interleaved
- code-2-5-matrix
- code-2-5-datalogic
- code-BCD-matrix
- code-11-matrix
- code-39
- code-39-extended
- code-32
- codabar 18
- codabar 2

Type: **Numeric**.

Default values:

- **0.5** for the "code-2-5-industrial" type.
- **1** for all other types.

**thinToThickRelation (Thin To Thick Relation)**
The thinToThickRelation property controls the ratio of thin bars to thick bars.

In the Properties view, this property is the **Thin To Thick Relation** property in the **Bar Code** category.

The value of a thick bar is calculated using the formula THICKBARWIDTH=THINBARWIDTH/thinToThickRelation. This parameter applies only to the following code types:

- code-2-5-industrial
- code-2-5-inverted
- code-2-5-IATA
- code-2-5-interleaved
- code-2-5-matrix
- code-2-5-datalogic
- code-BCD-matrix
- code-11-matrix
• code-39
• code-39-extended
• code-32
• codabar 2

Type: Numeric.

Default values: 1/3

Properties related to margins and borders
These are the margin and border-related properties for report elements.

• Margin width properties
• Border width properties
• Border style properties
• Border color properties
• Padding width properties

Related concepts
Modify an object's borders, margins, or padding on page 786
Any box object on a report design document can have margins, borders, and padding.

Margin width properties
The margin width properties control the width of the margins for a report object.

See Modify an object's borders, margins, or padding for illustrations.

All margin width properties are of type PXML, point value.

```
marginWidth
Sets the thickness of all an object's margins; can be overridden by specific margin properties.

marginRightWidth
Sets the thickness of an object's right margin; overrides the property marginWidth.

marginBottomWidth
Sets the thickness of an object's bottom margin; overrides the property marginWidth.

marginLeftWidth
Sets the thickness of an object's left margin; overrides the property marginWidth.

marginTopWidth
Sets the thickness of an object's top margin; overrides the property marginWidth.
```

Border width properties
The border width properties control the width of the borders for the report object.

See Modify an object's borders, margins, or padding for illustrations.

All border width properties are of type PXML, point value.

```
borderWidth
Sets the thickness of an object's borders; can be overridden by specific border properties. Default value is 2.

borderRightWidth
Sets the thickness of an object's right border; overrides the borderWidth property.

borderBottomWidth
Sets the thickness of an object's bottom border; overrides the borderWidth property.

borderLeftWidth
Sets the thickness of an object's left border; overrides the borderWidth property.
```
**borderTopWidth**
Sets the thickness of an object's top border; overrides the `borderWidth` property.

**Border style properties**
The border style properties control the style of the borders for the report object.
See [Modify an object's borders, margins, or padding](#) for illustrations.

- **borderStyle**
Sets the style for all an object's borders. Can be overridden by specific `borderStyle` properties.
  
  Type: `Enum`, choices are: none, solid, dotted, dashed, groove, ridge, inset, outset, double. Default is **none**.

- **borderRightStyle**
Sets the style for an object's right border. Can override the `borderStyle` property.
  
  Type: `Enum`, choices are: solid, dotted, dashed, groove, ridge, inset, outset, double.

- **borderBottomStyle**
Sets the style for an object's bottom border. Can override the `borderStyle` property.
  
  Type: `Enum`, choices are: solid, dotted, dashed, groove, ridge, inset, outset, double.

- **borderLeftStyle**
Sets the style for an object's left border. Can override the `borderStyle` property.
  
  Type: `Enum`, choices are: solid, dotted, dashed, groove, ridge, inset, outset, double.

- **borderTopStyle**
Sets the style for an object's top border. Can override the `borderStyle` property.
  
  Type: `Enum`, choices are: solid, dotted, dashed, groove, ridge, inset, outset, double.

- **roundedCorners**
Specifies that the object's border corners will be round. This applies to the border styles solid, dashed, and double only.
  
  Type: `Boolean`. Valid choices are True, False. The default value is False.

**Border color properties**
The border control properties control the color of the borders for the report object.
See [Modify an object's borders, margins, or padding](#) for illustrations.

All border color properties are of type `Color`. Valid colors are selected from the Edit Expression color palette.

- **borderColor**
Sets the color of all an object's borders. Can be overridden by specific `borderColor` properties. In the properties window, this property is identified by the label **Border Color**.

- **borderRightColor**
Sets the color of an object's right border. Can override the `borderColor` property.

- **borderBottomColor**
Sets the color of an object's bottom border. Can override the `borderColor` property.
borderLeftColor

Sets the color of an object's left border. Can override the borderColor property.

borderTopColor

Sets the color of an object's top border. Can override the borderColor property.

Padding width properties

The padding width properties control the width of the padding for the report object.

See Modify an object's borders, margins, or padding for illustrations.

All padding width properties are of type PXML, point value.

paddingWidth

Sets the width of all of an object's padding. Can be overridden by specific padding properties.

paddingRightWidth

Sets the width of an object's right padding. Overrides the paddingWidth property.

paddingBottomWidth

Sets the width of an object's bottom padding. Overrides the paddingWidth property.

paddingLeftWidth

Sets the width of an object's left padding. Overrides the paddingWidth property.

paddingTopWidth

Sets the width of an object's top padding. Overrides the paddingWidth property.

PDF cropping width properties

The PDF cropping width properties control the amount to be cropped from the margins of a PDF image.

All PDF cropping width properties are of type PXML. In the Properties view, they are found in the Pdf category.

cropRightWidth (Crop Right Width)

Sets the amount to crop from the right margin of the PDF.

cropBottomWidth (Crop Bottom Width)

Sets the amount to crop from the bottom margin of the PDF.

cropLeftWidth (Crop Left Width)

Sets the amount to crop from the left margin of the PDF.

cropTopWidth (Crop Top Width)

Sets the amount to crop from the top margin of the PDF.

Related concepts

PDF Box on page 864

A PDF Box displays one or more pages from a PDF document in the report.

Properties for Report Metadata

Report Metadata properties can be set in the report design document ( . 4 rp ).

For compatibility reports, which have no . 4 rp document, API functions are provided.

The metadata is inserted into the final document (such as SVG or PDF), if the format supports metadata.

All metadata properties are of type String. The following metadata properties are available:

title

Specifies the title of the object.

author

Specifies the metadata for the author of the report.

creator

Specifies the metadata for the creator of the report.

subject

Specifies the metadata for the subject of the report.

keywords

Specifies the metadata for the keyword of the report.
Related concepts
Report Design Document metadata on page 796
The Title, Author, Creator, Subject, and Keywords properties of the document root allow you to add metadata for a report.

Bar Codes
The report element container for a Bar Code is a Bar Code Box. This flow object draws bar codes.

See Properties related to bar codes for a list of properties specific to the Bar Code Box.

Additional properties are inherited from the Layout Node.

Properties that are specific to a bar code type are listed in the bar code type description.

Bar Code type listing
A table containing all bar codes that are supported by Genero Studio.

**Table 219: Bar Code Types**

<table>
<thead>
<tr>
<th>Bar Code Type</th>
<th>Number of Digits Supported</th>
<th>Normal size</th>
</tr>
</thead>
<tbody>
<tr>
<td>codabar-18 on page 913</td>
<td>varies</td>
<td>(calculated width x 20mm h)</td>
</tr>
<tr>
<td>codabar-2 on page 912</td>
<td>varies</td>
<td>(calculated width x 20mm h)</td>
</tr>
<tr>
<td>code-11-matrix on page 914</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-128 on page 915</td>
<td>varies</td>
<td>(calculated width x 6.5mm h)</td>
</tr>
<tr>
<td>code-2-5-datalogic on page 919</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-2-5-IATA on page 919</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-2-5-industrial on page 920</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-2-5-interleaved on page 920</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-2-5-inverted on page 920</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-2-5-matrix on page 920</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-BCD-matrix on page 920</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-32 on page 921</td>
<td>9</td>
<td>(calculated w x h)</td>
</tr>
<tr>
<td>code-39 on page 922</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-39-extended on page 927</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-93 on page 931</td>
<td>varies</td>
<td>(calculated w x h)</td>
</tr>
<tr>
<td>code-93-extended on page 933</td>
<td>varies</td>
<td>(calculated w X h)</td>
</tr>
<tr>
<td>data-matrix on page 937</td>
<td>varies</td>
<td></td>
</tr>
<tr>
<td>ean-8 on page 937</td>
<td>8</td>
<td>26.73 mm x 21.64 mm (w x h)</td>
</tr>
<tr>
<td>ean-13 on page 938</td>
<td>13</td>
<td>37.29 mm x 26.26 mm (w x h)</td>
</tr>
<tr>
<td>ean-code-128 on page 938</td>
<td>varies</td>
<td>(calculated width x 6.5mm h)</td>
</tr>
<tr>
<td>ean-data-matrix on page 938</td>
<td>varies</td>
<td></td>
</tr>
<tr>
<td>ean-supplemental-2 on page 938</td>
<td>2</td>
<td>6.6 mm x 26.26 mm (w x h)</td>
</tr>
<tr>
<td>ean-supplemental-5 on page 938</td>
<td>5</td>
<td>15.5 mm x 26.26 mm (w x h)</td>
</tr>
<tr>
<td>gs1-8 on page 939</td>
<td>8</td>
<td>26.73 mm x 21.64 mm (w x h)</td>
</tr>
<tr>
<td>gs1-13 on page 939</td>
<td>13</td>
<td>37.29 mm x 26.26 mm (w x h)</td>
</tr>
</tbody>
</table>
### Bar Code Type Details

A description of each code type, the expected value type, semantic constraints and size information.

The character set and semantic constraints depend on the code type selected.

**codabar-2**

Details on the codabar-2 bar code type.

Codabar 2 can be used to encode text of variable length by using characters from this character set:

#### Table 220: Character set for Codabar 2

<table>
<thead>
<tr>
<th>Reference Number</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>$</td>
</tr>
<tr>
<td>12</td>
<td>:</td>
</tr>
<tr>
<td>13</td>
<td>/</td>
</tr>
<tr>
<td>14</td>
<td>.</td>
</tr>
<tr>
<td>15</td>
<td>+</td>
</tr>
<tr>
<td>16</td>
<td>a</td>
</tr>
</tbody>
</table>
The first and the last character of any code must be either 'a', 'b', 'c' or 'd'. All other characters must have an ordinal value less than 16.

The last but one character is the checksum character that is calculated as follows: $\text{CS} = 16 - \left( \sum_{i=1}^{n} \text{Ref}(i) \right) \mod 16$ where Ref(i) is the reference number of the character i, and n is the total number of characters. Example: codeValue="a37859b" $\text{CS} = 16 - ((16+3+7+8+5+9+17) \mod 16), \text{CS} = 16 - (65 \mod 16), \text{CS} = 15$ Looking up reference number 15 yields the character '+' . The full code value including checksum is therefore: codeValue="a37859+b"

When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is calculated automatically.

The nominal height is 20mm. The nominal width of a thin bar is $\text{THINBARWIDTH} = 0.165$mm. The width of a thick bar is $\text{THICKBARWIDTH} = \text{THINBARWIDTH}/ \text{thinToThickRelation}$ where thinToThickRelation should take values between $1/3$ and $1/2$. Between digits a gap having the width $\text{GAPWIDTH} = \text{THINBARWIDTH}/ \text{thinToGapRelation}$ is drawn. The padding on both sides measures $10*\text{THINBARWIDTH}$.

The "American Blood Commission" defines a code type known as Codabar-ABC. There are two types of Codabar-ABC codes that can both be built using the Codabar 2 bar code type:

**Single bar Codabar-ABC** - this type is identical with Codabar 2, the constraint being that values have to be at least 5 digits long.

**Dual bar Codabar-ABC** - This type can be printed by printing two Codebar 2 horizontally adjacent to each other. The gap between the two bars should not exceed 15mm. Additionally the code of the first bar must end with a 'd' character while the second must start with the character 'd'. This code creates a valid Dual bar Codabar-ABC bar for the values "c1234d" and "d5678a".

codabar-18
Details on the codabar-18 bar code type.

Codabar 18 can be used to encode text of variable length by using characters from this character set:

**Table 221: Character set for Codabar 18**

<table>
<thead>
<tr>
<th>Reference Number</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>$</td>
</tr>
</tbody>
</table>
The first character of any code must be either 'a', 'b', 'c' or 'd'. The last character of any code must be either 't', 'n', '+' or 'e'. All other characters must have an ordinal value less than 16.

The last but one character is the checksum character that is calculated as follows: \( CS = 16 - (\text{Sum}(i=1 \text{ to } n \text{ of Ref}(i))) \mod 16 \) Where Ref(i) is the reference number of the character i, and n is the total number of characters. Example: codeValue="a37859n" CS=16-((16+3+7+8+5+9+17) mod 16), CS=16-(65 mod 16), CS=15 Looking up reference number 15 yields the character '+'. The full code value including checksum is therefore: codeValue="a37859+n"

When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated.

The nominal height is 20mm. The nominal width of a thin bar is THINBARWIDTH=0.165mm. The width of a character is 2.095mm. Between digits a gap with the width GAPWIDTH=THINBARWIDTH/thinToGapRelation is drawn. The padding on both sides measures 10*THINBARWIDTH.

code-11-matrix

Details on the code-11-matrix bar code type.

The code represents a character string with a variable number of characters. The string can contain the digits 0-9 and the '-' character. The number of digits can be specified by setting the noDigits attribute. The code can contain up to two check characters. The attribute noCheckDigits specifies how many check characters are used. If two check characters are used, the rightmost character is the 'K' checksum character and the last but one character is the 'C' checksum character. If only one checksum character is specified then the rightmost character is a 'C' checksum character. The 'C' checksum is calculated as \( C = (\text{Sum}(i=1 \text{ to } n \text{ of } ((i-1 \mod 10)+1) \times \text{Ref}(n-i+1))) \mod 11 \) and the 'K' checksum is calculated using \( K = (\text{Sum}(i=1 \text{ to } n \text{ of } ((i-1 \mod 9)+1) \times \text{Ref}(n-i+1))) \mod 11 \) where n specifies the number of characters to the left of the particular check digit and Ref(i) specifies the value of the character at position i, starting with the leftmost character having the value 1. For digits Ref() yields the digit value itself and for the '-' character Ref() yields the value 10.

Example calculating the 'C' checksum: codeValue="12-12345-67890", noDigits="16", noCheckDigits="2", n=14C=(1*0+2*9+3*8+4*7+5*6+6*5+7*4+8*3+9*2+0*1=140) \mod 11=350 \mod 11=8\) The K checksum can then be calculated as: n=15, Ref(15)=C=8K=(1*8+2*0+3*9+4*8+5*7+6*6+7*5+8*4+9*3+0*2+1*1=12345) \mod 11=350 \mod 11=9 resulting in the code value codeValue="12-12345-6789089".

If the value supplied in codeValue has the length noDigits- noCheckDigits then the system automatically calculates and supplies the check digits.

The nominal height is 1 in. Digits can differ in width so that two different values having the same number of digits can result in bar codes of differing width. The nominal width of a thin bar is THINBARWIDTH=0.0236in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/thinToThickRelation where thinToThickRelation should
take values between 1/3 and 1/2. Between digits a gap of width GAPWIDTH=THINBARWIDTH/\ thinToGapRelation
is drawn. The default relation value is 1. The padding on both sides measures 10*THINBARWIDTH.

code-128
Details on the code-128 bar code type.

Code 128 can be used to encode ASCII text of variable length. For this purpose characters can be selected from three
character sets, each containing 106 characters. Table 222: Available characters in the character sets A, B, and C on
page 915 lists the available characters in the character sets A, B and C.

Table 222: Available characters in the character sets A, B, and C

<table>
<thead>
<tr>
<th>Reference Number</th>
<th>Character Set A</th>
<th>Character Set B</th>
<th>Character Set C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SP</td>
<td>SP</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>!</td>
<td>!</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>&quot;</td>
<td>&quot;</td>
<td>02</td>
</tr>
<tr>
<td>3</td>
<td>#</td>
<td>#</td>
<td>03</td>
</tr>
<tr>
<td>4</td>
<td>$</td>
<td>$</td>
<td>04</td>
</tr>
<tr>
<td>5</td>
<td>%</td>
<td>%</td>
<td>05</td>
</tr>
<tr>
<td>6</td>
<td>&amp;</td>
<td>&amp;</td>
<td>06</td>
</tr>
<tr>
<td>7</td>
<td>'</td>
<td>'</td>
<td>07</td>
</tr>
<tr>
<td>8</td>
<td>(</td>
<td>(</td>
<td>08</td>
</tr>
<tr>
<td>9</td>
<td>)</td>
<td>)</td>
<td>09</td>
</tr>
<tr>
<td>10</td>
<td>*</td>
<td>*</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>+</td>
<td>+</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>.</td>
<td>.</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>-</td>
<td>-</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>:</td>
<td>:</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>/</td>
<td>/</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>5</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>6</td>
<td>6</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>7</td>
<td>7</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>9</td>
<td>9</td>
<td>25</td>
</tr>
<tr>
<td>26</td>
<td>:</td>
<td>:</td>
<td>26</td>
</tr>
<tr>
<td>27</td>
<td>;</td>
<td>;</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>&lt;</td>
<td>&lt;</td>
<td>28</td>
</tr>
<tr>
<td>Reference Number</td>
<td>Character Set A</td>
<td>Character Set B</td>
<td>Character Set C</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>29</td>
<td>=</td>
<td>=</td>
<td>29</td>
</tr>
<tr>
<td>30</td>
<td>&gt;</td>
<td>&gt;</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>?</td>
<td>?</td>
<td>31</td>
</tr>
<tr>
<td>32</td>
<td>@</td>
<td>@</td>
<td>32</td>
</tr>
<tr>
<td>33</td>
<td>A</td>
<td>A</td>
<td>33</td>
</tr>
<tr>
<td>34</td>
<td>B</td>
<td>B</td>
<td>34</td>
</tr>
<tr>
<td>35</td>
<td>C</td>
<td>C</td>
<td>35</td>
</tr>
<tr>
<td>36</td>
<td>D</td>
<td>D</td>
<td>36</td>
</tr>
<tr>
<td>37</td>
<td>E</td>
<td>E</td>
<td>37</td>
</tr>
<tr>
<td>38</td>
<td>F</td>
<td>F</td>
<td>38</td>
</tr>
<tr>
<td>39</td>
<td>G</td>
<td>G</td>
<td>39</td>
</tr>
<tr>
<td>40</td>
<td>H</td>
<td>H</td>
<td>40</td>
</tr>
<tr>
<td>41</td>
<td>I</td>
<td>I</td>
<td>41</td>
</tr>
<tr>
<td>42</td>
<td>J</td>
<td>J</td>
<td>42</td>
</tr>
<tr>
<td>43</td>
<td>K</td>
<td>K</td>
<td>43</td>
</tr>
<tr>
<td>44</td>
<td>L</td>
<td>L</td>
<td>44</td>
</tr>
<tr>
<td>45</td>
<td>M</td>
<td>M</td>
<td>45</td>
</tr>
<tr>
<td>46</td>
<td>N</td>
<td>N</td>
<td>46</td>
</tr>
<tr>
<td>47</td>
<td>O</td>
<td>O</td>
<td>47</td>
</tr>
<tr>
<td>48</td>
<td>P</td>
<td>P</td>
<td>48</td>
</tr>
<tr>
<td>49</td>
<td>Q</td>
<td>Q</td>
<td>49</td>
</tr>
<tr>
<td>50</td>
<td>R</td>
<td>R</td>
<td>50</td>
</tr>
<tr>
<td>51</td>
<td>S</td>
<td>S</td>
<td>51</td>
</tr>
<tr>
<td>52</td>
<td>T</td>
<td>T</td>
<td>52</td>
</tr>
<tr>
<td>53</td>
<td>U</td>
<td>U</td>
<td>53</td>
</tr>
<tr>
<td>54</td>
<td>V</td>
<td>V</td>
<td>54</td>
</tr>
<tr>
<td>55</td>
<td>W</td>
<td>W</td>
<td>55</td>
</tr>
<tr>
<td>56</td>
<td>X</td>
<td>X</td>
<td>56</td>
</tr>
<tr>
<td>57</td>
<td>Y</td>
<td>Y</td>
<td>57</td>
</tr>
<tr>
<td>58</td>
<td>Z</td>
<td>Z</td>
<td>58</td>
</tr>
<tr>
<td>59</td>
<td>[</td>
<td>[</td>
<td>59</td>
</tr>
<tr>
<td>60</td>
<td>\</td>
<td>\</td>
<td>60</td>
</tr>
<tr>
<td>61</td>
<td>]</td>
<td>]</td>
<td>61</td>
</tr>
<tr>
<td>62</td>
<td>^</td>
<td>^</td>
<td>62</td>
</tr>
<tr>
<td>63</td>
<td>_</td>
<td>_</td>
<td>63</td>
</tr>
<tr>
<td>64</td>
<td>NUL</td>
<td>\</td>
<td>64</td>
</tr>
<tr>
<td>Reference Number</td>
<td>Character Set A</td>
<td>Character Set B</td>
<td>Character Set C</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>65</td>
<td>SOH</td>
<td>a</td>
<td>65</td>
</tr>
<tr>
<td>66</td>
<td>STX</td>
<td>b</td>
<td>66</td>
</tr>
<tr>
<td>67</td>
<td>ETX</td>
<td>c</td>
<td>67</td>
</tr>
<tr>
<td>68</td>
<td>EOT</td>
<td>d</td>
<td>68</td>
</tr>
<tr>
<td>69</td>
<td>ENQ</td>
<td>e</td>
<td>69</td>
</tr>
<tr>
<td>70</td>
<td>ACK</td>
<td>f</td>
<td>70</td>
</tr>
<tr>
<td>71</td>
<td>BEL</td>
<td>g</td>
<td>71</td>
</tr>
<tr>
<td>72</td>
<td>BS</td>
<td>h</td>
<td>72</td>
</tr>
<tr>
<td>73</td>
<td>HT</td>
<td>i</td>
<td>73</td>
</tr>
<tr>
<td>74</td>
<td>LF</td>
<td>j</td>
<td>74</td>
</tr>
<tr>
<td>75</td>
<td>VT</td>
<td>k</td>
<td>75</td>
</tr>
<tr>
<td>76</td>
<td>FF</td>
<td>l</td>
<td>76</td>
</tr>
<tr>
<td>77</td>
<td>CR</td>
<td>m</td>
<td>77</td>
</tr>
<tr>
<td>78</td>
<td>SO</td>
<td>n</td>
<td>78</td>
</tr>
<tr>
<td>79</td>
<td>SI</td>
<td>o</td>
<td>79</td>
</tr>
<tr>
<td>80</td>
<td>DLE</td>
<td>p</td>
<td>80</td>
</tr>
<tr>
<td>81</td>
<td>DC1</td>
<td>q</td>
<td>81</td>
</tr>
<tr>
<td>82</td>
<td>DC2</td>
<td>r</td>
<td>82</td>
</tr>
<tr>
<td>83</td>
<td>DC3</td>
<td>s</td>
<td>83</td>
</tr>
<tr>
<td>84</td>
<td>DC4</td>
<td>t</td>
<td>84</td>
</tr>
<tr>
<td>85</td>
<td>NAK</td>
<td>u</td>
<td>85</td>
</tr>
<tr>
<td>86</td>
<td>SYN</td>
<td>v</td>
<td>86</td>
</tr>
<tr>
<td>87</td>
<td>ETB</td>
<td>w</td>
<td>87</td>
</tr>
<tr>
<td>88</td>
<td>CAN</td>
<td>x</td>
<td>88</td>
</tr>
<tr>
<td>89</td>
<td>EM</td>
<td>y</td>
<td>89</td>
</tr>
<tr>
<td>90</td>
<td>SUB</td>
<td>z</td>
<td>90</td>
</tr>
<tr>
<td>91</td>
<td>ESC</td>
<td>{</td>
<td>91</td>
</tr>
<tr>
<td>92</td>
<td>FS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>GS</td>
<td>}</td>
<td>93</td>
</tr>
<tr>
<td>94</td>
<td>RS</td>
<td>~</td>
<td>94</td>
</tr>
<tr>
<td>95</td>
<td>US</td>
<td>DEL</td>
<td>95</td>
</tr>
<tr>
<td>96</td>
<td>FNC3</td>
<td>FNC3</td>
<td>96</td>
</tr>
<tr>
<td>97</td>
<td>FNC2</td>
<td>FNC2</td>
<td>97</td>
</tr>
<tr>
<td>98</td>
<td>SHIFT</td>
<td>SHIFT</td>
<td>98</td>
</tr>
<tr>
<td>99</td>
<td>CODEC</td>
<td>CODEC</td>
<td>99</td>
</tr>
<tr>
<td>100</td>
<td>CODEB</td>
<td>FNC4</td>
<td>CODEB</td>
</tr>
</tbody>
</table>
The code value is expected as a comma-separated list of character names. It must start with a character set selection character STARTA, STARTB, or STARTC and must end with a checksum character followed by the STOP character. If these characters are omitted then the system calculates the checksum automatically and adds the required STOP character.

The control characters CODEA, CODEB and CODEC can be used to switch from one character set to another.

The control character SHIFT changes the character set for the immediately following character from A to B and vice versa.

**The smartParse property**

The smartParse property can be used when the code value consists solely of printable characters. This alleviates users of the need to manually select character sets. When enabled, the resulting bar code is encoded with a shortest possible encoding, for the given string, producing a minimally sized visual representation.

**EAN 128 (GS1 128) bar codes**

EAN 128 bar codes can be drawn using this bar code type.

**Note:** ean-code-128 on page 938 and gs1-code-128 on page 939 bar codes are synonymous.

Valid EAN 128 codes start with the sequence "STARTC,FNC1,"

**Tip:** With ean-code-128 or gs1-code-128, the data passes as a string (using smart parse) and the "STARTC,FNC1," is added by the engine. If the ean-code-128 or gs1-code-128 were not made available, you would have to manually encode the string starting with "STARTC,FNC1," and then manually select the appropriate character sets to encode the data.

What follows is a sequence of data packets. Each packet starts with a one digit application identifier (AI) from the C character set. The AI is followed by data. The type and amount of expected data is AI specific. The amount can be fixed or variable. In the case of variable amount of data, the end of the data must be indicated by a FNC1 character. Here is a table with some common AIs.

**Table 223: EAN 128 Bar Code Common AIs**

<table>
<thead>
<tr>
<th>AI</th>
<th>No of Data Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>18</td>
</tr>
<tr>
<td>01</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>up to 20 alphanumeric characters</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
</tr>
</tbody>
</table>

The complete list of AIs is country specific and is maintained by the local EAN organization. The textual representation requires AIs to be enclosed in round braces. Examples:
Table 224: EAN 128 Bar Code textual representation

<table>
<thead>
<tr>
<th>Textual representation</th>
<th>code value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(25)03x57</td>
<td>STARTC, FNC1, 25, 03, CODEB, x, 5, 7</td>
</tr>
<tr>
<td>(30)19(21)3456789012</td>
<td>STARTC, FNC1, 30, 19, CODEA, FNC1, CODEC, 21, 12, 34, 56, 78, 90, 12</td>
</tr>
</tbody>
</table>

If a code contains the control character FNC2 then the decoder does not transmit this value. Instead it appends the value to an internal storage. Only after the decoder encounters a value not containing the FNC2 control character, the decoder transmits the temporary storage and the value read. The temporary storage is then cleared. The purpose of this mechanism is to allow the breaking of long text sequences into several lines.

The effects of the control characters FNC3 and FNC4 are decoder specific.

Decoders remove the character STARTA, STARTB, STARTC, CODEA, CODEB, CODEC, SHIFT, FNC1, FNC2, FNC3, FNC4, STOP, and the checksum character from the data before displaying or transmitting the value.

The last but one character is the checksum character that is calculated as follows: \( CS = (\text{Ref}(1) + \sum_{i=2}^{n} (i-1) \times \text{Ref}(i)) \mod 103 \) where \( \text{Ref}(i) \) is the reference number of the character \( i \), and \( n \) is the total number of characters. Example: codeValue="STARTB,A,B,C"CS=(104+(1*33)+(2*34)+(3*35)) mod 103,CS=310 mod 103,CS=1 Looking up reference number 1 in character set B yields the exclamation mark ’!’ character. The full code value including checksum and stop character is therefore: codeValue="STARTB,A,B,C,!,STOP"

Height and width

The nominal height is 6.5mm. The width of the bar can be calculated using this formula: \( L/mm = (5.5Nc + 11Nab + 35) \times 0.19 \) where \( Nc \) is the number of characters from character set C and \( Nab \) denotes the number of characters from character sets A and B. On each side of the bar area 1.9mm padding is added.

code-2-5-datalogic

Details on the code-2-5-datalogic bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the noDigits attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars making a total of 5 bars per digit. The nominal width of a thin bar is \( \text{THINBARWIDTH} = 0.0236 \)in. The width of a thick bar is \( \text{THICKBARWIDTH} = \frac{\text{THINBARWIDTH}}{\text{thinToThickRelation}} \) where thinToThickRelation should take values between 1/3 and 1/2. Between digits a gap of width \( \text{GAPWIDTH} = \frac{\text{THINBARWIDTH}}{\text{thinToGapRelation}} \) is drawn. The padding on both sides measures 10*THINBARWIDTH.

code-2-5-IATA

Details on the code-2-5-IATA bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the noDigits attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars yielding a total of 5 bars per digit. The nominal width of a thin bar is \( \text{THINBARWIDTH} = 0.0236 \)in. The width of a thick bar is \( \text{THICKBARWIDTH} = \frac{\text{THINBARWIDTH}}{\text{thinToThickRelation}} \) where thinToThickRelation should take values between 1/3 and 1/2. Between digits a gap of width \( \text{GAPWIDTH} = \frac{\text{THINBARWIDTH}}{\text{thinToGapRelation}} \) is drawn. The padding on both sides measures 10*THINBARWIDTH.
**code-2-5-industrial**
Details on the code-2-5-industrial bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the noDigits attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars yielding a total of 5 bars per digit. The nominal width of a thin bar is THINBARWIDTH=0.0236in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/thinToThickRelation where thinToThickRelation should take values between 1/3 and 1/2. Between digits a gap of width GAPWIDTH=THINBARWIDTH/thinToGapRelation (Thin To Gap Relation) on page 907 is drawn. The padding on both sides measures 10*THINBARWIDTH.

**code-2-5-interleaved**
Details on the code-2-5-interleaved bar code type.

The code represents a decimal number with a variable number of digits. The number of digits must be a multiple of 2. The number of digits can be specified by setting the noDigits attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars resulting in a total of 5 bars per digit. The nominal width of a thin bar is THINBARWIDTH=0.0236in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/thinToThickRelation where thinToThickRelation should take values between 1/3 and 1/2. Between digits a gap of width GAPWIDTH=THINBARWIDTH/thinToGapRelation is drawn. The padding on both sides measures 10*THINBARWIDTH.

**code-2-5-inverted**
Details on the code-2-5-inverted bar code type.

The code is the same as code 2/5 industrial, the only difference being that the gaps are drawn instead of the bars.

**code-2-5-matrix**
Details on the code-2-5-matrix bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the noDigits attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars yielding a total of 5 bars per digit. The nominal width of a thin bar is THINBARWIDTH=0.0236in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/thinToThickRelation where thinToThickRelation should take values between 1/3 and 1/2. Between digits a gap of width GAPWIDTH=THINBARWIDTH/thinToGapRelation is drawn. The padding on both sides measures 10*THINBARWIDTH.

**code-BCD-matrix**
Details on the code-BCD-matrix bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the noDigits attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated by the system.

The nominal height is 1in. Digits can differ in width so that two different values having the same number of digits can result in bar codes of differing width. The nominal width of a thin bar is THINBARWIDTH=0.0236in. The width of
a thick bar is $\text{THICKBARWIDTH} = \text{THINBARWIDTH}/\text{thinToThickRelation}$ where \text{thinToThickRelation} should take values between $1/3$ and $1/2$. Between digits a gap of width $\text{GAPWIDTH} = \text{THINBARWIDTH}/\text{thinToGapRelation}$ is drawn. The default relation value is 1. The padding on both sides measures $10*\text{THINBARWIDTH}$.

code-32
Details on the code-32 bar code type.

The code represents a decimal number with 9 digits. The last digit is the check character. The check character is calculated as follows:

\[ CS = \text{Sum}(i=1 \text{ to } 8 \text{ of } CT((i-1)\%2*2*\text{Ref}(i))) \mod 10 \]

where $\text{Ref}(i)$ denotes the value of the digit at position $i$ (the leftmost digit has the index 1) and $\text{CT}()$ denotes the cross total of its argument (e.g. $\text{CT}(18)=1+8=9$).

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a 8 digit value then the check digit is automatically calculated.

The specified 9 digit decimal value is translated into a 6 digit base 32 value which is then drawn using characters from the base 39 character set and bar drawing scheme. This character table is used to encode the 6 digit base 32 value:

**Table 225: 32 character table is used to encode the 6 digit base 32 value**

<table>
<thead>
<tr>
<th>Reference Number</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>B</td>
</tr>
<tr>
<td>10</td>
<td>C</td>
</tr>
<tr>
<td>11</td>
<td>C</td>
</tr>
<tr>
<td>12</td>
<td>D</td>
</tr>
<tr>
<td>13</td>
<td>F</td>
</tr>
<tr>
<td>14</td>
<td>G</td>
</tr>
<tr>
<td>15</td>
<td>H</td>
</tr>
<tr>
<td>16</td>
<td>J</td>
</tr>
<tr>
<td>17</td>
<td>K</td>
</tr>
<tr>
<td>18</td>
<td>L</td>
</tr>
<tr>
<td>19</td>
<td>M</td>
</tr>
<tr>
<td>20</td>
<td>N</td>
</tr>
<tr>
<td>21</td>
<td>P</td>
</tr>
<tr>
<td>22</td>
<td>Q</td>
</tr>
<tr>
<td>23</td>
<td>R</td>
</tr>
</tbody>
</table>
Based on this table the following bar codes will produce identical bar pattern: `<BARCODEBOX codeType="code 32" check="true" codeValue="026089019" noText="true" orientation="vertical" mirrored="true"/> <BARCODEBOX codeType="code 39" check="false" codeValue="0SW5KV" noText="true" orientation="vertical" mirrored="true"/>
` > Note that the bars are drawn without text by setting noText="true". This is necessary to produce identical output since otherwise the code 32 box draws the decimal number "026089019" while the code 39 box would draw the string "0SW5KV".

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars making a total of 5 bars per digit. The nominal width of a thin bar is THINBARWIDTH=0.0197in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/thinToThickRelation where thinToThickRelation should take values between 1/3 and 1/2. Between digits a gap with the width GAPWIDTH=THINBARWIDTH/thinToGapRelation is drawn. The padding on both sides measures 10*THINBARWIDTH.

code-39
Details on the code-39 bar code type.

Code 39 can be used to encode ASCII text of variable length. Characters can be selected from this set of characters.

**Table 226: Characters for encoding ASCII text of variable length for Code 39**

<table>
<thead>
<tr>
<th>Reference Number</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>Reference Number</td>
<td>Character</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
</tr>
<tr>
<td>16</td>
<td>G</td>
</tr>
<tr>
<td>17</td>
<td>H</td>
</tr>
<tr>
<td>18</td>
<td>I</td>
</tr>
<tr>
<td>19</td>
<td>J</td>
</tr>
<tr>
<td>20</td>
<td>K</td>
</tr>
<tr>
<td>21</td>
<td>L</td>
</tr>
<tr>
<td>22</td>
<td>M</td>
</tr>
<tr>
<td>23</td>
<td>N</td>
</tr>
<tr>
<td>24</td>
<td>O</td>
</tr>
<tr>
<td>25</td>
<td>P</td>
</tr>
<tr>
<td>26</td>
<td>Q</td>
</tr>
<tr>
<td>27</td>
<td>R</td>
</tr>
<tr>
<td>28</td>
<td>S</td>
</tr>
<tr>
<td>29</td>
<td>T</td>
</tr>
<tr>
<td>30</td>
<td>U</td>
</tr>
<tr>
<td>31</td>
<td>V</td>
</tr>
<tr>
<td>32</td>
<td>W</td>
</tr>
<tr>
<td>33</td>
<td>X</td>
</tr>
<tr>
<td>34</td>
<td>Y</td>
</tr>
<tr>
<td>35</td>
<td>Z</td>
</tr>
<tr>
<td>36</td>
<td>-</td>
</tr>
<tr>
<td>37</td>
<td>-</td>
</tr>
<tr>
<td>38</td>
<td>$</td>
</tr>
<tr>
<td>39</td>
<td>/</td>
</tr>
<tr>
<td>40</td>
<td>+</td>
</tr>
<tr>
<td>41</td>
<td>%</td>
</tr>
</tbody>
</table>

The last but one character is the checksum character that is calculated as follows: $CS = \text{Sum}(i=1 \text{ to } n \text{ of } \text{Ref}(i)) \mod 43$ where $\text{Ref}(i)$ is the reference number of the character $i$, and $n$ is the total number of characters. Example: codeValue="DATALOGIC" $CS = (13+10+29+10+21+24+16+18+12) \mod 43, CS = 153 \mod 43, CS = 24$ Looking up reference number 24 yields the character 'O'. The full code value including checksum and stop character is therefore: codeValue="DATALOGICO"

Some scanners support an "extended mode" in which the scanner recognizes special two-character sequences of the code 39 character set and decodes these as ASCII characters. With this method the full 128 character ASCII character set can be encoded using the 43 basic characters of code 39. Scanners are switched into "extended mode" by a bar containing the sequence "+$". The sequence "-$" returns the scanner into regular mode. This table lists the character sequences needed to encode the ASCII character set in "extended mode".
Table 227: Character sequences needed to encode the ASCII character set in "extended mode"

<table>
<thead>
<tr>
<th>ASCII code</th>
<th>Code 39 sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td>%U</td>
</tr>
<tr>
<td>SOH</td>
<td>$A</td>
</tr>
<tr>
<td>STX</td>
<td>$B</td>
</tr>
<tr>
<td>ETX</td>
<td>$C</td>
</tr>
<tr>
<td>EOT</td>
<td>$D</td>
</tr>
<tr>
<td>ENQ</td>
<td>$E</td>
</tr>
<tr>
<td>ACK</td>
<td>$F</td>
</tr>
<tr>
<td>BEL</td>
<td>$G</td>
</tr>
<tr>
<td>BS</td>
<td>$H</td>
</tr>
<tr>
<td>HT</td>
<td>$I</td>
</tr>
<tr>
<td>LF</td>
<td>$J</td>
</tr>
<tr>
<td>VT</td>
<td>$K</td>
</tr>
<tr>
<td>FF</td>
<td>$L</td>
</tr>
<tr>
<td>CR</td>
<td>$M</td>
</tr>
<tr>
<td>SO</td>
<td>$N</td>
</tr>
<tr>
<td>SI</td>
<td>$O</td>
</tr>
<tr>
<td>DLE</td>
<td>$P</td>
</tr>
<tr>
<td>DC1</td>
<td>$Q</td>
</tr>
<tr>
<td>DC2</td>
<td>$R</td>
</tr>
<tr>
<td>DC3</td>
<td>$S</td>
</tr>
<tr>
<td>DC4</td>
<td>$T</td>
</tr>
<tr>
<td>NAK</td>
<td>$U</td>
</tr>
<tr>
<td>SYN</td>
<td>$V</td>
</tr>
<tr>
<td>ETB</td>
<td>$W</td>
</tr>
<tr>
<td>CAN</td>
<td>$X</td>
</tr>
<tr>
<td>EM</td>
<td>$Y</td>
</tr>
<tr>
<td>SUB</td>
<td>$Z</td>
</tr>
<tr>
<td>ESC</td>
<td>%A</td>
</tr>
<tr>
<td>FS</td>
<td>%B</td>
</tr>
<tr>
<td>GS</td>
<td>%C</td>
</tr>
<tr>
<td>RS</td>
<td>%D</td>
</tr>
<tr>
<td>US</td>
<td>%E</td>
</tr>
<tr>
<td>SP</td>
<td>/A</td>
</tr>
<tr>
<td>'</td>
<td>/A</td>
</tr>
<tr>
<td>&quot;</td>
<td>/B</td>
</tr>
<tr>
<td>ASCH code</td>
<td>Code 39 sequence</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
</tr>
<tr>
<td>#</td>
<td>/C</td>
</tr>
<tr>
<td>$</td>
<td>/D</td>
</tr>
<tr>
<td>%</td>
<td>/E</td>
</tr>
<tr>
<td>&amp;</td>
<td>/F</td>
</tr>
<tr>
<td>;</td>
<td>/G</td>
</tr>
<tr>
<td>(</td>
<td>/H</td>
</tr>
<tr>
<td>)</td>
<td>/I</td>
</tr>
<tr>
<td>*</td>
<td>/J</td>
</tr>
<tr>
<td>+</td>
<td>/K</td>
</tr>
<tr>
<td>,</td>
<td>/L</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>/</td>
<td>/O</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>:</td>
<td>/Z</td>
</tr>
<tr>
<td>;</td>
<td>%F</td>
</tr>
<tr>
<td>&lt;</td>
<td>%G</td>
</tr>
<tr>
<td>=</td>
<td>%H</td>
</tr>
<tr>
<td>&gt;</td>
<td>%I</td>
</tr>
<tr>
<td>?</td>
<td>%J</td>
</tr>
<tr>
<td>@</td>
<td>%V</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>ASCII code</td>
<td>Code 39 sequence</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>J</td>
<td>J</td>
</tr>
<tr>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>Q</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Z</td>
<td>Z</td>
</tr>
<tr>
<td>[</td>
<td>%K</td>
</tr>
<tr>
<td>\</td>
<td>%L</td>
</tr>
<tr>
<td>]</td>
<td>%M</td>
</tr>
<tr>
<td>^</td>
<td>%N</td>
</tr>
<tr>
<td>_</td>
<td>%O</td>
</tr>
<tr>
<td>`</td>
<td>%W</td>
</tr>
<tr>
<td>a</td>
<td>+A</td>
</tr>
<tr>
<td>b</td>
<td>+B</td>
</tr>
<tr>
<td>c</td>
<td>+C</td>
</tr>
<tr>
<td>d</td>
<td>+D</td>
</tr>
<tr>
<td>e</td>
<td>+E</td>
</tr>
<tr>
<td>f</td>
<td>+F</td>
</tr>
<tr>
<td>g</td>
<td>+G</td>
</tr>
<tr>
<td>h</td>
<td>+H</td>
</tr>
<tr>
<td>i</td>
<td>+I</td>
</tr>
<tr>
<td>j</td>
<td>+J</td>
</tr>
</tbody>
</table>
The code type "code 39 extended" automatically creates the necessary two-character sequences and can be used as a more convenient way of creating extended code 39 bar codes.

If the value specified by codeValue is shorter by one character than the number of digits specified by noDigits then the checksum character is automatically calculated and the character is appended to codeValue.

The nominal height is 1in. The nominal width of a thin bar is THINBARWIDTH=0.0197in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/thinToThickRelation where thinToThickRelation should take values between 1/3 and 1/2. Between digits a gap with the width GAPWIDTH=THINBARWIDTH/thinToGapRelation is drawn. The default relation value is 1. The padding on both sides measures 10*THINBARWIDTH.

code-39-extended
Details on the code-39-extended bar code type.

Code 39 Extended can be used to encode text of variable length using the ASCII character set. Depending on the value of the smartParse property, the code value is either expected as a comma-separated list of ASCII character names or as a plain string. Table 228: Code 39 Extended names and the textual representation that is used in printout on page 927 lists the names and the textual representation that is used in printout.

Table 228: Code 39 Extended names and the textual representation that is used in printout

<table>
<thead>
<tr>
<th>ASCII code</th>
<th>Code 39 sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>+K</td>
</tr>
<tr>
<td>l</td>
<td>+L</td>
</tr>
<tr>
<td>m</td>
<td>+M</td>
</tr>
<tr>
<td>n</td>
<td>+N</td>
</tr>
<tr>
<td>o</td>
<td>+O</td>
</tr>
<tr>
<td>p</td>
<td>+P</td>
</tr>
<tr>
<td>q</td>
<td>+Q</td>
</tr>
<tr>
<td>r</td>
<td>+R</td>
</tr>
<tr>
<td>s</td>
<td>+S</td>
</tr>
<tr>
<td>t</td>
<td>+T</td>
</tr>
<tr>
<td>u</td>
<td>+U</td>
</tr>
<tr>
<td>v</td>
<td>+V</td>
</tr>
<tr>
<td>w</td>
<td>+W</td>
</tr>
<tr>
<td>x</td>
<td>+X</td>
</tr>
<tr>
<td>y</td>
<td>+Y</td>
</tr>
<tr>
<td>z</td>
<td>+Z</td>
</tr>
<tr>
<td>{</td>
<td>%P</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td>%R</td>
</tr>
<tr>
<td>~</td>
<td>%S</td>
</tr>
<tr>
<td>DEL</td>
<td>%T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ASCII code</th>
<th>Textual representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td>&lt;NUL&gt;</td>
</tr>
<tr>
<td>ASCII code</td>
<td>Textual representation</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>SOH</td>
<td>&lt;SOH&gt;</td>
</tr>
<tr>
<td>STX</td>
<td>&lt;STX&gt;</td>
</tr>
<tr>
<td>ETX</td>
<td>&lt;ETX&gt;</td>
</tr>
<tr>
<td>EOT</td>
<td>&lt;EOT&gt;</td>
</tr>
<tr>
<td>ENQ</td>
<td>&lt;ENQ&gt;</td>
</tr>
<tr>
<td>ACK</td>
<td>&lt;ACK&gt;</td>
</tr>
<tr>
<td>BEL</td>
<td>&lt;BEL&gt;</td>
</tr>
<tr>
<td>BS</td>
<td>&lt;BS&gt;</td>
</tr>
<tr>
<td>HT</td>
<td>&lt;HT&gt;</td>
</tr>
<tr>
<td>LF</td>
<td>&lt;LF&gt;</td>
</tr>
<tr>
<td>VT</td>
<td>&lt;VT&gt;</td>
</tr>
<tr>
<td>FF</td>
<td>&lt;FF&gt;</td>
</tr>
<tr>
<td>CR</td>
<td>&lt;CR&gt;</td>
</tr>
<tr>
<td>SO</td>
<td>&lt;SO&gt;</td>
</tr>
<tr>
<td>SI</td>
<td>&lt;SI&gt;</td>
</tr>
<tr>
<td>DLE</td>
<td>&lt;DLE&gt;</td>
</tr>
<tr>
<td>DC1</td>
<td>&lt;DC1&gt;</td>
</tr>
<tr>
<td>DC2</td>
<td>&lt;DC2&gt;</td>
</tr>
<tr>
<td>DC3</td>
<td>&lt;DC3&gt;</td>
</tr>
<tr>
<td>DC4</td>
<td>&lt;DC4&gt;</td>
</tr>
<tr>
<td>NAK</td>
<td>&lt;NAK&gt;</td>
</tr>
<tr>
<td>SYN</td>
<td>&lt;SYN&gt;</td>
</tr>
<tr>
<td>ETB</td>
<td>&lt;ETB&gt;</td>
</tr>
<tr>
<td>CAN</td>
<td>&lt;CAN&gt;</td>
</tr>
<tr>
<td>EM</td>
<td>&lt;EM&gt;</td>
</tr>
<tr>
<td>SUB</td>
<td>&lt;SUB&gt;</td>
</tr>
<tr>
<td>ESC</td>
<td>&lt;ESC&gt;</td>
</tr>
<tr>
<td>FS</td>
<td>&lt;FS&gt;</td>
</tr>
<tr>
<td>GS</td>
<td>&lt;GS&gt;</td>
</tr>
<tr>
<td>RS</td>
<td>&lt;RS&gt;</td>
</tr>
<tr>
<td>US</td>
<td>&lt;US&gt;</td>
</tr>
<tr>
<td>SP</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td>#</td>
</tr>
<tr>
<td></td>
<td>$</td>
</tr>
<tr>
<td>ASCH code</td>
<td>Textual representation</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------</td>
</tr>
<tr>
<td>%</td>
<td>%</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
<tr>
<td>,</td>
<td>,</td>
</tr>
<tr>
<td>(</td>
<td>(</td>
</tr>
<tr>
<td>)</td>
<td>)</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>COMMA</td>
<td>,</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>@</td>
<td>@</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>ASCII code</td>
<td>Textual representation</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>J</td>
<td>J</td>
</tr>
<tr>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>Q</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Z</td>
<td>Z</td>
</tr>
<tr>
<td>[</td>
<td>[</td>
</tr>
<tr>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>]</td>
<td>]</td>
</tr>
<tr>
<td>^</td>
<td>^</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>e</td>
<td>e</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
</tr>
<tr>
<td>h</td>
<td>h</td>
</tr>
<tr>
<td>i</td>
<td>i</td>
</tr>
<tr>
<td>j</td>
<td>j</td>
</tr>
<tr>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>l</td>
<td>l</td>
</tr>
</tbody>
</table>
**ASCII code** | **Textual representation**
--- | ---
m | m
n | n
o | o
p | p
q | q
r | r
s | s
t | t
u | u
v | v
w | w
x | x
y | y
z | z
{ | {
| | |
} | }
~ | ~
DEL | <DEL>

**code-93**
Details on the code-93 bar code type.

Code 93 can be used to encode ASCII text of variable length. Characters can be selected from this set of characters. You can use the code-93-extended type to encode the full 128 character ASCII character set using the 47 basic characters of code 93.

**Table 229: code-93**

<table>
<thead>
<tr>
<th>Reference Number</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Reference Number</td>
<td>Character</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
</tr>
<tr>
<td>16</td>
<td>G</td>
</tr>
<tr>
<td>17</td>
<td>H</td>
</tr>
<tr>
<td>18</td>
<td>I</td>
</tr>
<tr>
<td>19</td>
<td>J</td>
</tr>
<tr>
<td>20</td>
<td>K</td>
</tr>
<tr>
<td>21</td>
<td>L</td>
</tr>
<tr>
<td>22</td>
<td>M</td>
</tr>
<tr>
<td>23</td>
<td>N</td>
</tr>
<tr>
<td>24</td>
<td>O</td>
</tr>
<tr>
<td>25</td>
<td>P</td>
</tr>
<tr>
<td>26</td>
<td>Q</td>
</tr>
<tr>
<td>27</td>
<td>R</td>
</tr>
<tr>
<td>28</td>
<td>S</td>
</tr>
<tr>
<td>29</td>
<td>T</td>
</tr>
<tr>
<td>30</td>
<td>U</td>
</tr>
<tr>
<td>31</td>
<td>V</td>
</tr>
<tr>
<td>32</td>
<td>W</td>
</tr>
<tr>
<td>33</td>
<td>X</td>
</tr>
<tr>
<td>34</td>
<td>Y</td>
</tr>
<tr>
<td>35</td>
<td>Z</td>
</tr>
<tr>
<td>36</td>
<td>-</td>
</tr>
<tr>
<td>37</td>
<td>.</td>
</tr>
<tr>
<td>38</td>
<td>$</td>
</tr>
<tr>
<td>39</td>
<td>/</td>
</tr>
<tr>
<td>40</td>
<td>+</td>
</tr>
<tr>
<td>41</td>
<td>%</td>
</tr>
<tr>
<td>42</td>
<td>!</td>
</tr>
<tr>
<td>43</td>
<td>?</td>
</tr>
<tr>
<td>44</td>
<td>\</td>
</tr>
</tbody>
</table>
The textual representation of the last four characters "!?|" can be configured by setting the controlCharacters attribute.

noCheckDigits specifies how many check characters are used. In the case of two check characters the rightmost character is the 'K' checksum character and the last but one character is the 'C' checksum character. If only one checksum character is specified then the rightmost character is a 'C' checksum character. The 'C' checksum is calculated as follows: $C=(\text{Sum}(i=1 \text{ to } n \text{ of } ((i-1 \text{ mod 20}+1)\times\text{Ref}(n-i+1))) \mod 47$ and the 'K' checksum is calculated as follows: $K=(\text{Sum}(i=1 \text{ to } n \text{ of } ((i-1 \text{ mod 15}+1)\times\text{Ref}(n-i+1))) \mod 47$ where $n$ specifies the number of characters left of the particular check digit and Ref(i) specifies the reference value of the character at position i starting with the leftmost character having the index value 1. Reference numbers can be looked up in the first column. Example calculating the 'C' checksum: codeValue="DATALOGIC", noDigits="11", noCheckDigits="2", n=9C=(1\times12+2\times18+3\times16+4\times24+\ldots+7\times29+8\times10+9\times13) \mod 47=757 \mod 47=5$ The K checksum can then be calculated as: $n=10$, Ref(10)=C=5K=(1\times5+2\times12+3\times18+4\times16+\ldots+8\times29+9\times10+10\times13) \mod 47=915 \mod 47=22$ resulting in the code value codeValue="DATALOGIC5M".

If the value supplied in codeValue has the length noDigits- noCheckDigits then the system automatically calculates and supplies the check digits.

code-93-extended
Details on the code-93-extended bar code type.

Some scanners support an "extended mode" in which the scanner recognizes special two-character sequences of the code 93 character set and decodes these as ASCII characters. With this mode, the full 128 character ASCII character set can be encoded using the 47 basic characters of code 93. T

The nominal height is 1in. The nominal width of a bar is BARWIDTH=0.022in. The width of the entire bar is $(1+(\text{noDigits}+2)\times9)\times\text{BARWIDTH}$. The padding on both sides measures 10*BARWIDTH.

This code type automatically creates the necessary two-character sequences and can be used as a more convenient way of creating extended code 93 bar codes.

Table 230: code-93-extended sequence table

<table>
<thead>
<tr>
<th>ASCII code</th>
<th>Code 93 sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td>?U</td>
</tr>
<tr>
<td>SOH</td>
<td>!A</td>
</tr>
<tr>
<td>STX</td>
<td>!B</td>
</tr>
<tr>
<td>ETX</td>
<td>!C</td>
</tr>
<tr>
<td>EOT</td>
<td>!D</td>
</tr>
<tr>
<td>ENQ</td>
<td>!E</td>
</tr>
<tr>
<td>ACK</td>
<td>!F</td>
</tr>
<tr>
<td>BEL</td>
<td>!G</td>
</tr>
<tr>
<td>BS</td>
<td>!H</td>
</tr>
<tr>
<td>HT</td>
<td>!I</td>
</tr>
<tr>
<td>LF</td>
<td>!J</td>
</tr>
<tr>
<td>VT</td>
<td>!K</td>
</tr>
<tr>
<td>FF</td>
<td>!L</td>
</tr>
<tr>
<td>CR</td>
<td>!M</td>
</tr>
<tr>
<td>ASCII code</td>
<td>Code 93 sequence</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
</tr>
<tr>
<td>SO</td>
<td>!N</td>
</tr>
<tr>
<td>SI</td>
<td>!O</td>
</tr>
<tr>
<td>DLE</td>
<td>!P</td>
</tr>
<tr>
<td>DC1</td>
<td>!Q</td>
</tr>
<tr>
<td>DC2</td>
<td>!R</td>
</tr>
<tr>
<td>DC3</td>
<td>!S</td>
</tr>
<tr>
<td>DC4</td>
<td>!T</td>
</tr>
<tr>
<td>NAK</td>
<td>!U</td>
</tr>
<tr>
<td>SYN</td>
<td>!V</td>
</tr>
<tr>
<td>ETB</td>
<td>!W</td>
</tr>
<tr>
<td>CAN</td>
<td>!X</td>
</tr>
<tr>
<td>EM</td>
<td>!Y</td>
</tr>
<tr>
<td>SUB</td>
<td>!Z</td>
</tr>
<tr>
<td>ESC</td>
<td>?A</td>
</tr>
<tr>
<td>FS</td>
<td>?B</td>
</tr>
<tr>
<td>GS</td>
<td>?C</td>
</tr>
<tr>
<td>RS</td>
<td>?D</td>
</tr>
<tr>
<td>US</td>
<td>?E</td>
</tr>
<tr>
<td>SP</td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>\A</td>
</tr>
<tr>
<td>&quot;</td>
<td>\B</td>
</tr>
<tr>
<td>#</td>
<td>\C</td>
</tr>
<tr>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>%</td>
<td>%</td>
</tr>
<tr>
<td>&amp;</td>
<td>\F</td>
</tr>
<tr>
<td>'</td>
<td>\G</td>
</tr>
<tr>
<td>(</td>
<td>\H</td>
</tr>
<tr>
<td>)</td>
<td>\I</td>
</tr>
<tr>
<td>*</td>
<td>\J</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>COMMA</td>
<td>\L</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ASCII code</td>
<td>Code 93 sequence</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>;</td>
<td>\Z</td>
</tr>
<tr>
<td>;</td>
<td>?F</td>
</tr>
<tr>
<td>&lt;</td>
<td>?G</td>
</tr>
<tr>
<td>=</td>
<td>?H</td>
</tr>
<tr>
<td>&gt;</td>
<td>?I</td>
</tr>
<tr>
<td>?</td>
<td>?J</td>
</tr>
<tr>
<td>@</td>
<td>?V</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>J</td>
<td>J</td>
</tr>
<tr>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>Q</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>ASCII code</td>
<td>Code 93 sequence</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
</tr>
<tr>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Z</td>
<td>Z</td>
</tr>
<tr>
<td>[</td>
<td>?K</td>
</tr>
<tr>
<td>\</td>
<td>?L</td>
</tr>
<tr>
<td>]</td>
<td>?M</td>
</tr>
<tr>
<td>^</td>
<td>?N</td>
</tr>
<tr>
<td>_</td>
<td>?O</td>
</tr>
<tr>
<td>`</td>
<td>?W</td>
</tr>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
</tr>
<tr>
<td>j</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
</tr>
<tr>
<td>o</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td></td>
</tr>
<tr>
<td>q</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td></td>
</tr>
<tr>
<td>u</td>
<td></td>
</tr>
<tr>
<td>v</td>
<td></td>
</tr>
<tr>
<td>w</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td></td>
</tr>
<tr>
<td>ASCII code</td>
<td>Code 93 sequence</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
</tr>
<tr>
<td>z</td>
<td>\Z</td>
</tr>
<tr>
<td>{</td>
<td>\P</td>
</tr>
<tr>
<td>[</td>
<td>\Q</td>
</tr>
<tr>
<td>}</td>
<td>\R</td>
</tr>
<tr>
<td>~</td>
<td>\S</td>
</tr>
<tr>
<td>DEL</td>
<td>\T</td>
</tr>
</tbody>
</table>

**data-matrix**
Details on the data-matrix bar code type.

A Data matrix bar code can be used to encode text and binary data of variable length. It is possible to encode up to 1558 code words. Text, binary data and numeric data are compressed into code words so that the limits for these types are:

- Alphanumeric data - up to 2335 characters
- 8-bit byte data - 1555 characters
- Numeric data - 3116 digits

The limits for mixed text are lower, as control characters are inserted to switch between the different compression modes.

Data matrix symbols exist in several flavors with varying methods and degrees of error correction. Of the possible options ECC 000, ECC 050, ECC 080, ECC 100, ECC 140 and ECC 200 this implementation supports only ECC 200 using Reed Solomon error correction.

Special control characters like FNC1, ECI and "structured append" are currently not available.

The conversion of the data specified in the `codeValue` attribute to the internal representation is done by an algorithm that minimizes space. Byte values that cannot be represented in an XML document (for example all characters lower than 0x20 except 0x9, 0xa and 0xd) can be represented by a backslash (\) character followed by a 3 digit octal literal. The backslash character itself can be escaped by a sequence of two backslash characters.

The current implementation can encode any character that exists in the code page ISO-8859-1 (Latin 1). Any attempt to encode other characters will fail. Future versions will insert ECI control characters to switch to other code pages if the character is available there.

The `preferRectangularSymbols` attribute is unique to this bar code type. If you are concerned about running out of space in the vertical of the page, you might prefer a symbol that is wider than it is high. This property produces a rectangular shaped symbol if the encoded data does not exceed 49 code words.

**ean-8**
Details on the ean-8 bar code type.

**Note:** The ean-8 and gs1-8 on page 939 bar codes are synonymous. The gs1-8 synonym can be used for the ean-8 bar code.

The code represents a 8 digit decimal number. First two digits select the country. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with such a 7 digit value then the 8th digit is automatically calculated by the system.

The nominal size is 26.73mm x 21.64mm (w x h). The bar code area has the measurements 22.11mm x 19.88mm. The left padding measures 2.31mm.
ean-13
Details on the ean-13 bar code type.

**Note:** The ean-13 and **gs1-13** on page 939 bar codes are synonymous. The gs1-13 synonym can be used for the ean-13 bar code.

The code represents a 13-digit decimal number. First two digits are the flag code, the next ten digits are the data characters and the last digit is the check character. The check character is calculated as follows:

1. Designate the rightmost character odd.
2. Sum all of the characters in the odd positions and multiply the result by three.
3. Sum all of the characters in the even positions.
4. Add the odd and even totals from steps two and three.
5. Determine the smallest number that, when added to the result from step four, will result in a multiple of 10. This is the check character.

In Europe the first two characters (flag code) are the country identifier and the data characters are split into two groups of five digits each where the first five are a company identifier and the latter five identify an article within that company. The last digit is the check character.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with such a 12 digit value then the 13th digit is automatically calculated by the system.

The nominal size is 37.29mm x 26.26mm (w x h). The bar code area has the measurements 31.35mm x 24.50mm. The left padding measures 3.63mm.

ean-code-128
Details on the ean-code-128 bar code type.

Based on the **code-128** on page 915 bar code.

**Note:** The ean-code-128 and **gs1-code-128** on page 939 bar codes are synonymous. The gs1-code-128 synonym can be used for the ean-code-128 bar code.

ean-data-matrix
Details on the ean-data-matrix bar code type.

Based on the **data-matrix** bar code. The ean-data-matrix supports smart parse, eliminating the need to manually insert the FNC1 character into the bar code.

**Note:** The ean-data-matrix and **gs1-data-matrix** on page 939 bar codes are synonymous. The gs1-data-matrix synonym can be used for the ean-data-matrix bar code.

ean-supplemental-2
Details on the ean-supplemental-2 bar code type.

The code represents a 2 digit decimal number that can be used in conjunction with the ean 13 bar code type. Typically the code is placed to the right of the ean 13 bar code. The gap between the rightmost bar of the ean 13 code and the leftmost bar of the supplemental code should not be less than 2.31mm and should not exceed 3.3mm.

The nominal size is 6.6mm x 26.26mm (w x h). The bar code is painted without padding at the sides.

**Note:** The ean-supplemental-2 and **gs1-supplemental-2** on page 939 bar codes are synonymous. The gs1-supplemental-2 synonym can be used for the ean-supplemental-2 bar code.

ean-supplemental-5
Details on the ean-supplemental-5 bar code type.

The code represents a 5 digit decimal number that can be used in conjunction with the ean 13 bar code type. Typically the code is placed to the right of the ean 13 bar code. The gap between the rightmost bar of the ean 13 code and the leftmost bar of the supplemental code should not be less than 2.31mm and should not exceed 3.3mm.

The nominal size is 15.5mm x 26.26mm (w x h). The bar code is painted without padding at the sides.

**Note:** The ean-supplemental-5 and **gs1-supplemental-5** on page 939 bar codes are synonymous. The gs1-supplemental-5 synonym can be used for the ean-supplemental-5 bar code.
gs1-8
A synonym of the ean-8 bar code type.
See ean-8 on page 937.

gs1-13
A synonym of the ean-13 bar code type.
See ean-13 on page 938.

gs1-code-128
A synonym for the ean-code-128 bar code type.
See ean-code-128 on page 938.

gs1-data-matrix
A synonym for the ean-data-matrix bar code type.
See ean-data-matrix on page 938.

gs1-supplemental-2
A synonym of the ean-supplemental-2 bar code type.
See ean-supplemental-2 on page 938.

gs1-supplemental-5
A synonym of the ean-supplemental-5 bar code type.
See ean-supplemental-5 on page 938.

intelligent-mail
Details on the Intelligent Mail bar code type.

Intelligent Mail is a United States postal service bar code for usage with the USPS mail stream. It is also known as the USPS OneCode Solution or USPS 4-State Customer Barcode. Valid abbreviations for this bar code type include 4CB, 4-CB, and USPS4CB.

The bar code encodes up to 31 digits. The code value is expected as two comma-separated fields. Each field consists solely of digits. The first field contains the tracking code. The second field contains the routing code.

The encoding is illustrated in the following table. There can be a total of 31 digits maximum.

Table 231: Encoding for the Intelligent Mail bar code

<table>
<thead>
<tr>
<th>Type</th>
<th>Field</th>
<th>Digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracking code field</td>
<td>Barcode Identifier</td>
<td>2 digits; 2nd digit must be 0-4.</td>
</tr>
<tr>
<td>Tracking code field</td>
<td>Service Type Identifier</td>
<td>3 digits.</td>
</tr>
<tr>
<td>Tracking code field</td>
<td>Mailer Identifier</td>
<td>6 or 9 digits.</td>
</tr>
<tr>
<td>Tracking code field</td>
<td>Serial Number</td>
<td>9 digits, when used with a 6-digit Mailer Identifier. 6 digits, when used with a 9-digit Mailer Identifier.</td>
</tr>
<tr>
<td>Routing code field</td>
<td>Delivery Point ZIP Code</td>
<td>0, 5, 9, or 11 digits.</td>
</tr>
</tbody>
</table>

pdf-417
Details on the pdf-417 bar code type.

Pdf-417 can be used to encode text and binary data of variable length. It is possible to encode up to 925 code words (minimal error correction degree). Text, binary data and numeric data are compressed into code words so that the limits for these types are:

- 1850 ASCII characters
- 1108 Byte values
• 2710 numeric values

The limits for mixed text are lower, as control characters are inserted to switch between the different compression modes.

Setting the error correction degree to the recommended minimum lowers the maximum number of encodable code words to 863 yielding these limits for the encodable types:

• 1726 ASCII characters
• 1033 Byte values
• 2528 numeric values

The conversion of the data specified in the codeValue attribute to the internal representation is done by an algorithm that minimizes space (Sometimes there is more than one option to encode a particular piece of data). Non-ASCII characters are encoded using the specified encoding table. Byte values that cannot be represented in an XML document (for example all characters lower 0x20 except 0x9, 0xa and 0xd) can be represented by a backslash (\) character followed by a 3 digit octal literal. The backslash character itself can be escaped by a sequence of two backslash characters.

These attributes (properties) are used by the pdf-417 bar code type:

• dataSymbolsPerLine (Data Symbols per Line) on page 904:
  • Type: Integer value.
  • The value must be an integer between 1 and 30. Low values cause more narrow printout with more lines. The number of lines is not allowed to exceed 90. It should be noted, that the overall required image space usually grows with lower values because there is a constant amount of organizational information which is added with each additional line. This is not generally the case, since lines have to be filled with padding so that specially with small amounts of data a larger value may actually create a larger image. If the value is not specified, the system computes a value that minimizes image space.
  • Fails if: Value cannot be parsed as a integer value. Value is not in the range 1...30.
  • Default value: A value that minimizes the overall image size.

• rawCodeValue (Raw Code Value) on page 906:
  • Type: A comma-separated list of integers in the range 0...899.
  • This attribute can be used instead of codeValue to specify the code value at a lower level giving more control on the encoded data.
  • Fails if: Encoding for non-ASCII characters in the code value.
  • Default value: not set

• errorCorrectionDegree (Error Correction Degree) on page 905:
  • Type: Integer value.
  • The errorCorrectionDegree property specifies the error correction degree.
  Valid values are in the range 0...8. Higher values make the image more robust.
  • Fails if: Value cannot be parsed as a integer value. Value is not in the range 0...8.
  • Default value: A value that proportional to the data size.

• encoding (Encoding) on page 904:
  • Type: Encoding
  • The encoding property sets the encoding for non-ASCII characters in the code value.

  Run "java CharsetInfo" for a list of character set encodings available on a particular platform. Valid example values are 'ISO-8859-15' or 'IBM437'.
  • Fails if:
    • Value is not a valid host name
    • Socket connection cannot be established
  • Default value: not set (the lower 8 bits of the unicode values are encoded)
**qr-code**

QR code (abbreviated from Quick Response Code) is a type of matrix bar code consisting of black modules (square dots) arranged in a square grid on a white background.

The QR code can be used to encode text and binary data of variable length. It is possible to encode up to 2953 code words. The code value is converted from the XML unicode value to the specified encoding.

By default, the bytes encoded in a QR code image are interpreted as characters from the ISO-8859-1 (Latin 1) encoding. Other encodings can be specified in the encoding attribute and an extended channel interpretations (ECI) code will be inserted if available for the encoding. The code covers issues such as:

- Is the data compressed? If yes, how?
- Is the data part of a larger message? If yes, which part?
- Is the data encoded in a non standard way? If yes, which encoding was used?

An ECI code exists for the following encodings:

- Cp437 (0,2)
- ISO-8859-1 (ECI codes 1,3)
- ISO-8859-2 (ECI code 4)
- ISO-8859-3 (ECI code 5)
- ISO-8859-4 (ECI code 6)
- ISO-8859-5 (ECI code 7)
- ISO-8859-6 (ECI code 8)
- ISO-8859-7 (ECI code 9)
- ISO-8859-8 (ECI code 10)
- ISO-8859-9 (ECI code 11)
- ISO-8859-10 (ECI code 12)
- ISO-8859-11 (ECI code 13)
- ISO-8859-13 (ECI code 15)
- ISO-8859-14 (ECI code 16)
- ISO-8859-15 (ECI code 17)
- ISO-8859-16 (ECI code 18)
- Shift_JIS (ECI code 20)
- Cp1250, windows-1250(ECI code 21)
- Cp1251, windows-1251(ECI code 22)
- Cp1252, windows-1252(ECI code 23)
- Cp1256, windows-1256(ECI code 24)
- UnicodeBigUnmarked, UTF-16BE, UnicodeBig (ECI code 25)
- UTF-8 (ECI code 26)
- US-ASCII (ECI codes 27,170)
- Big5 (ECI code 28)
- GB18030, GB2312, EUC_CN, GBK (ECI code 29)
- EUC-KR (ECI code 30)

The following attributes are used by the qr-code bar code type:

- **errorCorrectionDegree**
  - Type: Integer value.
  - The errorCorrectionDegree property specifies the error correction degree.
  - Valid values are in the range of 0 - 3. Higher values make the image more robust (ISO 18004:2006, 6.5.1 defines: 0=~7%, 1=−15%, 2=−25% and 3=−30%).
  - Fails if: Value cannot be parsed as an integer value. Value is not in the range of 0 - 3.
  - Default value: 3

- **encoding**
• Type: Encoding
• The encoding property sets the encoding for non-ASCII characters in the code value.
• The following encodings can be set:

<table>
<thead>
<tr>
<th>Encoding Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO-8859-1</td>
<td>Use this setting if all characters in codeValue are from this code page. Some scanners or scanner apps interpret the non-ASCII characters non standard (e.g. as Japanese characters). In this case the scanner may have a setting to change the interpretation. If this is not the case then try using &quot;UTF-8&quot; encoding. Please note that for ISO-8859-1 encoding no ECI code is embedded (Use the encoding ISO-8859-15 to force an ECI inclusion.)</td>
</tr>
<tr>
<td>Bytes</td>
<td>Use this setting to set the byte values. Literal characters in codeValue are mapped to ISO-8859-1 byte representation and characters not representable in XML documents can be escaped by a backslash () character followed by a 3 digit octal literal. The backslash character itself can be escaped by a sequence of two backslash characters or its octal representation \134 in ISO-8859-1.</td>
</tr>
<tr>
<td>UTF-8</td>
<td>This is the default value and should be used unless the two other options from above are applicable. If a scanner fails to interpret the characters correctly and all character in codeValue are available from a different encoding listed by CharsetInfo, then this encoding should be tried next.</td>
</tr>
<tr>
<td>Any encoding listed by CharsetInfo (e.g. &quot;Shift_JIS&quot;, &quot;Big5&quot; or &quot;ISO-8859-8&quot;)</td>
<td>If all of the option above are not applicable this option should be used. For encodings for which a ECI code exists, the code will be embedded allowing the scanner to change the interpretation accordingly.</td>
</tr>
<tr>
<td>Any encoding listed by CharsetInfo prefixes with the string &quot;RAW.&quot; (e.g. &quot;RAW-Shift_JIS&quot;, &quot;RAW-Big5&quot; or &quot;RAW-UTF-8&quot;)</td>
<td>Some scanners do not recognize ECI codes and expect a specific encoding (other than ISO-8859-1). For this case, this setting should be used.</td>
</tr>
</tbody>
</table>

• Fails if: Value is not a valid encoding name.
• Default value: UTF-8

QR code examples

![QR code example](image)

Figure 413: Hello World: size not specified (default error correction (3))
upc-a
Details on the upc-a bar code type.

The code represents a 12 digit decimal number. First digit is the 'number system character' code, the next five identify the manufacturer and the following five identify an article. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with such a 11 digit value then the 12th digit is automatically calculated by the system.
The nominal size is 1.469in x 1.020in (w x h). The bar code area has the measurements 1.235in x 0.965in. The left padding measures 0.117in.

**upc-e**
Details on the upc-e bar code type.

The code represents a 8 digit decimal number. Upc-e is a compressed version of upc-a. By the method of zero suppression some upc-a codes are available as upc-e codes. This translation table shows which upc-a numbers can be transformed to short versions:

### Table 232: upc-a numbers that can be transformed to short versions

<table>
<thead>
<tr>
<th>UPC-E Number</th>
<th>Digits to insert</th>
<th>Position of Insertion</th>
<th>Resulting UPC-A Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>sNNNNN0c</td>
<td>00000</td>
<td>3</td>
<td>sNN000000NNNc</td>
</tr>
<tr>
<td>sNNNNN1c</td>
<td>10000</td>
<td>3</td>
<td>sNN100000NNNc</td>
</tr>
<tr>
<td>sNNNNN2c</td>
<td>20000</td>
<td>3</td>
<td>sNN200000NNNc</td>
</tr>
<tr>
<td>sNNNNN3c</td>
<td>00000</td>
<td>4</td>
<td>sNNN000000NNNc</td>
</tr>
<tr>
<td>sNNNNN4c</td>
<td>00000</td>
<td>5</td>
<td>sNNNN000000Nc</td>
</tr>
<tr>
<td>sNNNNN5c</td>
<td>00005</td>
<td>6</td>
<td>sNNNNNN00005c</td>
</tr>
<tr>
<td>sNNNNN6c</td>
<td>00006</td>
<td>6</td>
<td>sNNNNNN00006c</td>
</tr>
<tr>
<td>sNNNNN7c</td>
<td>00007</td>
<td>6</td>
<td>sNNNNNN00007c</td>
</tr>
<tr>
<td>sNNNNN8c</td>
<td>00008</td>
<td>6</td>
<td>sNNNNNN00008c</td>
</tr>
<tr>
<td>sNNNNN9c</td>
<td>00009</td>
<td>6</td>
<td>sNNNNNN00009c</td>
</tr>
</tbody>
</table>

The rightmost digit (checksum digit) can be omitted. When the system is supplied with such a 7 digit value then the 8th digit is automatically calculated by the system.

The nominal size is 0.897in x 1.020in (w x h). The bar code area has the measurements 0.663in x 0.965in. The left padding measures 0.117in.

**upc-supplemental-2**
Details on the upc-supplemental-2 bar code type.

The code represents a 2 digit decimal number that can be used in conjunction with the upc-a bar code type. Typically the code is placed to the right of the upc-a bar code. The gap between the rightmost bar of the upc-a code and the leftmost bar of the supplemental code should not be less than 2.31mm and should not exceed 3.3mm.

The nominal size is 0.26in x 1.02in (w x h). The bar code is painted without padding at the sides.

**upc-supplemental-5**
Details on the bar code type.

The code represents a 5 digit decimal number that can be used in conjunction with the upc-a bar code type. Typically the code is placed to the right of the ean upc-a code. The gap between the rightmost bar of the upc-a code and the leftmost bar of the supplemental code should not be less than 2.31mm and should not exceed 3.3mm.

The nominal size is 0.611in x 1.02in (w x h). The bar code is painted without padding at the sides.

This example shows how the supplemental code can be used in conjunction with a upc-a code:

```xml
<LAYOUTNODE orientation="horizontal" width="min" length="min">
  <BARCODEBOX codeType="upc a" codeValue="01234567891" fontSize="10" mirrored="true"/>
  <BARCODEBOX codeType="upc supplemental 2" codeValue="47" fontSize="10" mirrored="true"/>
</LAYOUTNODE>
```
RTL Class Reference
Reference information for Report Template Language (RTL) classes.

- The Boolean Class on page 946
- The Color Class on page 946
- The Date Class on page 951
- The Enum Classes on page 952
- The FGLNumericVariable Class on page 960
- The FGLStringVariable Class on page 960
- The Numeric Class on page 961
- The Runtime Class on page 972
- The String Class on page 975

RTL Classes Overview
The Report Template Language (RTL) contains object classes for each type of report item property.

The type of each property is indicated in the Properties page. The methods of these classes may be used in RTL Expressions to define a property value.

Table 233: RTL Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Contains methods used for all logical operations.</td>
</tr>
<tr>
<td>Color</td>
<td>Contains methods and static member variables related to color.</td>
</tr>
<tr>
<td>Enum</td>
<td>A set of classes, consisting of a class for each property of this type; each class contains static member variables that provide a list of valid values for the corresponding property.</td>
</tr>
<tr>
<td>Date</td>
<td>Provides methods for date formatting and parsing.</td>
</tr>
<tr>
<td>FGLNumericVariable</td>
<td>For every 4GL numeric variable of report data, an object is created that is an instance of an FGLNumericVariable. These objects hold the value of the 4GL variable.</td>
</tr>
<tr>
<td>FGLStringVariable</td>
<td>For every CHAR, VARCHAR, STRING, TEXT, DATE, DATETIME and INTERVAL 4GL variable of report data, an object is created that is an instance of an FGLStringVariable. These objects hold the value of the 4GL variable.</td>
</tr>
<tr>
<td>Numeric</td>
<td>Contains methods used for all numeric operations. The class has the precision of a double and the arithmetic operators are defined for objects of this type.</td>
</tr>
<tr>
<td>Runtime</td>
<td>Provides functions that simplify the creation of dynamic designs that change behavior based on the runtime setup.</td>
</tr>
<tr>
<td>String</td>
<td>Contains methods used for all string operations.</td>
</tr>
</tbody>
</table>

Related concepts
Using RTL classes on page 852
The Report Template Language (RTL) expressions are typed; that is, the expression must return a particular class.

Expressions in properties on page 847
A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

**The Boolean Class**
The Boolean class provides the ability to represent logical TRUE/FALSE values.

- Syntax
- Member Objects
- Usage

**Syntax**

Boolean

**Member Objects**

**Table 234: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Contains the value TRUE.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Contains the value FALSE.</td>
</tr>
</tbody>
</table>

**Usage**

With RTL classes, it is not possible to create and subclass objects. The new keyword is not supported.

**Member Objects Usage**

All expressions in the RTL expression language that contain relational operators return one of these objects, which are static member variables that do not require an object instance.

To specify TRUE or FALSE in a formula, the variables are prefixed with the 'Boolean' class name and the '.' character.

**Example**

Boolean.TRUE

**Related concepts**

Expressions in properties on page 847

A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

**The Color Class**
The Color class provides methods for specifying the color of an object.

- Syntax
- Member Objects
- Class Methods
- Object Methods

**Syntax**

Color
**Member Objects: Static Member Variables**

These member variables do not require an object. They are prefixed by the Color class name.

**Table 235: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLACK</td>
<td>Specifies a Color object having the color BLACK</td>
</tr>
<tr>
<td>BLUE</td>
<td>Specifies a Color object having the color BLUE</td>
</tr>
<tr>
<td>CYAN</td>
<td>Specifies a Color object having the color CYAN</td>
</tr>
<tr>
<td>DARK_GRAY</td>
<td>Specifies a Color object having the color DARK_GRAY</td>
</tr>
<tr>
<td>GRAY</td>
<td>Specifies a Color object having the color GRAY</td>
</tr>
<tr>
<td>GREEN</td>
<td>Specifies a Color object having the color GREEN</td>
</tr>
<tr>
<td>LIGHT_GRAY</td>
<td>Specifies a Color object having the color LIGHT_GRAY</td>
</tr>
<tr>
<td>MAGENTA</td>
<td>Specifies a Color object having the color MAGENTA</td>
</tr>
<tr>
<td>ORANGE</td>
<td>Specifies a Color object having the color ORANGE</td>
</tr>
<tr>
<td>PINK</td>
<td>Specifies a Color object having the color PINK</td>
</tr>
<tr>
<td>RED</td>
<td>Specifies a Color object having the color RED</td>
</tr>
<tr>
<td>WHITE</td>
<td>Specifies a Color object having the color WHITE</td>
</tr>
<tr>
<td>YELLOW</td>
<td>Specifies a Color object having the color YELLOW</td>
</tr>
</tbody>
</table>

**Usage**

With RTL classes, it is not possible to create and subclass objects. The `new` keyword is not supported.

The default color for a report item, such as a LayoutNode, is black. You can change the color of the item by entering a value for the `color` or `bgColor` property in the Properties View. These properties are of type Color; Instead of selecting a color from the Edit Expressions color palette, you can use the members of this class in an expression that returns a Color object.

**Member Objects Usage**

These objects are static member variables that do not require an object instance. The variables are prefixed with the Color class name and the `.` character.

Example:

```java
Color.RED
```

**Class Methods**

Class methods do not require a Color object instance. When you invoke a class method, it is prefixed with the Color class name and the `.` character.
### Table 236: Class Methods (Static Member Methods)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fromHSBA(h Numeric, s Numeric, b Numeric)</code></td>
<td>Returns a Color object based on hue, saturation, and brightness for the HSB color model.</td>
</tr>
<tr>
<td><code>fromHSBA(h Numeric, s Numeric, b Numeric, a Numeric)</code></td>
<td>Returns a Color object based on hue, saturation, brightness, and alpha for the HSB color model.</td>
</tr>
<tr>
<td><code>fromRGBA(r Numeric, g Numeric, b Numeric)</code></td>
<td>Returns an opaque sRGB Color object with the specified red, green, and blue values.</td>
</tr>
<tr>
<td><code>fromRGBA(r Numeric, g Numeric, b Numeric, a Numeric)</code></td>
<td>Returns an sRGB Color object with the specified red, green, blue, and alpha values.</td>
</tr>
</tbody>
</table>

### Object Methods

Color object methods are called on a Color object instance. Object methods require an object reference. When you invoke the method, it is prefixed with the object instance name and the "." character.
### Table 237: Object Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>brighter()</td>
<td>Brightens the Color object. Creates a brighter version of the RGB components. Invoking a series of invocations of the brighter and darker methods might give an inconsistent result because of rounding errors.</td>
</tr>
<tr>
<td>darker()</td>
<td>Darkens the Color object. Creates a darker version of the RGB components. Invoking a series of invocations of the darker and brighter methods might give an inconsistent result because of rounding errors.</td>
</tr>
<tr>
<td>getAlpha()</td>
<td>Returns the Numeric alpha component in the range 0-255.</td>
</tr>
<tr>
<td>getBrightness()</td>
<td>Returns the Numeric brightness value of the value in the HSB color model.</td>
</tr>
<tr>
<td>getBlue()</td>
<td>Returns the Numeric blue component in the range 0-255 in the default sRGB space.</td>
</tr>
<tr>
<td>getGreen()</td>
<td>Returns the Numeric green component in the range 0-255 in the default sRGB space.</td>
</tr>
<tr>
<td>getRGBA()</td>
<td>Returns the Numeric RGB value representing the color in the default sRGB color model. Bits 24-31 are alpha, 16-23 are red, 8-15 are green, and 0-7 are blue.</td>
</tr>
<tr>
<td>getHue()</td>
<td>Returns the hue value of the value in the HSB color model.</td>
</tr>
<tr>
<td>getRed()</td>
<td>Returns the Numeric red component in the range 0-255 in the default sRGB space.</td>
</tr>
<tr>
<td>getSaturation()</td>
<td>Returns the saturation value of the value in the HSB color model.</td>
</tr>
<tr>
<td>toString()</td>
<td>Return a string representation in the form “#argb”.</td>
</tr>
</tbody>
</table>

#### Example - Color Object Method

```java
Color.RED.darker()
```

#### Related concepts

*Expressions in properties* on page 847
A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

**fromHSBA()**
Color class method that returns a Color object based on specified values for the HSB color model.

**Syntax**
To return a Color object based on hue, saturation, and brightness:

```
Color.fromHSBA(h Numeric, s Numeric, b Numeric)
```

To return a Color object based on hue, saturation, brightness, and alpha:

```
Color.fromHSBA(h Numeric, s Numeric, b Numeric, a Numeric)
```

**Parameters**
- h (hue) is any floating-point number. The floor of this number is subtracted from it to create a fraction between 0 and 1, which is then multiplied by 360.
- s (saturation) is a floating-point value between zero and one (numbers within the range 0.0-1.0).
- b (brightness) is a floating-point value between zero and one (numbers within the range 0.0-1.0).
- a (alpha) is the alpha component, which specifies opacity.

**Usage**
Class methods do not require a Color object instance. When you invoke a class method, it is prefixed with the Color class name and the '.' character.

**Example**
```
Color.fromHSBA(0.5,0.5,0.5,0.5)
```

This returns a Color object that is red and partially transparent.

**Related concepts**
- **fromRGBA()** on page 950
  Color class method that returns an sRGB Color object with the specified values.

- **fromRGBA()**
  Color class method that returns an sRGB Color object with the specified values.

**Syntax**
To return an opaque sRGB Color object based on red, green, and blue values:

```
fromRGBA(r Numeric, g Numeric, b Numeric)
```

To return an sRGB Color object based on red, green, blue, and alpha values:

```
fromRGBA(r Numeric, g Numeric, b Numeric, a Numeric)
```

**Parameters**
- r (red) is within the range (0.0 - 255)
- g (green) is within the range (0.0 - 255)
- b (blue) is within the range (0.0 - 255)
• \( a \) (alpha) is within the range (0.0 - 1.0)

Alpha defaults to 1.0 if not specified.

The color used depends on the best match from the colors available for the output device.

Usage

Class methods do not require a Color object instance. When you invoke a class method, it is prefixed with the Color class name and the '.' character.

Example

```
Color.fromRGBA(200,162,200)
```

This returns a Color object that is lilac and opaque.

Related concepts

defFromHSBA() on page 950

Color class method that returns a Color object based on specified values for the HSB color model.

The Date Class

The Date class provides methods for date formatting and parsing.

• Syntax
• Methods
• Usage
• Formatting Symbols for Dates

Syntax

```
Date
```

Methods

Class methods are static methods that do not require an object. The method name is prefixed by the class name.

Table 238: Class Methods (Static Member Methods)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fromIsoValue(String value)</td>
<td>Constructs a Date from the date value; returns a Date instance representing the value. The value is expected in iso format (&quot;YYYY-MM-DD&quot;). Using the function guarantees the Date value is constructed correctly without the danger of runtime parse errors due to changing 4GL dates.</td>
</tr>
<tr>
<td>parseString(String value, String format)</td>
<td>Constructs a Date from value. The parsing is based on the passed format pattern. See Formatting Symbols for Dates.</td>
</tr>
<tr>
<td>format(java.lang.String.format)</td>
<td>Formats a Date as a String according to a format specification. The format specification is 4GL-compatible. Returns a string representation of the date. See Formatting Symbols for Dates.</td>
</tr>
<tr>
<td>today()</td>
<td>Constructs a Date from the current date value. Returns a Date instance representing the current date.</td>
</tr>
</tbody>
</table>

Usage

With RTL classes, it is not possible to create and subclass objects. The new keyword is not supported.
These static methods do not require a Date object instance. When you invoke the method, it is prefixed with the `Date` class name and the `.`

**Formatting symbols**

The formatting symbol can use either uppercase or lowercase letters.

**Table 239: Formatting Symbols for Dates**

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dd</td>
<td>Day of the month as a two-digit integer.</td>
</tr>
<tr>
<td>ddd</td>
<td>Three-letter English-language abbreviation of the day of the week, for example, Mon, Tue.</td>
</tr>
<tr>
<td>mm</td>
<td>Month as a two-digit integer.</td>
</tr>
<tr>
<td>mmm</td>
<td>Three-letter English-language abbreviation of the month, for example, Jan, Feb.</td>
</tr>
<tr>
<td>yy</td>
<td>Year, as a two-digit integer representing the two trailing digits.</td>
</tr>
<tr>
<td>yyyy</td>
<td>Year as a four-digit number.</td>
</tr>
</tbody>
</table>

**Examples**

The following expression displays today's date in the format "dd mmm yyyy", for example, "10 Jun 2016":

```
Date.today().format("DD MMM YYYY")
```

The following expression uses the value of the variable `orderline.orders.orderdate` to create a valid date for this value in the specified format:

```
Date.fromIsoValue(orderline.orders.orderdate.isoValue).format("DDD DD MMM YYYY")
```

**Related concepts**

Expressions in properties on page 847

A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

**The Enum Classes**

The Enum classes provide the list of valid values for form item properties that are of type Enum.

- The Alignment Class on page 953
- The TextAlignement Class on page 953
- The BaselineType Class on page 953
- The LayoutDirection Class on page 954
- The Y-SizeAdjustment Class on page 954
- The PageNoFormat Class on page 955
- The TrimText Class on page 955
- The X-SizeAdjustment Class on page 955
- FloatingBehavior Class on page 956
- Section Class on page 956
- XYChartDrawAs Class on page 956
- MapChartDrawAs Class on page 957
- CategoryChartDrawAs Class on page 957
- CodeType Class on page 958
- BorderStyles Classes on page 959
**The Alignment Class**
Public static member variables that represent the valid values for the Alignment property.

**Syntax**

```
Alignment
```

This class consists of a set of public static member variables that represent the valid values for the alignment property.

**Table 240: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
</tr>
<tr>
<td>Center</td>
</tr>
<tr>
<td>Far</td>
</tr>
<tr>
<td>Near</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

**The TextAlignment Class**
Public static member variables that represent the valid values for the textAlignment property.

**Syntax**

```
TextAlignment
```

The class consists of a set of public static member variables that represent the valid values for the textAlignment property.

**Table 241: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center</td>
</tr>
<tr>
<td>Left</td>
</tr>
<tr>
<td>Right</td>
</tr>
</tbody>
</table>

**The BaselineType Class**

Public static member variables that represent the valid values for the baselineType property.

**Syntax**

```
BaselineType
```

The class consists of a set of public static member variables that represent the valid values for the baselineType property.

**Table 242: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leftleft</td>
</tr>
<tr>
<td>Leftright</td>
</tr>
</tbody>
</table>
The **LayoutDirection Class**

Public static member variables that represent the valid values for the layoutDirection property.

**Syntax**

```c
LayoutDirection
```

The class consists of a set of public static member variables that represent the valid values for the layoutDirection property.

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rightleft</td>
</tr>
<tr>
<td>RightRight</td>
</tr>
</tbody>
</table>

---

**Table 243: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BottomToTop</td>
</tr>
<tr>
<td>HorizontalNatural</td>
</tr>
<tr>
<td>LeftToRight</td>
</tr>
<tr>
<td>RightToLeft</td>
</tr>
<tr>
<td>Swapped</td>
</tr>
<tr>
<td>TopToBottom</td>
</tr>
<tr>
<td>TurnLeft</td>
</tr>
<tr>
<td>TurnRight</td>
</tr>
<tr>
<td>Unturned</td>
</tr>
<tr>
<td>UpsideDown</td>
</tr>
<tr>
<td>VerticalNatural</td>
</tr>
<tr>
<td>VerticalUnnatural</td>
</tr>
</tbody>
</table>

---

The **Y-SizeAdjustment Class**

Public static member variables that represent the valid values for the Y-SizeAdjustment property.

**Syntax**

```c
Y-SizeAdjustment
```

The class consists of a set of public static member variables that represent valid values for the Y-SizeAdjustment property.

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExpandToParent</td>
</tr>
<tr>
<td>ShrinkToChildren</td>
</tr>
</tbody>
</table>
The **PageNoFormat Class**
Public static member variables that represent the valid values for the `pageNoFormat` property.

**Syntax**

```c
PageNoFormat
```

The class consists of a set of public static member variables that represent the valid values for the `pageNoFormat` property.

**Table 245: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
</tr>
<tr>
<td>Lowerroman</td>
</tr>
<tr>
<td>Upperroman</td>
</tr>
</tbody>
</table>

The **TrimText Class**
Public static member variables that represent the valid values for the `trimText` property.

**Syntax**

```c
TrimText
```

The class consists of a set of public static member variables that represent the valid values for the `trimText` property.

**Table 246: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both</td>
</tr>
<tr>
<td>Compress</td>
</tr>
<tr>
<td>Left</td>
</tr>
<tr>
<td>Right</td>
</tr>
</tbody>
</table>

The **X-SizeAdjustment Class**
Public static member variables that represent the valid values for the `X-SizeAdjustment` property.

**Syntax**

```c
X-SizeAdjustment
```

The class consists of a set of public static member variables that represent valid values for the `X-SizeAdjustment` property.

**Table 247: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExpandtoParent</td>
</tr>
<tr>
<td>ShrinktoChildren</td>
</tr>
</tbody>
</table>
**FloatingBehavior Class**
Public static member variables that represent the valid values for the floatingBehavior property.

**Syntax**

```
FloatingBehavior
```

The class consists of a set of public static member variables that represent the valid values for the property.

**Table 248: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encloses</td>
</tr>
<tr>
<td>Free</td>
</tr>
</tbody>
</table>

**Section Class**
Public static member variables that represent the valid values for the Section property.

**Syntax**

```
Section
```

The class consists of a set of public static member variables that represent the valid values for the property.

**Table 249: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnyPageFooter</td>
</tr>
<tr>
<td>AnyPageHeader</td>
</tr>
<tr>
<td>EvenPageHeader</td>
</tr>
<tr>
<td>EvenPageFooter</td>
</tr>
<tr>
<td>FirstPageHeader</td>
</tr>
<tr>
<td>FirstPageFooter</td>
</tr>
<tr>
<td>LastPageFooter</td>
</tr>
<tr>
<td>OddPageFooter</td>
</tr>
<tr>
<td>OddPageHeader</td>
</tr>
<tr>
<td>ItemSeparator</td>
</tr>
</tbody>
</table>

**XYChartDrawAs Class**
Public static member variables that represent the valid values for the XY chart DrawAs property.

**Syntax**

```
XYChartDrawAs
```

The class consists of a set of public static member variables that represent the valid values for the property.
### Table 250: Member Objects (Static Member Variables)

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
</tr>
<tr>
<td>Line</td>
</tr>
<tr>
<td>Polar</td>
</tr>
<tr>
<td>Scatter</td>
</tr>
<tr>
<td>StackedArea</td>
</tr>
<tr>
<td>Step</td>
</tr>
<tr>
<td>StepArea</td>
</tr>
<tr>
<td>TimeSeries</td>
</tr>
</tbody>
</table>

*MapChartDrawAs Class*

Public static member variables that represent the valid values for the map chart DrawAs property.

**Syntax**

```csharp
MapChartDrawAs
```

The class consists of a set of public static member variables that represent the valid values for the property.

### Table 251: Member Objects (Static Member Variables)

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bar</td>
</tr>
<tr>
<td>Bar3D</td>
</tr>
<tr>
<td>Pie</td>
</tr>
<tr>
<td>Pie3D</td>
</tr>
<tr>
<td>Ring</td>
</tr>
</tbody>
</table>

*CategoryChartDrawAs Class*

Public static member variables that represent the valid values for the category chart DrawAs property.

**Syntax**

```csharp
CategoryChartDrawAs
```

The class consists of a set of public static member variables that represent the valid values for the property.

### Table 252: Member Objects (Static Member Variables)

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
</tr>
<tr>
<td>Bar</td>
</tr>
<tr>
<td>Bar3D</td>
</tr>
<tr>
<td>Line</td>
</tr>
</tbody>
</table>
Name
Line3D
SpiderWeb
StackedArea
StackedBar
Waterfall

**CodeType Class**
Public static member variables that represent the valid values for the codeType property.

**Syntax**

```
CodeType
```

The class consists of a set of public static member variables that represent the valid values for the property.

**Table 253: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upc_a</td>
</tr>
<tr>
<td>Upc_e</td>
</tr>
<tr>
<td>Upc_supplemental_2</td>
</tr>
<tr>
<td>Upc_supplemental_5</td>
</tr>
<tr>
<td>Gs1_13</td>
</tr>
<tr>
<td>Ean_13</td>
</tr>
<tr>
<td>Gs1-8</td>
</tr>
<tr>
<td>Ean_8</td>
</tr>
<tr>
<td>Gs1_supplemental_2</td>
</tr>
<tr>
<td>Ean_supplemental_2</td>
</tr>
<tr>
<td>Gs1_supplemental_5</td>
</tr>
<tr>
<td>Ean_supplemental_5</td>
</tr>
<tr>
<td>Code_128</td>
</tr>
<tr>
<td>Gs1_code_128</td>
</tr>
<tr>
<td>Ean_Code_128</td>
</tr>
<tr>
<td>Code_2_5_industrial</td>
</tr>
<tr>
<td>Code_2_5_inverted</td>
</tr>
<tr>
<td>Code_2_5_IATA</td>
</tr>
<tr>
<td>Code_2_5_interleaved</td>
</tr>
<tr>
<td>Code_2_5_matrix</td>
</tr>
<tr>
<td>Code_2_5_datalogic</td>
</tr>
<tr>
<td>Code_BCD_matrix</td>
</tr>
</tbody>
</table>
### Public member variables that represent the valid values for the various border style properties.

#### Syntax

The attributes `borderStyle`, `borderTopStyle`, `borderRightStyle`, `borderBottomStyle` and `borderLeftStyle` have a set of public member variables that represent the valid values for the property:

#### Table 254: Member Objects (Static Member Variables)

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashed</td>
</tr>
<tr>
<td>Dotted</td>
</tr>
<tr>
<td>Double</td>
</tr>
<tr>
<td>Groove</td>
</tr>
<tr>
<td>Inset</td>
</tr>
<tr>
<td>Outset</td>
</tr>
<tr>
<td>Ridge</td>
</tr>
<tr>
<td>solid</td>
</tr>
</tbody>
</table>

#### Related concepts

- Border style properties on page 909
The border style properties control the style of the borders for the report object.

**The FGLNumericVariable Class**

For every Genero numeric variable (INTEGER, SMALLINT, FLOAT, SMALLFLOAT, DECIMAL and MONEY) of report data, an object is created that is an instance of an FGLNumericVariable. These objects hold the value of the 4GL variable.

- Syntax on page 960
- Member Objects on page 960
- Class Methods on page 960
- Class Usage on page 960

**Syntax**

```
FGLNumericVariable
```

**Member Objects**

Defined fields for the FGLNumericVariable class.

**Table 255: Member Objects (Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A Numeric containing the value of the field; this is the same as the value of the 4GL variable. See Usage.</td>
</tr>
<tr>
<td>fglValue</td>
<td>A String containing the value of the field as formatted by the DVM. See Usage.</td>
</tr>
<tr>
<td>name</td>
<td>A String specifying the name of the field.</td>
</tr>
<tr>
<td>caption</td>
<td>A String specifying the title of the field.</td>
</tr>
<tr>
<td>type</td>
<td>A String specifying the type of the field.</td>
</tr>
</tbody>
</table>

**Class Methods**

This subclass inherits the public methods of the Numeric class.

**Class Usage**

In addition to the value of the 4GL variable, these objects contain member variables, among which is the variable `value` also containing the numeric value. For this reason, it is legal to write "order_line.itemprice" in your expression as a shortcut for "order_line.itemprice.value".

The member variable `fglValue` contains a String representing the numeric value as formatted by the DVM, taking into consideration such parameters as USING, DBMONEY, etc. In contrast, the member variable `value` is a numeric value without formatting.

**The FGLStringVariable Class**

For every CHAR, VARCHAR, STRING, TEXT, DATE, DATETIME and INTERVAL Genero variable of report data, an object is created that is an instance of an FGLStringVariable. These objects hold the value of the Genero variable.

- Syntax
- Member Objects on page 961
- Class Methods on page 961
- Class Usage on page 961
Syntax

FGLStringVariable

**Member Objects**
Defined fields for the FGLStringVariable class.

**Table 256: Member Objects (Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A String containing the value of the field; this is the same as the value of the Genero variable. See <a href="#">Usage</a>.</td>
</tr>
<tr>
<td>name</td>
<td>A String specifying the name of the field.</td>
</tr>
<tr>
<td>caption</td>
<td>A String specifying the title of the field.</td>
</tr>
<tr>
<td>type</td>
<td>A String specifying the type of the field.</td>
</tr>
</tbody>
</table>

**Class Methods**
This subclass inherits the public methods of the String class.

**Class Usage**
In addition to the value of the Genero variable, these objects contain a member variable "value" which also contains the value. For this reason, it is legal to write "order_line.billState" in your expression as a shortcut for "order_line.billState.value".  

**The Numeric Class**
The Numeric class provides methods for working with numbers.

Values for this data type are limited to 15 significant digits.

- Syntax
- Methods
- Usage
- Examples

**Syntax**

Numeric
## Methods

### Table 257: Object Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs()</td>
<td>Returns the absolute value.</td>
</tr>
<tr>
<td>atan2(Numeric x)</td>
<td>Returns the angle ( \theta ) from the conversion of rectangular coordinates ((x, y)) to polar coordinates ((r, \theta)).</td>
</tr>
<tr>
<td>byteValue()</td>
<td>Returns the value converted to a byte.</td>
</tr>
<tr>
<td>cbert()</td>
<td>Returns the cube root of a numeric value.</td>
</tr>
<tr>
<td>ceil()</td>
<td>Returns the smallest numeric value that is greater than or equal to the value and is equal to a mathematical integer.</td>
</tr>
<tr>
<td>cos()</td>
<td>Returns the trigonometric cosine of an angle.</td>
</tr>
<tr>
<td>cosh()</td>
<td>Returns the hyperbolic cosine of a numeric value.</td>
</tr>
<tr>
<td>exp()</td>
<td>Returns the base -e exponential.</td>
</tr>
<tr>
<td>floor()</td>
<td>Returns the largest numeric value that is less than or equal to the value and is equal to a mathematical integer.</td>
</tr>
<tr>
<td>format(&quot;format-string&quot;)</td>
<td>Converts the value to a string representation defined by a format string.</td>
</tr>
<tr>
<td>getExponent()</td>
<td>Returns the unbiased exponent used in the representation of a numeric value.</td>
</tr>
<tr>
<td>intValue()</td>
<td>Returns the value converted to an integer (signed 32 bit).</td>
</tr>
<tr>
<td>isNaN()</td>
<td>Returns TRUE if the value is not a number.</td>
</tr>
<tr>
<td>isInfinite()</td>
<td>Returns TRUE if the value has infinite value.</td>
</tr>
<tr>
<td>isNaN()</td>
<td>Returns TRUE if the object is tagged as null, otherwise false. This is the case for null valued input variables read from the input stream. For backward compatibility, null values do not have special behavior when used with the various operators. Specifically a numeric input variable that is null behaves in arithmetic like the 0 value.</td>
</tr>
<tr>
<td>isNull()</td>
<td>Returns TRUE if the object is tagged as null, otherwise false. This is the case for null valued input variables read from the input stream. For backward compatibility, null values do not have special behavior when used with the various operators. Specifically a numeric input variable that is null behaves in arithmetic like the 0 value.</td>
</tr>
<tr>
<td>log()</td>
<td>Returns the natural logarithm (base e) of a numeric value.</td>
</tr>
<tr>
<td>log10()</td>
<td>Returns the base 10 logarithm of a numeric value.</td>
</tr>
<tr>
<td>sqrt()</td>
<td>Returns the correctly rounded positive square root of a double value.</td>
</tr>
<tr>
<td>sin()</td>
<td>Returns the trigonometric sine, measured in radians.</td>
</tr>
<tr>
<td>sinh()</td>
<td>Returns the hyperbolic sine of a numeric value.</td>
</tr>
<tr>
<td>tan()</td>
<td>Returns the tangent of an angle measured in radians.</td>
</tr>
<tr>
<td>tanh()</td>
<td>Returns the hyperbolic tangent of a numeric value.</td>
</tr>
<tr>
<td>toBoolean()</td>
<td>Returns the Boolean false when the value is 0. Returns true for any other value.</td>
</tr>
<tr>
<td>toChar()</td>
<td>Returns the unicode character representation of a numeric value. For example, <code>65.toChar()</code> yields &quot;A&quot;.</td>
</tr>
<tr>
<td>toColor()</td>
<td>Returns a color object. The value is interpreted as a RGB integer.</td>
</tr>
<tr>
<td>toDegrees()</td>
<td>Converts the value from radians to degrees.</td>
</tr>
<tr>
<td>toRadians()</td>
<td>Converts the value from degrees to radians.</td>
</tr>
</tbody>
</table>
Usage

Important: This data type is limited to 15 significant digits.

With RTL classes, it is not possible to create and subclass objects. The new keyword is not supported.

Related concepts

Expressions in properties on page 847

A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

abs()

Returns the absolute value of an int value.

Syntax

```
abs()
```

Usage

- If the value is not negative, the value is returned.
- If the value is negative, the negation of the value is returned.
- If the value is equal to the value of Integer.MIN_VALUE, the most negative representable int value, the result is that same value, which is negative.

atan2()

Returns the angle theta from the conversion of rectangular coordinates \((x, y)\) to polar coordinates \((r, \theta)\).

Syntax

```
atan2(Numeric x)
```

Parameters

- \(x\) - the abscissa coordinate

Usage

This method computes the phase theta by computing an arc tangent of \(y/x\) in the range of \(-\pi\) to \(\pi\).

Special cases:

- If either value is NaN, then the result is NaN.
- If the value is positive zero and the argument is positive, or the value is positive and finite and the argument is positive infinity, then the result is positive zero.
- If the value is negative zero and the argument is positive, or the value is negative and finite and the argument is positive infinity, then the result is negative zero.
- If the value is positive zero and the argument is negative, or the value is positive and finite and the argument is negative infinity, then the result is the double value closest to \(\pi\).
- If the value is negative zero and the argument is negative, or the value is negative and finite and the argument is negative infinity, then the result is the double value closest to \(-\pi\).
- If the value is positive and the argument is positive zero or negative zero, or the value is positive infinity and the argument is finite, then the result is the double value closest to \(\pi/2\).
- If the value is negative and the argument is positive zero or negative zero, or the value is negative infinity and the argument is finite, then the result is the double value closest to \(-\pi/2\).
- If both value and argument are positive infinity, then the result is the double value closest to \(\pi/4\).
• If the value is positive infinity and the argument is negative infinity, then the result is the double value closest to $3\pi/4$.
• If the value is negative infinity and the argument is positive infinity, then the result is the double value closest to $-\pi/4$.
• If both value and argument are negative infinity, then the result is the double value closest to $-3\pi/4$.

The computed result must be within 2 ulps of the exact result. Results must be semi-monotonic.

cbrt()
Returns the cube root of a double value.

Syntax

cbrt()

Usage

For positive finite $x$, $\text{cbrt}(-x) == -\text{cbrt}(x)$; that is, the cube root of a negative value is the negative of the cube root of that value's magnitude.

Special cases:

• If the value is NaN, then the result is NaN.
• If the value is infinite, then the result is an infinity with the same sign as the value.
• If the value is zero, then the result is a zero with the same sign as the value.

The computed result must be within 1 ulp of the exact result.

Related concepts

sqrt() on page 971
Returns the correctly rounded positive square root of a double value.

cceil()
Returns the smallest (closest to negative infinity) double value that is greater than or equal to the value and is equal to a mathematical integer.

Syntax

cceil()

Usage

Special cases:

• If the value is already equal to a mathematical integer, then the result is the same as the value.
• If the value is NaN or an infinity or positive zero or negative zero, then the result is the same as the value.
• If the value is less than zero but greater than -1.0, then the result is negative zero.

Note: The value of $\text{Numeric.ceil}(x)$ is exactly the value of $-\text{Numeric.floor}(-x)$.

Related concepts

floor() on page 966
Returns the largest (closest to positive infinity) double value that is less than or equal to the value and is equal to a mathematical integer.

\[ \text{cos()} \]

Returns the trigonometric cosine of an angle.

**Syntax**

```java
\text{cos()} \n```

**Usage**

If the value is NaN or an infinity, then the result is NaN.

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

**Related concepts**

- `sin()` on page 970
  Returns the trigonometric sine of an angle.
- `tan()` on page 971
  Returns the trigonometric tangent of an angle.

\[ \text{cosh()} \]

Returns the hyperbolic cosine of a double value.

**Syntax**

```java
\text{cosh()} \n```

**Usage**

The hyperbolic cosine of \( x \) is defined to be \((e^x + e^{-x})/2\) where \( e \) is Euler's number.

Special cases:

- If the value is NaN, then the result is NaN.
- If the value is infinite, then the result is positive infinity.
- If the value is zero, then the result is 1.0.

The computed result must be within 2.5 ulps of the exact result.

**Related concepts**

- `sinh()` on page 970
  Returns the hyperbolic sine of a double value.
- `tanh()` on page 971
  Returns the hyperbolic tangent of a double value.

\[ \text{exp()} \]

Returns Euler's number \( e \) raised to the power of a double value.

**Syntax**

```java
\text{exp()} \n```

**Usage**

Special cases:
• If the value is NaN, the result is NaN.
• If the value is positive infinity, then the result is positive infinity.
• If the value is negative infinity, then the result is positive zero.

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

floor()
Returns the largest (closest to positive infinity) double value that is less than or equal to the value and is equal to a mathematical integer.

**Syntax**

```plaintext
floor()
```

**Usage**

Special cases:

• If the value is already equal to a mathematical integer, then the result is the same as the value.
• If the value is NaN or an infinity or positive zero or negative zero, then the result is the same as the value.

**Related concepts**

ceil() on page 964
Returns the smallest (closest to negative infinity) double value that is greater than or equal to the value and is equal to a mathematical integer.

format()
Converts the numeric value to a string representation defined by a format string.

**Syntax**

```plaintext
format("format-string")
```

**Usage**

The format string syntax is compatible to the D4GL "USING" format string. The formatting takes the values of the environment variables DBFORMAT and DBMONEY into account.

### Table 258: Formatting symbols

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Fills with asterisks any position that would otherwise be blank.</td>
</tr>
<tr>
<td>&amp;</td>
<td>Fills with zeros any position that would otherwise be blank.</td>
</tr>
<tr>
<td>#</td>
<td>This does not change any blank positions in the display.</td>
</tr>
<tr>
<td>&lt;</td>
<td>Causes left alignment.</td>
</tr>
<tr>
<td>, (comma)</td>
<td>Defines the position of the thousands separator. The thousands separator is not displayed if there are no digits to the left. By default, the thousands separator is a comma, but it can be another character as defined by DBFORMAT.</td>
</tr>
<tr>
<td>Character</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>. (period)</td>
<td>Defines the position of the decimal separator. Only a single decimal separator may be specified. By default, the decimal separator is a period, however it can be another character as defined by DBMONEY or DBFORMAT.</td>
</tr>
<tr>
<td>-</td>
<td>Displays a minus sign for negative numbers.</td>
</tr>
<tr>
<td>$</td>
<td>This is the placeholder for the front specification of DBMONEY or DBFORMAT.</td>
</tr>
<tr>
<td>(</td>
<td>Displayed as left parentheses for negative numbers (accounting parentheses).</td>
</tr>
<tr>
<td>)</td>
<td>Displayed as right parentheses for negative numbers (accounting parentheses).</td>
</tr>
</tbody>
</table>

**Example**

For DECIMAL and FLOAT data types, *format-string* consists of pound signs (#) that represent digits and a decimal point. This code formats the numeric value of the overall total as a string, producing three places to the left of the decimal point and exactly two places to the right:

```
overallTotal.format("###.##")
```

**getExponent()**

Returns the unbiased exponent used in the representation of a float.

**Syntax**

```
getExponent()
```

**Usage**

Special cases:

- If the value is NaN or infinite, then the result is Float. MAX_EXPONENT + 1.
- If the value is zero or subnormal, then the result is Float. MIN_EXPONENT -1.

**log()**

Returns the natural logarithm (base e) of a double value.

**Syntax**

```
log()
```

**Usage**

Special cases:

- If the value is NaN or less than zero, then the result is NaN.
- If the value is positive infinity, then the result is positive infinity.
- If the value is positive zero or negative zero, then the result is negative infinity.

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

**Related concepts**

[log10() on page 968](#)
Returns the base 10 logarithm of a double value.

\[ \log_{10}(x) \]

Returns the base 10 logarithm of a double value.

**Syntax**

```
log10()
```

**Usage**

Special cases:

- If the value is NaN or less than zero, then the result is NaN.
- If the value is positive infinity, then the result is positive infinity.
- If the value is positive zero or negative zero, then the result is negative infinity.
- If the value is equal to 10^n for integer n, then the result is n.

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

**Related concepts**

- `log()` on page 967
  Returns the natural logarithm (base e) of a double value.

- `max()`
  Returns the greater of two int values.

**Syntax**

```
max(Numeric b)
```

**Parameters**

- `b` - a Numeric argument.

**Usage**

The result is the argument closer to the value of `Integer.MAX_VALUE`. If the argument has the same value as the object's value the result is that same value.

See Also: `Long.MAX_VALUE`

**Related concepts**

- `min()` on page 968
  Returns the smaller of two int values.

- `round()` on page 969
  Returns the closest int to the value.

- `min()
  Returns the smaller of two int values.

**Syntax**

```
min(Numeric b)
```
Parameters

- \( \text{b} \) - a Numeric argument.

Usage

The result is the argument closer to the value of Integer.MIN_VALUE. If the argument has the same value as the objects value the result is that same value.

See Also: Long.MIN_VALUE

Related concepts

max() on page 968
Returns the greater of two int values.

round() on page 969
Returns the closest int to the value.

rint()
Returns the double (floating-point) value that is closest in value to the value and is equal to a mathematical integer.

Syntax

```plaintext
rint()
```

Usage

If two double values that are mathematical integers are equally close, the result is the integer value that is even.

Special cases:

- If the value is already equal to a mathematical integer, then the result is the same as the value.
- If the value is NaN or an infinity or positive zero or negative zero, then the result is the same as the value.

round()
Returns the closest int to the value.

Syntax

```plaintext
round()
```

Usage

The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type int. In other words, the result is equal to the value of the expression:

\[(\text{int})\text{Math.floor}(a + 0.5f)\]

Special cases:

- If the value is NaN, the result is 0.
- If the value is negative infinity or any value less than or equal to the value of Integer.MIN_VALUE, the result is equal to the value of Integer.MIN_VALUE.
- If the value is positive infinity or any value greater than or equal to the value of Integer.MAX_VALUE, the result is equal to the value of Integer.MAX_VALUE.

Related concepts

max() on page 968
Returns the greater of two int values.

min() on page 968
Returns the smaller of two int values.

\textit{signum()}

Returns the signum function of the value; zero if the value is zero, 1.0 if the value is greater than zero, -1.0 if the value is less than zero.

\textbf{Syntax}

\begin{tabular}{|l|}
\hline
\textit{signum()} \\
\hline
\end{tabular}

\textbf{Usage}

Special cases:

\begin{itemize}
\item If the value is NaN, then the result is NaN.
\item If the value is positive zero or negative zero, then the result is the same as the value.
\end{itemize}

\textit{sin()}

Returns the trigonometric sine of an angle.

\textbf{Syntax}

\begin{tabular}{|l|}
\hline
\textit{sin()} \\
\hline
\end{tabular}

\textbf{Usage}

Special cases:

\begin{itemize}
\item If the value is NaN or an infinity, then the result is NaN.
\item If the value is zero, then the result is a zero with the same sign as the value.
\end{itemize}

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

\textbf{Related concepts}

\textit{cos()} on page 965

Returns the trigonometric cosine of an angle.

\textit{tan()} on page 971

Returns the trigonometric tangent of an angle.

\textit{sinh()}

Returns the hyperbolic sine of a double value.

\textbf{Syntax}

\begin{tabular}{|l|}
\hline
\textit{sinh()} \\
\hline
\end{tabular}

\textbf{Usage}

The hyperbolic sine of \( x \) is defined to be \((e^x - e^{-x})/2\) where \( e \) is Euler's number.

Special cases:

\begin{itemize}
\item If the value is NaN, then the result is NaN.
\item If the value is infinite, then the result is an infinity with the same sign as the value.
\item If the value is zero, then the result is a zero with the same sign as the value.
\end{itemize}

The computed result must be within 2.5 ulps of the exact result.
Related concepts

`cosh()` on page 965
Returns the hyperbolic cosine of a double value.

`tanh()` on page 971
Returns the hyperbolic tangent of a double value.

`sqrt()`
Returns the correctly rounded positive square root of a double value.

Syntax

```
sqrt()
```

Usage

Special cases:

• If the value is NaN or less than zero, then the result is NaN.
• If the value is positive infinity, then the result is positive infinity.
• If the value is positive zero or negative zero, then the result is the same as the value.

Otherwise, the result is the double value closest to the true mathematical square root of the value.

Related concepts

`cbrt()` on page 964
Returns the cube root of a double value.

`tan()`
Returns the trigonometric tangent of an angle.

Syntax

```
tan()
```

Usage

Special cases:

• If the value is NaN or an infinity, then the result is NaN.
• If the value is zero, then the result is a zero with the same sign as the value.

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

Related concepts

`cos()` on page 965
Returns the trigonometric cosine of an angle.

`sin()` on page 970
Returns the trigonometric sine of an angle.

`tanh()`
Returns the hyperbolic tangent of a double value.

Syntax

```
tanh()
```
Usage
The hyperbolic tangent of x is defined to be \((e^x - e^{-x})/(e^x + e^{-x})\), in other words, \(\sinh(x)/\cosh(x)\).

Note: The absolute value of the exact \(\tanh\) is always less than 1.

Special cases:
- If the value is NaN, then the result is NaN.
- If the value is zero, then the result is a zero with the same sign as the value.
- If the value is positive infinity, then the result is +1.0.
- If the value is negative infinity, then the result is -1.0.

The computed result must be within 2.5 ulps of the exact result. The result of \(\tanh\) for any finite input must have an absolute value less than or equal to 1. Note that once the exact result of \(\tanh\) is within 1/2 of an ulp of the limit value of ±1, correctly signed ±1.0 should be returned.

Related concepts
- \(\cosh()\) on page 965
  Returns the hyperbolic cosine of a double value.
- \(\sinh()\) on page 970
  Returns the hyperbolic sine of a double value.

The Runtime Class
The Runtime class provides methods that simplify the creation of dynamic designs that change behavior based on the runtime setup.

- Syntax
- Methods
- Usage

Syntax

Methods
Class methods are static methods that do not require an object. The method name is prefixed by the class name.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getDebugLevel()</td>
<td>Numeric, returns the current debug level specified in the environment variable GREDEBUG, or 0 if no debug level was set.</td>
</tr>
<tr>
<td>getEnvironmentVariable(String variableName)</td>
<td>Returns a String containing the value of the specified environment variable.</td>
</tr>
<tr>
<td>getOutputDeviceName()</td>
<td>Returns a STRING indicating the output device name selected in the API call fgl_report_selectDevice.</td>
</tr>
<tr>
<td>getPrinterMediaName()</td>
<td>Returns a STRING indicating the media name specified in the API call fgl_report_setPrinterMediaName.</td>
</tr>
<tr>
<td>getPrinterMediaSizeName()</td>
<td>Returns a STRING indicating the media size name specified in the API call fgl_report_setPrinterMediaSizeName.</td>
</tr>
<tr>
<td>getPrinterMediaTray()</td>
<td>Returns a STRING indicating the media tray name specified in the API call fgl_report_setPrinterMediaTray.</td>
</tr>
<tr>
<td>getPrinterName()</td>
<td>Returns a STRING indicating the printer name specified in the API call fgl_report_setPrinterName.</td>
</tr>
<tr>
<td>getSVGPaperSource()</td>
<td>Returns a STRING indicating the paper source specified in the API call fgl_report_setSVGPaperSource.</td>
</tr>
<tr>
<td>getSVGPrinterName()</td>
<td>Returns a STRING indicating the printer name specified in the API call fgl_report_setSVGPrinterName.</td>
</tr>
<tr>
<td>inDesigner()</td>
<td>BOOLEAN; returns TRUE if the report is currently being edited in Genero Studio; otherwise FALSE.</td>
</tr>
<tr>
<td>producingBrowserOutput()</td>
<td>BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was &quot;Browser&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingExcelOutput()</td>
<td>BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was &quot;XLS&quot; or &quot;XLSX&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingHTMLOutput()</td>
<td>BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was &quot;HTML&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingImageOutput()</td>
<td>BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was &quot;Image&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingOORTFOutput()</td>
<td>BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was &quot;OORTF&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingPDFOutput()</td>
<td>BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was &quot;PDF&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingPostscriptOutput()</td>
<td>BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was &quot;Postscript&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingRTFOutput()</td>
<td>BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was &quot;RTF&quot;; otherwise FALSE.</td>
</tr>
</tbody>
</table>
**Usage**

The class provides a number of methods that simplify the job of creating dynamic designs that change behavior based on the runtime setup.

With RTL classes, it is not possible to create and subclass objects. The `new` keyword is not supported.

The methods can be used from within RTL expressions. Some common uses might be:

- To suppress headers and footers when Excel™ output is selected.
- To conditionally Insert a logo based on the printer tray is selected.
- To set a background color when debugging is enabled

These static methods do not require a Runtime object instance. When you invoke the method, it is prefixed with the Runtime class name and the `.`. For example, you can suppress a header by setting its `visibilityCondition` property to this expression:

```
"!Runtime.producingExcelOutput()"
```

**Related concepts**

**Expressions in properties** on page 847

A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

**getOutputDeviceName()**

Runtime class method that returns a STRING indicating the output device name selected in the API call `fgl_report_selectDevice`.

**Syntax**

```
getOutputDeviceName()
```

**Possible values**

- PDF
- XLS
- XLSX
- HTML
- Image
- Printer
- Postscript
- SVG
- Browser
- RTF
- OORTF

**Related concepts**

`fgl_report_selectDevice` on page 743

Select the output device.

**getSVGPaperSource()**

Runtime class method that returns a STRING indicating the paper source specified in the API call ` fg1_report_setSVGPaperSource`.

**Syntax**

```
getSVGPaperSource()
```
Possible values

- Auto
- Cassette
- Envelope
- EnvelopeManual
- FormSource
- LargeCapacity
- LargeFormat
- Lower
- Middle
- Manual
- OnlyOne
- Tractor
- SmallFormat

Related concepts

`fgl_report_setSVGPaperSource` on page 765
Select the input source of the printer.

The String Class
The String class provides methods for working with text.

- Syntax
- Methods
- Usage and Examples

Syntax

```
String
```
### Methods

#### Table 260: Object Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>charAt(index Numeric)</code></td>
<td>Returns a String containing the character at the specified index in the current String.</td>
</tr>
<tr>
<td><code>contains(s String)</code></td>
<td>Returns a Boolean value (TRUE/FALSE) specifying whether s is contained within the current String.</td>
</tr>
<tr>
<td><code>endsWith(s String)</code></td>
<td>Returns a Boolean value (TRUE/FALSE) specifying whether the current String ends with s.</td>
</tr>
<tr>
<td><code>equals(s String)</code></td>
<td>Returns a Boolean value (TRUE/FALSE) specifying whether s matches the current String. If one of the Strings is null the method returns FALSE.</td>
</tr>
<tr>
<td><code>equalsIgnoreCase(s String)</code></td>
<td>Returns a Boolean value (TRUE/FALSE) specifying whether s matches the current String, ignoring character case. If one of the Strings is null the method returns FALSE.</td>
</tr>
<tr>
<td><code>format (number Numeric, format Enum)</code></td>
<td>Sets the format of the page number string for a PAGENOOBOX. The value for format can be: ARABIC, LOWERROMAN, UPPERROMAN</td>
</tr>
<tr>
<td><code>indexOf(s String)</code></td>
<td>Returns a Numeric value representing the index of s within the current String.</td>
</tr>
<tr>
<td><code>indexOf(s String , index Numeric)</code></td>
<td>Returns a Numeric value representing the index of s within the current String, starting from byte position index. Returns zero if the substring was not found. Returns -1 if the current String is null.</td>
</tr>
<tr>
<td><code>isEmpty()</code></td>
<td>Returns a Boolean value. Returns true if the current string has a length of zero (length()=0), otherwise false.</td>
</tr>
<tr>
<td><code>isNull()</code></td>
<td>Returns a Boolean value. Returns true if the current string has a length of zero (length()=0) and the string is tagged as null, otherwise false. This is the case for null valued input variables read from the input stream. For backward compatibility, null values do not have special behavior when used with the various operators. Specifically an input variable of type string that is null behaves like the empty string.</td>
</tr>
<tr>
<td><code>lastIndexOf(s String)</code></td>
<td>Returns a Numeric value representing the index within the current String of the last occurrence of s, searching backward.</td>
</tr>
<tr>
<td><code>lastIndexOf(s String , index Numeric)</code></td>
<td>Returns a Numeric value representing the index within the current String of the last occurrence of s, searching backward. starting at the specified position index.</td>
</tr>
<tr>
<td><code>length()</code></td>
<td>Returns a Numeric value representing the length of the current String.</td>
</tr>
</tbody>
</table>
Usage and Examples
With RTL classes, it is not possible to create and subclass objects. The `new` keyword is not supported.

All literal String values in an expression must be delimited by double quotes.

All the methods require an object instance. When you invoke the method, it is prefixed with the object instance name and the "." character. You can get an object instance by referencing a 4GL variable or by calling a method on another object. The object can be a literal value, for example:

"Test".length()

Numeric data items in WordBoxes and WordWrapBoxes
If you enter an expression for the text property of a WordBox or WordWrapBox, the value must be a String. You can use the `toString()` function in your expressions to convert numbers to Strings. When you drag a Numeric data item onto the Report Design Window, it is automatically placed in a WordBox element, and an expression for the `text` property is created to convert it to a String.

For example:

`order_line.unitprice.toString()`

The indexes of a String (example `substring`)
When specifying the character position (index) of a string, the first character value is at position 0.

For example, when using the `substring()` function, the substring begins at the specified `startIndex` and ends at the character at `endIndex` -1. The length of the string is `endIndex` minus `startIndex`:

`order_line.billState.substring(1, 5)`

If the value of the String `billState` is "smiles" (indexes 012345), the substring returned is "mile", and the length of the string is 5 minus 1 = 4.

Concatenating Strings
Use the `+` operator to concatenate strings. For example:

("Total:" + order_line.totalorderprice).toString()

This expression returns the current value of `totalorderprice` as part of a String value:

Total: 12.95

Related concepts
Expressions in properties on page 847
A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

Dimensions
The dimensions for units that are used in reports.

Related concepts
Expressions in properties on page 847
A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

Using the PXML expression language on page 854
Genero Report Writer provides the PXML Expression language to define the value of a property that is of the PXML (dimension) type.

**Unit Names**
A list of unit names supported in Genero Studio, for example, point value and centimeter.

### Table 261: Unit Names

<table>
<thead>
<tr>
<th>Unit abbreviations</th>
<th>Unit description</th>
<th>Point value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>point[pt]</td>
<td>Point value (72.27point = 1in)</td>
<td>1</td>
<td>10point</td>
</tr>
<tr>
<td>scrpixels[xy]</td>
<td>Screen pixel value (e.g. for 96DPI: 96pixel = 1in = 72.27point)</td>
<td>Depends on screen resolution of the current screen. The default value is taken from the local VM, not from a potential viewer residing on a different machine.</td>
<td>640scrpixelsx</td>
</tr>
<tr>
<td>prnpixels[xy]</td>
<td>Printer pixel value (e.g. for 300DPI 300pixel = 1in = 72.27point)</td>
<td>Depends on printer resolution of current printer (printer resolution defaults to 300 when there is no printer on the current pipe or when the printer is on a part of the pipe that resides on another machine).</td>
<td>150prnpixelsy</td>
</tr>
<tr>
<td>pica[pc]</td>
<td>Pica value (12point = 1pica)</td>
<td>12</td>
<td>3pica</td>
</tr>
<tr>
<td>inch[in]</td>
<td>Inch value (72.27point = 1in)</td>
<td>72.27</td>
<td>10in</td>
</tr>
<tr>
<td>bigpoint[bp]</td>
<td>Big point (72bigpoint = 1in)</td>
<td>72.27/72</td>
<td>10bp</td>
</tr>
<tr>
<td>cm</td>
<td>Centimeter value (2.54cm = 1in)</td>
<td>72.27/2.54</td>
<td>10cm</td>
</tr>
<tr>
<td>mm</td>
<td>Centimeter value (10mm = 1cm)</td>
<td>72.27/2.54/10</td>
<td>10mm</td>
</tr>
<tr>
<td>didot[dd]</td>
<td>Didot point (1157dd = 1238pt)</td>
<td>1238/1157</td>
<td>10dd</td>
</tr>
<tr>
<td>cicero[cc]</td>
<td>Cicero value (1cc = 12dd)</td>
<td>12*1238/1157</td>
<td>0.5cc</td>
</tr>
</tbody>
</table>

**Paper Format Abbreviations**
A list of paper format abbreviations supported in Genero Studio, for example, ISO 4A0 and Letter.

### Table 262: Paper Format Abbreviations

<table>
<thead>
<tr>
<th>Unit abbreviations</th>
<th>Unit Description</th>
<th>Width value</th>
<th>Length value</th>
</tr>
</thead>
<tbody>
<tr>
<td>iso4a0(width</td>
<td>length)</td>
<td>ISO 4A0</td>
<td>1682mm</td>
</tr>
<tr>
<td>iso2a0(width</td>
<td>length)</td>
<td>ISO 2A0</td>
<td>1189mm</td>
</tr>
<tr>
<td>Unit abbreviations</td>
<td>Unit Description</td>
<td>Width value</td>
<td>Length value</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>(isodesignatedlong</td>
<td>ISO designated long</td>
<td>110mm</td>
<td>220mm</td>
</tr>
<tr>
<td>dl</td>
<td>width</td>
<td>length</td>
<td></td>
</tr>
<tr>
<td>executive(width</td>
<td>Executive</td>
<td>7.25in</td>
<td>10.5in</td>
</tr>
<tr>
<td>length)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(folio</td>
<td>Folio/German legal fanfold</td>
<td>8.5in</td>
<td>13in</td>
</tr>
<tr>
<td>germanlegalfanfold)</td>
<td>width</td>
<td>length)</td>
<td></td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(invoice</td>
<td>Invoice/Statement</td>
<td>5.5in</td>
<td>8.5in</td>
</tr>
<tr>
<td>statement)(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ledger</td>
<td>Ledger/Tabloid</td>
<td>11in</td>
<td>7in</td>
</tr>
<tr>
<td>tabloid</td>
<td>width</td>
<td>length)</td>
<td></td>
</tr>
<tr>
<td>(11x7)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(naletter</td>
<td>Letter</td>
<td>8.5in</td>
<td>11in</td>
</tr>
<tr>
<td>letter</td>
<td>width</td>
<td>length)</td>
<td></td>
</tr>
<tr>
<td>note)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(nalegal</td>
<td>Legal</td>
<td>8.5in</td>
<td>14in</td>
</tr>
<tr>
<td>legal)</td>
<td>width</td>
<td>length)</td>
<td></td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>quarto(width</td>
<td>Quarto</td>
<td>215mm</td>
<td>275mm</td>
</tr>
<tr>
<td>length)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a(width</td>
<td>Engineering A</td>
<td>8.5in</td>
<td>11in</td>
</tr>
<tr>
<td>length)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b(width</td>
<td>Engineering B</td>
<td>11in</td>
<td>17in</td>
</tr>
<tr>
<td>length)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c(width</td>
<td>Engineering C</td>
<td>17in</td>
<td>22in</td>
</tr>
<tr>
<td>length)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d(width</td>
<td>Engineering D</td>
<td>22in</td>
<td>34in</td>
</tr>
<tr>
<td>length)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e(width</td>
<td>Engineering E</td>
<td>34in</td>
<td>44in</td>
</tr>
<tr>
<td>length)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(na10x15envelope</td>
<td>Envelope 10x15</td>
<td>10in</td>
<td>15in</td>
</tr>
<tr>
<td>10x15envelope)(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(na10x14envelope</td>
<td>Envelope 10x14</td>
<td>10in</td>
<td>14in</td>
</tr>
<tr>
<td>10x14envelope)(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(na10x13envelope</td>
<td>Envelope 10x13</td>
<td>10in</td>
<td>13in</td>
</tr>
<tr>
<td>10x13envelope)(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(na9x12envelope</td>
<td>Envelope 9x12</td>
<td>9in</td>
<td>12in</td>
</tr>
<tr>
<td>9x12envelope)(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(na9x11envelope</td>
<td>Envelope 9x11</td>
<td>9in</td>
<td>11in</td>
</tr>
<tr>
<td>9x11envelope)(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(na7x9envelope</td>
<td>Envelope 7x9</td>
<td>7in</td>
<td>9in</td>
</tr>
<tr>
<td>7x9envelope)(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(na6x9envelope</td>
<td>Envelope 6x9</td>
<td>6in</td>
<td>9in</td>
</tr>
<tr>
<td>6x9envelope)(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(nanumber9envelope</td>
<td>Envelope number 9</td>
<td>3+7/8in</td>
<td>8+7/8in</td>
</tr>
<tr>
<td>number9envelope</td>
<td>width</td>
<td>length)</td>
<td></td>
</tr>
<tr>
<td>env9)</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit abbreviations</td>
<td>Unit Description</td>
<td>Width value</td>
<td>Length value</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>------------------------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>(nanumber10envelope</td>
<td>number10envelope</td>
<td>env10) (width</td>
<td>length)</td>
</tr>
<tr>
<td>(nanumber11envelope</td>
<td>number11envelope</td>
<td>env11) (width</td>
<td>length)</td>
</tr>
<tr>
<td>(nanumber12envelope</td>
<td>number12envelope</td>
<td>env12) (width</td>
<td>length)</td>
</tr>
<tr>
<td>(nanumber14envelope</td>
<td>number14envelope</td>
<td>env14) (width</td>
<td>length)</td>
</tr>
<tr>
<td>(envinvite</td>
<td>inviteenvelope</td>
<td>invite)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(envitaly</td>
<td>italyenvelope</td>
<td>italy)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(envmonarch</td>
<td>monarchenvelope</td>
<td>monarch)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(envpersonal</td>
<td>personalenvelope</td>
<td>personal)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(usstandardfanfold</td>
<td>usstdfanfold)(width</td>
<td>length)</td>
<td>US standard fanfold</td>
</tr>
<tr>
<td>(germanstandardfanfold</td>
<td>germanstdfanfold)(width</td>
<td>length)</td>
<td>German standard fanfold</td>
</tr>
<tr>
<td>(iso0</td>
<td>a0</td>
<td>dina0)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(iso1</td>
<td>a1</td>
<td>dina1)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(iso2</td>
<td>a2</td>
<td>dina2)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(iso3</td>
<td>a3</td>
<td>dina3)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(iso4</td>
<td>a4</td>
<td>dina4)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(iso5</td>
<td>a5</td>
<td>dina5)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(iso6</td>
<td>a6</td>
<td>dina6)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(iso7</td>
<td>a7</td>
<td>dina7)(width</td>
<td>length)</td>
</tr>
<tr>
<td>Unit abbreviations</td>
<td>Unit Description</td>
<td>Width value</td>
<td>Length value</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>(isoa8</td>
<td>a8</td>
<td>dina8)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoa9</td>
<td>a9</td>
<td>dina9)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoa10</td>
<td>a10</td>
<td>dina10)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isob0</td>
<td>b0</td>
<td>dinb0)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isob1</td>
<td>b1</td>
<td>dinb1)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isob2</td>
<td>b2</td>
<td>dinb2)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isob3</td>
<td>b3</td>
<td>dinb3)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isob4</td>
<td>b4</td>
<td>dinb4)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isob5</td>
<td>b5</td>
<td>dinb5)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isob6</td>
<td>b6</td>
<td>dinb6)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isob7</td>
<td>b7</td>
<td>dinb7)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isob8</td>
<td>b8</td>
<td>dinb8)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isob9</td>
<td>b9</td>
<td>dinb9)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isob10</td>
<td>b10</td>
<td>dinb10)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoc0</td>
<td>c0</td>
<td>dinc0)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoc1</td>
<td>c1</td>
<td>dinc1)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoc2</td>
<td>c2</td>
<td>dinc2)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoc3</td>
<td>c3</td>
<td>dinc3)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoc4</td>
<td>c4</td>
<td>dinc4)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoc5</td>
<td>c5</td>
<td>dinc5)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoc6</td>
<td>c6</td>
<td>dinc6)(width</td>
<td>length)</td>
</tr>
<tr>
<td>Unit abbreviations</td>
<td>Unit Description</td>
<td>Width value</td>
<td>Length value</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>(isoc7</td>
<td>c7</td>
<td>dinc7)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoc8</td>
<td>c8</td>
<td>dinc8)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoc9</td>
<td>c9</td>
<td>dinc9)(width</td>
<td>length)</td>
</tr>
<tr>
<td>(isoc10</td>
<td>c10</td>
<td>dinc10) (width</td>
<td>length)</td>
</tr>
<tr>
<td>jisb0(width</td>
<td>length)</td>
<td>JIS B0</td>
<td>1030mm</td>
</tr>
<tr>
<td>jisb1(width</td>
<td>length)</td>
<td>JIS B1</td>
<td>728mm</td>
</tr>
<tr>
<td>jisb2(width</td>
<td>length)</td>
<td>JIS B2</td>
<td>515mm</td>
</tr>
<tr>
<td>jisb3(width</td>
<td>length)</td>
<td>JIS B3</td>
<td>364mm</td>
</tr>
<tr>
<td>jisb4(width</td>
<td>length)</td>
<td>JIS B4</td>
<td>257mm</td>
</tr>
<tr>
<td>jisb5(width</td>
<td>length)</td>
<td>JIS B5</td>
<td>182mm</td>
</tr>
<tr>
<td>jisb6(width</td>
<td>length)</td>
<td>JIS B6</td>
<td>128mm</td>
</tr>
<tr>
<td>jisb7(width</td>
<td>length)</td>
<td>JIS B7</td>
<td>91mm</td>
</tr>
<tr>
<td>jisb8(width</td>
<td>length)</td>
<td>JIS B8</td>
<td>64mm</td>
</tr>
<tr>
<td>jisb9(width</td>
<td>length)</td>
<td>JIS B9</td>
<td>45mm</td>
</tr>
<tr>
<td>jisb10(width</td>
<td>length)</td>
<td>JIS B10</td>
<td>32mm</td>
</tr>
</tbody>
</table>

**Report Writer preferences**

Select **Tools > Preferences > Report Writer** to access these preferences.

**Document View**
- The **Prefer to display item name over RTL expression text** checkbox: When selected, user-defined labels are displayed instead of the expressions. A label is considered user-defined if it does not match the generated name "[NodeType][Index]" (e.g. "WordBox12").

**Paper Settings Preferences**
You can specify the default paper settings to be used for report design documents.

- **Orientation**: portrait, landscape
- **Units**: centimeter, inch
- **Page Size Format**: standard - choose a value from the combobox, custom
- **Margins** - left, right, top, bottom

**Related concepts**
*Setting Preferences* on page 128
Customize Genero Studio to meet your needs.

*Specify the paper settings of a report* on page 807
Paper settings set the paper orientation, the page size format, and the report margins for a report.

**Upgrading GRD**

Take the required steps to upgrade to the next release of Genero Report Designer, and identify which features were added for a specific version.

**Update report design documents to the current version**

The `gsreport` command line utility updates report design documents (.4rp) from previous versions to the current version.

**Syntax**

```
gsreport [OPTIONS] filename [filename [...]]
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td></td>
<td>Show help.</td>
</tr>
<tr>
<td>-V</td>
<td></td>
<td>Display program name and version.</td>
</tr>
<tr>
<td>-c</td>
<td></td>
<td>Convert old format .4rp files to the latest format</td>
</tr>
<tr>
<td>-encoding</td>
<td>ENCODING</td>
<td>Set the encoding to ENCODING (default: CP1252)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> When you use the <code>-h</code> option to display command help, the default encoding for your current environment displays.</td>
</tr>
<tr>
<td>-translate</td>
<td>LOCALE</td>
<td>Set the translation file to LOCALE.</td>
</tr>
</tbody>
</table>

**Usage**

This command line utility accepts a list of files, separated by a space. To include all report files in a directory, use the wildcard symbol (*).

Invoked without any options, the program performs a dry run that reports issues but does not modify the files.

Use option “-c” to convert files to the current version.

⚠️ **Warning:** The tool does not back up the files.

If the data schema file (.rdd or .xsd) associated with the report design document is not encoded in system encoding, the `-encoding` option should be used to specify the encoding.

If error messages and warnings need to be displayed in a language different from the system language, the “-translate” option can be used to specify an alternate language.

**GRD 3.10 new features**

Genero Report Designer v 3.10 includes information about new features and changes in existing functionality.

<table>
<thead>
<tr>
<th>Table 264: Genero Report Designer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview</strong></td>
</tr>
<tr>
<td>Original data produced by an application can now be transformed within the report designer without coding. Transformations include selecting, duplicating, moving, re-ordering, pivotizing and computing aggregations.</td>
</tr>
</tbody>
</table>
The Image Box can now embed the first page of a PDF file.

The new Toolbox object, PDFBOX, can embed an entire PDF.

You can create a Spider Web chart.

The business graphs now use a new "flat" design by default, although you can still specify the "classic" style.

If the parent object is a propagating container, the child object does not fit in the remaining space for the parent object, and you set the Y-Size property to rest, the child now expands to the maximum extent of the parent rather than just the remainder of the parent.

For X-Size Adjustment and Y-Size Adjustment, a value of expandToParent now causes the box to stretch as much as possible without intersecting the borders of a parent or sibling.

For business graphs and pivot tables, the new rangeUpperBound and rangeLowerBound properties define the highest and lowest values on the Y-Axis.

For XY charts, the new domainUpperBound and domainLowerBound properties define the highest and lowest values on the X-Axis.

Report elements now include a comment property.

GRD 3.00 new features
Genero Report Designer v 3.00 includes new features and changes in existing functionality.

These changes and enhancements are relevant to this publication.

Table 265: Genero Report Designer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genero Report Designer provides a LastPageFooter section property.</td>
<td>See section (Section) on page 891.</td>
</tr>
<tr>
<td>Support of Intelligent Mail bar code type.</td>
<td>See intelligent-mail on page 939.</td>
</tr>
<tr>
<td>New smartParse bar code property for bar code Code-128. When enabled, this allows you to enter the bar code value, and the internal code will be computed for you resulting in the shortest visual representation.</td>
<td>See smartParse (Smart Parse) on page 907 and code-128 on page 915.</td>
</tr>
<tr>
<td>New gs1* bar code aliases.</td>
<td>See Bar Code type listing on page 911.</td>
</tr>
</tbody>
</table>

GRD 2.50 new features
This publication includes information about new features and changes in existing functionality.

These changes and enhancements are relevant to this publication.

Table 266: Genero Report Designer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables support.</td>
<td>See Working with tables on page 819</td>
</tr>
<tr>
<td>Pivot tables support.</td>
<td>See Working with pivot tables on page 836.</td>
</tr>
<tr>
<td>Overview</td>
<td>Reference</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PDF enhancements. Improved PDF output, to include better memory</td>
<td>No further reference.</td>
</tr>
<tr>
<td>consumption, use of the PDF referencing mechanism to improve Page M of</td>
<td></td>
</tr>
<tr>
<td>N processing, share recurring images and CID keyed fonts support.</td>
<td>See The String Class on page 975 and The Numeric Class on page 961,</td>
</tr>
<tr>
<td></td>
<td>Conditional Expressions.</td>
</tr>
<tr>
<td>Null value support.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See Triggers on page 791.</td>
</tr>
<tr>
<td>Improved trigger updates. Algorithm improved to remove the need for</td>
<td></td>
</tr>
<tr>
<td>frequent manual adjustments for each change within the data schema</td>
<td></td>
</tr>
<tr>
<td>(r dd) file.</td>
<td></td>
</tr>
<tr>
<td>QR code barcode support.</td>
<td>See qr-code on page 941.</td>
</tr>
<tr>
<td>Display position of footers. Layout nodes designated as footers display</td>
<td>See Add headers and footers to a report on page 792.</td>
</tr>
<tr>
<td>at the bottom of the Mini Page, providing a WYSIWYG experience for the</td>
<td></td>
</tr>
<tr>
<td>report designer.</td>
<td></td>
</tr>
<tr>
<td>Element creation by context. Create elements based on the document</td>
<td>See Adding data values and captions on page 797.</td>
</tr>
<tr>
<td>context in the report design. The object type created for a field is</td>
<td></td>
</tr>
<tr>
<td>determined by the location in the document.</td>
<td></td>
</tr>
<tr>
<td>Splitting of oversized elements across pages to prevent overfill.</td>
<td>See splitOversizedItem (Split Oversized Items) on page 892.</td>
</tr>
<tr>
<td>Rotation of items. When the Transform transparently property is set on</td>
<td>See Transform transparently (transformTransparently) on page 894.</td>
</tr>
<tr>
<td>a parent, its children map their orientation based on the parent's</td>
<td></td>
</tr>
<tr>
<td>parent orientation rather than the parent.</td>
<td></td>
</tr>
<tr>
<td>Support for reverse side printing.</td>
<td>See Specify a different reverse side page on page 803.</td>
</tr>
<tr>
<td>Chart sorting. For MapCharts and CategoryCharts, the sortBy property</td>
<td>See sortBy (Sort By) on page 902 and sortAscending (Sort Ascending) on</td>
</tr>
<tr>
<td>allows you to specify how the data is sorted: alphabetic, numeric, or</td>
<td>page 901.</td>
</tr>
<tr>
<td>by order of declaration of the chart items. The sortAscending property</td>
<td></td>
</tr>
<tr>
<td>allows you to sort in ascending or descending order.</td>
<td></td>
</tr>
<tr>
<td>Fallback image support when the requested image for an Image Box is</td>
<td>See Image Box on page 865.</td>
</tr>
<tr>
<td>not found.</td>
<td></td>
</tr>
<tr>
<td>Edit triggers with a Repeat selected items on menu option in the</td>
<td>See Place a trigger within the report structure on page 792.</td>
</tr>
<tr>
<td>context menu in the Report Structure view, allowing you to select a</td>
<td></td>
</tr>
<tr>
<td>trigger to be the parent of a document node.</td>
<td></td>
</tr>
<tr>
<td>Class property added for report elements.</td>
<td>See class (Class) on page 879.</td>
</tr>
<tr>
<td>Display and modify the sizing policy of containers.</td>
<td>See Modify the sizing policy of containers on page 785.</td>
</tr>
<tr>
<td>The fidelity property has been added to business charts and the pivot</td>
<td>See Business Graphs on page 869.</td>
</tr>
<tr>
<td>table, applied only when the object in question is drawn as a table.</td>
<td></td>
</tr>
<tr>
<td>The layout direction of a parent container is highlighted in the</td>
<td>See layoutDirection (Layout Direction) on page 887.</td>
</tr>
<tr>
<td>Genero Report Designer by the addition of a dashed, slowly moving, U-</td>
<td></td>
</tr>
<tr>
<td>shaped yellow border.</td>
<td></td>
</tr>
<tr>
<td>Preference added to control the appearance of RTL expressions in the</td>
<td>See Report Writer preferences on page 982.</td>
</tr>
<tr>
<td>document view.</td>
<td></td>
</tr>
</tbody>
</table>
Create and manage report templates

A report template defines the layout of a professionally-designed report that you can use to quickly create an initial report design.

When you have multiple reports with the same structure, but with different data sources, you can use report templates to provide a consistent look-and-feel for all similar reports.

A template is a .4rt file. It is a general report design that is not tied to a specific data source. It is a graphical design of a report, and you associate data with data placeholders that have been set in the graphical design. You decide which data to inject by specifying the data source, when you select the data schema to use.

Once the data source is selected, two things need to be mapped: the structure of the data with the structure of the template, and the fields from the schema to the data placeholders in the template.

When creating a report, you can choose from the existing list of templates, or you can create your own report template.

Related concepts
Overview of Genero Report Designer on page 778
When you launch the Genero Report Designer, you have the tools you need to design Genero reports.

Designing a Report on page 781
Design a simple or complex report using the views, menus, and tools in the Report Designer.

Create a report from an existing template

In this procedure, the New Report from Template wizard is used to create a report design document (.4rp) from an existing template.

The New Report from Template wizard results in a report design document (.4rp) being created. Once created, the report design document is a stand-alone document and no longer has any connection to the template that created it. Any changes made to the report template (.4rt) have no effect on existing report designs (.4rp).

   A list of templates appear. You can refine the types of templates that appear in the list by selecting from the Filters drop-down list. See Template filters on page 987. Click on a template icon to view a sample of the report that can be created from that template.

2. Select a template and click OK.
   The New Report From Template wizard opens.

3. In the Schema Association page, you provide information about the data source.
   For complete details on this step, see Schema Association page on page 988.
   a) In the Schema Location field, select a data schema (.xsd).
      Once selected, a list of available classes appears in the Schema Root combo-box. The Record1 appears in the rows box.
   b) In the Schema Root combobox, select the schema root.
   c) Complete the Repetitions section, mapping the template repetitions to the schema repetitions.
   d) Click Next >.
4. In the **Add Fields** page, select the fields to display in a report object designed to accept a variable number of fields.

Refer to the report diagram on the right-hand side of the wizard, the object will be highlighted. A new **Add Fields** page displays for each variable-field object contained on your report. For complete details on this step, see **Add Fields page** on page 989.

5. On the **Variables** page, provide values for variables in the template.

The **Placeholders** section provides you with the names of the various variables you can modify. Click on a variable, and the report image highlights the area or object that is affected by the variable.

For complete details on this step, see **Variables page** on page 990.

6. Select **Finish**.

The report is created as an independent `.rpt` file. It is no longer associated with the template. It is a report design definition, and can now be treated as a standard report design document (`.rpt`) file.

**Template filters**

When you start the New Reports from Templates wizard, you can refine the templates that appear in the list by selecting from the filters provided.

**Table 267: List of default filters** on page 987 provides a list of the default filters and a description of each filter.

<table>
<thead>
<tr>
<th><strong>Filter</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Batcheable</td>
<td>Templates that can be used to create a single document that contains multiple sub-documents (for example, a single document that contains invoices from different customers). This is useful to ensure uninterrupted printout.</td>
</tr>
<tr>
<td>Correspondence</td>
<td>Templates for reports that can be used for correspondence purposes, typically containing an address.</td>
</tr>
<tr>
<td>DYNAMIC GREEN Theme</td>
<td>Templates that use the DYNAMIC GREEN graphical style.</td>
</tr>
<tr>
<td>DYNAMIC ORANGE Theme</td>
<td>Templates that use the DYNAMIC ORANGE graphical style.</td>
</tr>
<tr>
<td>DYNAMIC PURPLE Theme</td>
<td>Templates that use the DYNAMIC PURPLE graphical style.</td>
</tr>
<tr>
<td>Grand Total</td>
<td>Templates that can be used to print grand totals so that a running total is calculated with the data.</td>
</tr>
<tr>
<td>Group Headers</td>
<td>Templates that contain headers for groups of data.</td>
</tr>
<tr>
<td>Group Totals</td>
<td>Templates that can be used to print group totals for the groups available from the data.</td>
</tr>
<tr>
<td>Invoice</td>
<td>Templates that create an invoice document</td>
</tr>
<tr>
<td>List</td>
<td>Templates that create a list report.</td>
</tr>
<tr>
<td>PLAIN Theme</td>
<td>Templates that use the PLAIN graphical style.</td>
</tr>
<tr>
<td>PULSE Theme</td>
<td>Templates that use the PULSE graphical style.</td>
</tr>
<tr>
<td>Totals Block</td>
<td>Templates that create a summary block on the last page.</td>
</tr>
<tr>
<td>Filter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td>Two Groups</td>
<td>Templates that requires the data to be grouped by at least two dimensions.</td>
</tr>
</tbody>
</table>

**Note:** When creating a new template, you can add existing filters to your template or create new filters to add to the list. See Customize the appearance of a new report template in the wizard on page 998.

**Schema Association page**

The Schema Association page in the New Report From Template wizard lets you select your schema and associate schema repetitions to template repetitions.

There are two sections on the Schema Association page.

![Schema Association](image)

**Figure 418: Schema Association**

**Schema**

In the Schema Location field, you select the name of the data schema file to use. Selecting this file populates both the Schema Root combobox and Repetitions section with suggested default values.

The Schema Root field, select the root of the data model. If you have a master and details listed, select the master. If the data schema was generated with the Business Application Modeler, there will only be one schema root.

**Repetitions**

A report template has repetitions. You can repeat the whole document, you can repeat sections of the report, and you can repeat each row. The sections of the report that repeat are listed under the Template Repetitions column. When you click on a row, the associated repetition is highlighted in the report sample.

A data schema has repetitions. These repetitions represent the hierarchy of data, as defined in the ORDER BY clause in the query used to extract the data. Once added to a report design document, these repetitions are identified by
Report triggers. When you click on a cell in the Schema Repetitions column, a combobox appears with a list of possible schema repetition fields to select from.

In completing this section, map the template repetitions to the schema repetitions. If the data schema has groups defined, the wizard attempts the mapping for you.

To change a group setting, click in the Schema Repetitions cell for that group. A combobox appears, allowing you to select one of the other grouping fields defined in the data schema. The section of the report that relates to this group is highlighted in the report sample.

The last row of the repetitions box is for the individual data rows themselves; the rows that will appear in the listing. For most reports, this is going to be the record defined in the data schema.

**Add Fields page**
The Add Fields page in the New Report From Template wizard lets you select fields for a report object that is designed to accept a variable number of fields.

This page displays for each report placeholder that can accept a variable number of fields. As such, it may not appear at all, or it may appear multiple times. The report image on the right-hand side of the wizard highlights the area of the report that is in focus when using this page.

There is only one section in the Add Fields page, the Select fields section.

![Add Fields](image)

**Figure 419: Add Fields**

Select the fields to use for the highlighted placeholder. While you can click the double-arrow icon to include all fields in the placeholder, it is typical to pick a subset of the fields to populate the placeholder.

The order of the columns in the list determines the order they appear in the row.
Aggregate fields

As a report designer, you have received a data schema from the reporting application developer. The developer may have created this application using the Business Application Modeler. One of the options available to the developer is to specify that a field be aggregated: give me the sum of the price of all orders, or give me a count of the number of orders.

In the Select fields list, these aggregates appear for each level of the grouping, as well as for the overall report.

For example, each order has a field called totalprice, that gives the total price for that order. The total price is something that is typical to summarize in a report, so the developer will specify that the SUM should be provided for the totalprice field. The data is grouped by account and by country, which means we are given fields for the sum of all orders for an account or the sum of all orders for a country.

The names generated for these fields are shown in Figure 420: Aggregate fields on page 990.

![Select fields](image)

**Figure 420: Aggregate fields**

The non-summarized field in this example is orders_totalprice. When the designer specified that a summary be kept, the additional fields added were:

- order_totalprice_Account_userid_sum to hold the sum of the totalprice for the current account.
- order_totalprice_Country_code_sum to hold the sum of the totalprice for the current country.
- order_totalprice_grand_sum to hold the running total of the totalprice for all the rows.

Further examination of the fields in our example also show the aggregate fields created for the count aggregate.

Variables page

The Variables page in the New Report From Template wizard lets you provide values for the variables defined in the template.

There is only one section in the Variables page, the Placeholders section.
In the **Placeholders** section, a table lists all template variables. Click on a row containing a variable to see its location in the report; as you select a variable, the section of the report that relates to the variable highlights in the report sample.

Each variable can be set to a single value. There are three types of values that can be applied:

- A value can be static text. To enter a fixed value, type the value directly into the cell.
- A value can be a field name from your data source. Click on a cell to activate the combobox and select a field from the list.
- A value can be an expression. Click on the expression icon (fx) to launch the expression editor.

**Tip:** An expression can be comprised of multiple fields. For example, you may wish to put a customer's first and last name in a field. You would use an expression to concatenate the fields, with proper spacing.

The types of variables will depend on the template; a template designer has complete freedom to define the variables necessary for the template being developed. Here are some examples of types of variables:

- Variables can hold strings. Common variables of this type are report titles.
- Variables can display the content of a field or fields. Common variables of this type would be including grouping data in sub-sections of a report.
- Variables can be boolean flags. You can have a variable that includes a section to print page totals, based on whether you specify 1 (true) or 0 (false). For example, see printPageTotals in **Figure 421: Variables page** on page 991

These are just a few examples of the types of variables that one may have added to a report template. As a user of a template, ensure you know what each variable does, and what types of values are expected.
Expanding templates at runtime
Genero Report Writer provides the option to add data sources and the associated mapping to a template at runtime. Therefore, you can provide end users with generic reports that can be expanded at runtime.

This method uses a report template (.4rt) and design-time APIs to provide the details about the schema, the schema root, the relationships, and the field mappings in order to output the report, bypassing any need for a report design document. While more complex to set up, the advantage is that changes to your template are reflected with each new run of the report.

Generic reports typically present themselves to end users in the following three phases:

1. The user is prompted for information regarding the template to use and the values to use (variables and placeholders). While all the same options as sophisticated as the template assistant in the report designer are available, typically the following things will be simplified:
   - You can't choose multiple different templates. If choices are given, they are limited to styles that match the data source (e.g. don't offer a grouped list when the data source doesn't have groups).
   - You won't be prompted for difficult placeholders, for example those that require the construction of formulas (e.g. Total expressions), and will only be prompted for simple values (e.g. the Title of the report).
   - You must choose fields from a single list of fields. Therefore, templates that offer more than one field lists will not be used.

   Note: Building the dialog in a generic way to avoid hard coding placeholder and field lists requires software that can introspect schema files and .4rt files. A library is provided for that.

2. The information entered by the user is used to expand the template and generate a .4rp file.

   Note: Expanding the template can be done in one of two ways:
   - By calling an existing library function.
   - By invoking the GenerateReport executable provided with GRE.

3. The data source is run using the generated .4rp file.

   Note: The data source is run with the .4rp via the normal runtime API functions.

Create a new report template
In addition to the templates provided, you can create your own report templates and add them to the Designer Wizard.

To create a new report template, first create a report template schema definition (.rsd) to define the expected structure for the data source and then create the report template (.4rt). Any new templates must be added to the Designer Wizard.

Before you begin: If you have not yet setup a custom directory to store your templates (e.g: My Report Templates), see Set the report template directory on page 1001.

Create a new report template schema
1. Go to File > New > Reports, select Report Template Schema Definition(.rsd), and click OK.
   A new .rsd file opens.

2. Using the available elements, create a structure for the elements that will be used as placeholders in the report template.

   For example:

   ```xml
   <?xml version="1.0" encoding="utf-8"?>
   <ReportSchema rootElementName="verySimpleSchema" fileVersion="30000"
   gstVersion="30000" >
   <Field name="testBool" type="boolean" sampleValue="1"/>
   <Field name="testString" type="string"/>
   <Field name="testNumber" type="double"/>
   </ReportSchema>
   ```

   See the Report template schema definition file (.rsd) on page 993 topic for more information.
3. Save the file to your report template directory.

Create a new report template

4. Go to File > New > Reports, select Empty Report Template(.4rt), and click OK. A new .4rt file opens.

5. In the Data View tab, click Open Schema File.

6. Navigate to the location that you saved your .rsd file in Step 3, select the .rsd file, and click Open. The structure of the elements from the .rsd file that you selected is visible in the Data View tab.

7. Save the .4rt file to your report template directory. You have now created a report template that you can design to suit your requirements.

Note: Before you design your report, see Report template (.4rt) file design features on page 995 for more information.

When you have completed the above procedure, go to File > New to open the wizard. Select Report from Template to view your new report template in the list.

Report template schema definition file (.rsd)

The report template schema definition (.rsd) file defines the field, trigger, and template field trigger elements that serve as placeholders in a report template.

The .rsd file is a high level data schema file that creates the XML structure of the elements that will be included in the report template as placeholders. It is unique to report templates, and is used only when creating a report template (.4rt).

When creating a report from a template, the .rsd placeholders may be populated with data from the report design document data schema, or may be populated by static values entered by the report designer.

Creating a report template schema definition file

An .rsd file starts with a rootElementName element. Withing this element, structure the file with the elements identified in Table 268: Available elements in the .rsd document on page 993.

Table 268: Available elements in the .rsd document

<table>
<thead>
<tr>
<th>Element in .rsd</th>
<th>Required syntax</th>
<th>Description</th>
<th>In New Report from Template wizard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>• name=&quot;&quot;</td>
<td>The Field element adds the variables to include as placeholders in the report template. The type can be a string, double, integer, boolean, or date.</td>
<td>Placeholders on the Variables page.</td>
</tr>
<tr>
<td></td>
<td>• type=&quot;&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• sampleValue=&quot;&quot; (optional)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trigger</td>
<td>name=&quot;&quot;</td>
<td>The Trigger element represents the groups to add as template repetitions to the report template. These repetitions can be mapped to the repetitions in the data schema.</td>
<td>Repetitions on the Schema Association page.</td>
</tr>
<tr>
<td>Element in .rsd</td>
<td>Required syntax</td>
<td>Description</td>
<td>In New Report from Template wizard</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>-------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>TemplateFieldsTrigger</td>
<td>groupName=&quot;&quot;</td>
<td>The TemplateFieldsTrigger element specifies where a dynamic list of fields can be included in a report. When a report is created from a template and the template is expanded, the report designer can choose to populate this element from the fields in the report design data schema.</td>
<td>Each uniquely-named TemplateFieldsTrigger group becomes an Add Fields page. See also Template field triggers on page 995.</td>
</tr>
</tbody>
</table>

Examine the template schema definition files in $GREDIR/templates to understand the different ways that a template schema definition file can be written.

**Sample**

The following sample is a subset of the .rsd file for the **DIN 5008 Invoice (PULSE)** default template:

```xml
<?xml version="1.0" encoding="utf-8"?>
<ReportSchema fileVersion="30000" gstVersion="30000"
rootElementName="model">
  <Field name="showMeasures" type="boolean" sampleValue="1"/>
  <Field name="fontName" type="string"/>
  <Field name="logoURL" type="string"/>
  ...
  <Trigger name="outerGroups">
    <Trigger name="innerGroups">
      <TemplateFieldsTrigger groupName="fields"/>
      <TemplateFieldsTrigger groupName="fields"/>
      <Trigger name="rows">
        <TemplateFieldsTrigger groupName="fields"/>
      </Trigger>
    </Trigger>
  </Trigger>
</ReportSchema>
```

In this file:

- `FileVersion` and `gstVersion` are determined by the version of Studio you are running. When you create a new .rsd file, these values are already set.
- `rootElementName` is the name of the top node in the Data View of the report template.
- The Boolean field `showMeasures` and the String fields `fontName` and `logoURL` become placeholder variables in the report template.
- The triggers `outerGroups`, `innerGroups`, and `rows` become repetitions in the report template.
- Three template triggers are created in the report template.
- However, because all the TemplateFieldsTrigger groups have the same name (`fields`), only one Add Fields page is created in the New Report from Template wizard.
Report template (.4rt) file design features
A report template can contain the same design elements as a standard report design document (.4rp), as well as elements unique to report templates.

The elements unique to report templates are template field triggers and template fields. The former are defined in the template schema definition file, and the latter are added to a report template from the Toolbar view.

Related concepts
Create a report design document on page 777
The Report Design document (.4rp) defines the report page, including the report data and the report elements for organizing and displaying this data.

Example 1: Template with placeholders on page 1005
In this example, you create a report template that uses two placeholders. The placeholders will appear on the Variables page of the New Report from Template wizard.

Example 2: Template with data groups and repetitions on page 1008
In this example, you create a template with a trigger.

Example 3: Template with a dynamic list on page 1010
In this example, you create a template with a dynamic list.

Template field triggers
The template field trigger specifies where a dynamic list of fields can be included in a report created from a template.

When you create the report template schema definition (.rsd), you add the `TemplateFieldsTrigger` element to provide a placeholder for data. For example:

```xml
<TemplateFieldsTrigger groupName="fields"/>
<TemplateFieldsTrigger groupName="anotherName"/>
```

When you associate the report template schema definition with the report template, each `TemplateFieldsTrigger` group becomes a template field trigger. In the Report Structure view, the template field trigger displays as blue to distinguish it from other triggers. You can place items under this trigger, as in Figure 422: The template fields trigger in the Report Structure view on page 995.

![Figure 422: The template fields trigger in the Report Structure view](image)

When the user creates a report from the template, the template field trigger displays as the Add fields page in the New Report from Template dialog. Each uniquely-named `TemplateFieldsTrigger` group becomes a separate Add Fields page. See Figure 423: New Report from Template dialog box: Add fields on page 996.
Add Fields [fields]
Select the fields to display in report

1. Schema Association
2. Add Fields [fields]
3. Add Fields [anotherName]
4. Variables

Figure 423: New Report from Template dialog box: Add fields

Related concepts
Triggers on page 791

The report data is passed from your report program to the report one row at a time, sorted by the criteria specified in your report. The triggers in the report structure specify what should be printed when a change in data occurs.

Template fields
The template field is a Toolbox element unique to the report template.

Use a template field when you want to:

- Create different report elements for different fields, depending on the data. For example, you might want a string field to create a Word Box or Word Wrap Box, and a numeric field to create a Decimal Format Box.
- Create a dynamic number of report elements, depending on the number of fields selected.

The Template Field can be found under Templates in the Tool Box view. See Figure 424: The Template Field element in the Tool Box view on page 996.

Figure 424: The Template Field element in the Tool Box view
Add template fields to the report template in the same way as other elements. Figure 425: The template field in the Report Structure view on page 997 shows an example of a template field in the Report Structure view. In most cases, you place a template field as a descendant of a template field trigger.

Figure 425: The template field in the Report Structure view
When you create a report using a template containing a Template Field, the user is prompted to select the fields from the data schema to populate the template field. The number of fields they can select is unlimited, although there may be a logical limit to how much data they want to add to that section of the report. In the final report, the Template Field becomes one or more standard report elements (such as Word Boxes or Image Boxes), depending on which fields were selected. For example, if a DECIMAL field is selected for inclusion in the Template Field, a Decimal Box is placed on the report.

**Template field properties**

The template field has the following unique properties:

**Note:** In most cases, you do not need to change the defaults for these properties.

- **Name**: Name of a data variable. The data is specified when a report is created from the template.
- **Type**: Type of data variable, for example, CHAR or INTEGER.
- **Size**: Size of the data variable. The units depend on the type, for example, character types are in bytes or characters, while decimal types are in digits.
- **Title**: Title of the template field.
- **Role**: Determines what report element is created by the field. If this value is not specified, the template field creates objects based upon the data variable properties and contextual information.

**Customize the appearance of a new report template in the wizard**

You can add text and images to customize the appearance of your new template in the Wizard.

Templates in the Designer Wizard include preview images, a label and description, and are organized into different categories by filters. When you have created a new template and added it to the design wizard, you can customize the appearance of your new template in the wizard.

**Template label, description, and filters**

Templates are categorized by filters. You can view the list of filters by going to **File > New...**, selecting **Report from Template**, and clicking on the **Filters** drop-down list.

**Note:** The template will only appear in the **Report from Template** list if the report template directory has been set. See [Set the report template directory](#) on page 1001.

Each template has a label, a description, and is filtered by the tags that are included in the associated properties file (.4rt.prop). For more information about the list of existing filters, see [Template filters](#) on page 987.
Figure 426: Report from Template wizard: A = Filters, B = Label, C = Description

See the following example of the syntax in the PULSEDIN5008Template.4rt.prop:

```
tags: Correspondence, Invoice, Batcheable, Two Groups, PULSE Theme
label:DIN 5008 Invoice (PULSE)
description: General: Invoice following the placement guidelines of the German norm DIN 5008
  Paper Format: A4
  Style: PULSE
  Remarks:
    - The measurements can be included in the unrolled report as an editing aid
    - The report is batcheable by the outer group
```
Template images

In the Report From Template page in the Designer Wizard, each template has a thumbnail image in the list and a large version of the same image displayed on the right of the page. This image is a screen capture of the report and is stored in the $ProjectDir/gre/templates directory as <TEMPLATENAME>.4rt.png.

When you have selected a template in the Designer Wizard and the New Report from Template Wizard opens, images are displayed in each of the three pages; the Schema Association page, the Add Fields page, and the Variables page, that are designed to call-out the specific section of the report that is relevant to the current selection in the form.

The sample template images are stored in the $ProjectDir/gre/templates/<TEMPLATENAME> directories and adhere to the following naming conventions:

- Schema association images: <PLACEHOLDERNAME>_placeholder.png
- Add fields images: field_field<NUMBER>.png
- Variables images: <TRIGGERNAME>_trigger.png

Create a report template from an existing report

You can reuse the content from your report by using that report as a basis for a new report template.

You have an existing report, and you want to use it as the basis for a new report template. When you have finished, you must add the new templates to the Designer Wizard.

Before you begin:

- Have a new directory on your disk (e.g: My Report Templates) where you will store the report template files that you create in the following procedure. If you have not set up a template directory, see Set the report template directory on page 1001.
- Have an existing report.

1. Create a copy of your report design (.4rp) and save it in your report templates folder.
2. Rename the copy to a different name, and with a .4rt extension.
   It is recommended that you not have a .4rp and .4rt sharing the same name in a template folder. If your report is names SalesList.4rp, you would not simply name it as SalesList.4rt. You would modify the name in some manner; for example, you can add the word Template to the name and save it as SalesListTemplate.4rt.
3. Open the .4rt file in Report Designer and examine the report. In the next step, you must identify which fields are variable placeholders and which parts are template fields.
4. Create a report schema template definition (.rsd).
   Rather than starting with an empty file, it is recommended that you take an existing report data schema (.rsd) file from the templates provided during the installation and modify it to meet your report needs. You can find report schema template definitions (.rsd) in $GREDIR/gre/templates.
5. Return to your .4rt file, to the Data View tab, and change the schema to the schema you created in the previous step.
   You are not going to use the actual data fields anymore, you are going to use the fields from your template schema. In the Design View, the places where you had data fields (from the original report) are now marked as errors.
6. For the variable placeholders, double-click on each error to display the Edit Expression dialog. Replace the value with one of the names from the template schema file.
7. For the row of fields, such as those contained within a Table Row container, you must replace the fields with a Template Field placeholder.
   a) Delete the existing fields.
   b) Go to Tool Box.
   c) Drop the Template Field onto the Table Row (or similar container).
   d) In the Template Field section in the Properties view, the Name, Type, Size, Title and Role properties should be preset.
When you drag something from the Data View into the Document View, it creates an object based on the context of where it is being placed. These five properties provide the information needed to create the correct object. This field is repeated as many times as needed.

8. Replace any column headers with a Template Field.
9. In the **Report Structure**, organize the TemplateField objects to sit under the fields triggers.
   
   Right-click on a TemplateField, select **Repeat selected items on**, then select the template fields trigger. The template is now finished.

10. Create a copy of a `.prop` file, and modify the contents of the file for your new template. Give the copy the same name as the template file.
11. Create a template image (.4rt.png) to serve as the image displayed in the Report From Template selection list. Save it with the same name as the template file.
12. Create an empty image directory for the new template, with the same name as the template file.
   
   Eventually, you will provide an image for each of the placeholders. This can be done at a later time.

13. Select **Tools > Specific setup > Reload**.

   The new template is available for use.

**Modify an existing report template**

You can modify an existing report template and add them to the Designer Wizard.

To modify an existing report template, you need to create copies of the existing template files and then modify the copies. When finished, the modified templates must be added to the Designer Wizard.

1. Create a new directory on your disk (e.g: **My Report Templates**) to hold your customized report template files.
2. Locate the files for the existing report template, and move them into your new directory.
   
   These files include:
   
   - `*.4rt` - the existing report template file.
   - `*.4rt.png` - the existing image used for the report template file in the new report from template wizard.
   - `*.4rt.prop` - the existing configuration file used to categorize the report template.
   - `*.rsd` - the report schema design file used by the report template.
   
   - the template-specific sub-directory, containing the many images used by pages within the template wizards.
3. Rename the files and sub-directory to use a unique name. All of the files (with the exception of the .rsd file) and the image directory should share the same name.
5. Update the `*.4rt.prop` file to specify your template name, description, and filtering tags.
   
   See **Customize the appearance of a new report template in the wizard** on page 998.
6. Update the `creatables.conf` file in your user-based `$GSTSETUPDIR` directory and add details about the directory.

   See **Set the report template directory** on page 1001 for more details.
7. Update the image files, as necessary. See the section on template images in **Customize the appearance of a new report template in the wizard** on page 998.

**Set the report template directory**

When you create a new report template directory, you need to configure Genero Studio to recognize the directory. Templates used to create a new report must appear in the **File > New > Report from Template** menu. When you create a directory to hold your custom template files, you must update the `creatables.conf` configuration file to add the custom templates to the menu.
The initial template directory is defined in the createables.conf file provided in the Genero Studio installation directory (at \$GSTDIR/conf). It is recommended that you do not modify this installed createables.conf file, but create a user-specific createables.conf file in your \$GSTSETUPDIR directory.

Before you begin, know the full path to the new template directory.

1. Open the creatables.conf file from your \$GSTSETUPDIR directory (for example, C:\Users\Jean Dupont\AppData\Roaming\FourJs\Genero Studio 3.20.28-150486\tpl\dbapp4.0).

   **Tip:** To see the value \$GSTSETUPDIR directory, go to **Tools > Genero Configurations** and select the environment set for your template.

2. Add the following text directly under the <Creatables> root:

   ```xml
   <DocumentDirectory index="35" label="Report From Template"
   name="RWReportFromTemplate" icon="document_4rp"
   directoryPath="DirectoryPath">
   <DocumentType extension="4rt" icon="document_4rt"
   action="RWTemplateWizard"/>
   <DocumentType extension="4rp" icon="document_4rp"
   action="RWTemplateWizard"/>
   </DocumentDirectory>
   ```

   Update the `directoryPath` parameter to the location of your template files.

   **Note:** This createables.conf file is merged with the GST installation createables.conf file (located in \$GSTDIR/conf). The index attribute of DocumentDirectory indicates where the node is inserted in the merged file. If the DocumentDirectory node's name is the same as an existing node in the GST installation createables.conf file, then the two nodes are merged.

   **Tip:** To list multiple directory paths, separate the paths with a semicolon, for example, `directoryPath="D:/myTemplates;C:/Users/JeanDupont/MyTemplates"

3. Select **Tools > Specific Setup > Reload**.

   The templates within the specified directory appear in the **Report From Template** selection list when you select **File > New**.

**Related concepts**

- **The code generation template set** on page 1110
- Genero Studio provides a standard template set of files that are used for code generation. The files are written in XML and Tcl.

- **creatables.conf elements** on page 1163
- A listing of all available elements in the creatables.conf file.

- **Environment sets** on page 164
- Define sets of environment variables to add to or overwrite the default environment. These environment settings are also usable by each Genero Studio client that adds a remote Genero Studio Server to its list.

**GenerateReport command options**

The GenerateReport command creates report design files (.4rp) based on a predefined template and schema.

**Table 269: GenerateReport options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-help</code> or <code>-h</code></td>
<td>Displays a usage text and then exits.</td>
</tr>
<tr>
<td><code>-schemaFileName</code></td>
<td>Specifies an XML schema file (.xsd) describing the data source of the report.</td>
</tr>
</tbody>
</table>

**Important:** Mandatory option.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rootElementName</td>
<td>Specifies the expected document root in the XML schema file.</td>
</tr>
<tr>
<td><strong>Important</strong>: Mandatory option.</td>
<td>For example, if the schema specifies the elements &quot;invoice&quot; and &quot;invoice-batch&quot;, then rootElementName would be set to &quot;invoice&quot; if the report will be run against a source that produces documents whose root element is of type &quot;invoice&quot;.</td>
</tr>
<tr>
<td>-triggerMapping</td>
<td>Specifies the mapping between the element names in the XML schema file and the trigger names in the design template.</td>
</tr>
</tbody>
</table>
| **Important**: Mandatory option. | The map syntax is as follows: map: map-item (',' map-item)*
map-item: '{' element-name ',,' trigger-name '}'
As an example, consider a report template designed against the schema defined by SimpleListTemplate.rsd.
**Note**: Report template schema definition (.rsd) files are located in GREDIR/templates.
The schema defines the mappable triggers "outerGroups", "innerGroups" and "rows", where each is a descendant of its predecessor. If the input schema defines the elements "ProductGroups", "Areas", "Orders" and "Items", then the following are valid maps:

- {ProductGroups,outerGroups}, {Areas,innerGroups}, {Items,rows}
- {ProductGroups,outerGroups}, {Orders,innerGroups}, {Items,rows}
- {ProductGroups,outerGroups}, {Orders,rows}
- {ProductGroups,rows}
- {Orders,rows}

The following example mappings are invalid because they violate the ancestry:

- {ProductGroups,innerGroup}, {Areas,outerGroups}, {Items,rows}
- {Orders,outerGroups}, {ProductGroups,rows}
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| -placeholderMapping | Specifies the mapping between fields names in the design template and expressions of the same type that may be composed using fields from the XML schema file. Specifying this value is mandatory if the template contains references to fields.  
**Note:** All placeholder values in placeholderMapping can either be constant values or RTL expressions enclosed in curly braces.  
The map syntax is as follows:  
```plaintext
StringMap:          MapEntry (',' MapEntry)*  
MapEntry:           '{' Key ',' Value? '}'
Key:                IdentifierStartChar  
IdentifierFollowChar*  
Value:              '"' StringToken* '"'
StringToken:        \[^"\]  
|                   | EscapedQuote  
|                   | EscapedBackslash  
|                   | ExtraEscapes  
EscapedQuote:       '\"' '"'  
EscapedBackslash:   '\ '  
ExtraEscapes:       '\n'  
|                   | '\r'  
|                   | '\t'  

This means that encoders need to perform the following operations on all characters in the input strings:  
- Replace `'\' by `'\ '  
- Replace `" by `'\ '"'  
- Replace `\n by `'\ 'n'  
- Replace `\r by `'\ 'r'  
- Replace `\t by `'\ 't'  

As an example, consider a report template that contains:  
- the string field "groupTitle", mapped to the RTL expression "Customer: +orderline.orders.user_id"  
- the field "reportTitle", mapped to the constant string "Customer list"  
- the placeholder "optionalSubtitle", set to null  

In this example, the placeholder mapping would be:  
```python
-placeholderMapping {groupTitle,"{{"Customer: \"+orderline.orders.user_id}}"},  
    {reportTitle,"Customer list"},{optionalSubtitle,}
```  
This assigns the RTL expression  
"Customer: +orderline.orders.user_id" to the placeholder "groupTitle", the constant string "Customer list" to the placeholder "reportTitle", and the value null to the placeholder "optionalSubtitle".  
**Note:** For clarity, no quoting was done to protect the string against shell expansion.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-templateFileName</td>
<td>Specifies the name of the template (a '.rp' or '.rt' file) used as the base. If this parameter is not specified, then a default list template is used. The default template is designed against the schema of the SimpleListTemplate.rsd.</td>
</tr>
<tr>
<td></td>
<td><strong>Notes:</strong> Report template schema definition (.rsd) files are located in GREDIR/templates.</td>
</tr>
<tr>
<td>-fieldNamePatterns</td>
<td>Specifies a selection of fields from the XML schema file that are to be used in the resulting report.</td>
</tr>
<tr>
<td></td>
<td>The expected syntax is a comma-separated list of field name patterns, which may contain the wildcard characters &quot;*&quot; and &quot;?&quot;. The expression can be prefixed with an optional name followed by a colon. This name denotes a specific field trigger to cater to templates with multiple field lists.</td>
</tr>
<tr>
<td></td>
<td>As an example, consider a report that has the field triggers &quot;outerGroupFields&quot; and &quot;rowFields&quot;. We would like to see the fields product_id and product_description on the group, and all fields from the record order_details in the rows. We would specify two fieldNamePatterns as follows:</td>
</tr>
<tr>
<td></td>
<td>-fieldNamePatterns&lt;br&gt;outerGroupFields:product_id,product_description&lt;br&gt;-fieldNamePatterns rowFields:order_details.*</td>
</tr>
<tr>
<td>-outputFileName</td>
<td>Specifies the name of the resulting .rp file.</td>
</tr>
<tr>
<td></td>
<td><strong>Important:</strong> Mandatory option.</td>
</tr>
<tr>
<td>-debuglevel level</td>
<td>Sets the debug level to the specified integer level. The debug level controls the level of verbosity of GRE components during execution. Higher values increase verbosity. By default, the value is set to 0 (no debugging output).</td>
</tr>
<tr>
<td>-stdin</td>
<td>Instructs the program to read the command line arguments from stdin. The list of arguments needs to be terminated by an empty line. In this case, all other regular command line arguments are ignored.</td>
</tr>
</tbody>
</table>

**Example 1: Template with placeholders**

In this example, you create a report template that uses two placeholders. The placeholders will appear on the Variables page of the New Report from Template wizard.

When users create a report from this template, they must supply values for all placeholders in the template. Users can:

- Populate a placeholder with a literal value; for example, "Orders By Product Groups And Areas".
- Populate a placeholder with a variable from the data schema; for example, `{orderline.product.catname}`.

**Example 1: Create the files required for a "Users" report template**

Create the files to support the new "Users" report template. The template involves a report design with two variables: a string field and a numeric field.

**About this task**

You will create four files:

- Properties file: Users.rp.prop.
Before you begin:

- Ensure that you have a directory on your disk (for example, MyReportTemplates) to hold your report template files. You may need to configure Genero Studio to recognize the directory.
- In Genero Studio, create the project (for example, MyTemplates.4pw) to hold the template files.

1. Create a template schema definition file (File > New > Reports > Template Schema Definition (.rsd)) and enter the following text:

   ```xml
   <?xml version="1.0" encoding="utf-8"?>
   <ReportSchema fileVersion="30000" gstVersion="30000"
     rootElementName="model">
     <Field name="StringUserName" type="string" sampleValue="John Doe"/>
     <Field name="NumericUserID" type="double" sampleValue="1234"/>
     <Trigger name="requiredButNotUsed" minOccurs="0"
      maxOccurs="unbounded"/>
   </ReportSchema>
   
   Note: The trigger is required for the template to work, but it is not used in this template.
   
   Save the template schema definition file as Users.rsd in your report template directory.
   
   This code creates two fields:
   - StringUserName of type String with a default value of John Doe.
   - NumericUserID of type Double with a default value of 1234.

2. Create a report template file (File > New > Reports > Empty Report Template (.4rt)):
   a) In the Data View, open the Users.rsd schema.
   b) Drag and drop the StringUserName and the NumericUserID fields from the Data View to the work area. A Word Wrap Box and a Decimal Format Box are created, using the appropriate class and data.
   c) Add Word Boxes and edit the text to provide labels. The report template should look like Figure 427: Report template example on page 1006.

   ![Figure 427: Report template example](image)
   
   Save the report template file as Users.4rt in your report template directory.
3. Create a text file in the same directory as the template files. Save it as Users.4rt.prop and enter the following text:

```
tags: Users
label: User Name and ID
description: Template with two fields
```

4. Create an image file in the same directory as the template files. Save it as Users.4rt.png.

   **Note:** You can take a screen capture of the existing template, or copy an image from elsewhere.

5. Select **Tools > Specific Setup > Reload**.

**Example 1: Use the "Users" template to create a report**
Create a report using the new template.

1. Go to **File > New > Report from Template**.
2. In the **Filters** drop-down menu, select **Users**, and then select the **User Name and ID** template.

   **Tip:** If you do not see the template in this list, select **Tools > Specific Setup > Reload**. If the template is still not visible, check that you have set the report template directory.

![Select template example](image.png)

**Figure 428: Select template example**

The **New Report from Template** wizard opens.
3. In the **Schema Associations** tab, select the **Schema Location** you want to use.
   
   **Note:** The schema data is not used in this example, but you must still select a schema location.
   
   **Note:** You do not need to select the repetition. The trigger is required for the template to work, but is not used in this example.

4. In the **Variable** tab, edit the StringUserName value to read Anne Brown and the NumericUserID to 2468.

5. Click **Finish**.

**Example 2: Template with data groups and repetitions**

In this example, you create a template with a trigger.

The triggers in the report structure specify what should be printed when a change in data occurs.

**Example 2: Create the files required for a "Triggers" report template**

Create the files to support the new "Triggers" report template. This template will include a repetition trigger.

**About this task**

You will create four files:

- Template schema definition file: Triggers.rsd.
- Report template file: Triggers.4rt.
- Properties file: Triggers.4rt.prop.
- Image file: Triggers.4rt.png.

**Before you begin:**

- Ensure that you have a directory on your disk (for example, MyReportTemplates) to hold your report template files. You may need to configure Genero Studio to recognize the directory.
- In Genero Studio, create the project (for example, MyTemplates.4pw) to hold the template files.

1. Create a template schema definition file (File > New > Reports > Template Schema Definition (.rsd)) and enter the following text:

   ```xml
   <?xml version="1.0" encoding="utf-8"?>
   <ReportSchema fileVersion="30000" gstVersion="30000"
   rootElementName="model">
   <Trigger name="TheTrigger" minOccurs="0" maxOccurs="unbounded"/>
   </ReportSchema>
   ```

   Save the template schema definition file as Triggers.rsd in your report template directory.

   This code creates a single trigger.

2. Create a report template file (File > New > Reports > Empty Report Template (.4rt)):
   
   a) In the Data View, click **Open Schema File** and select the Triggers.rsd schema.
   
   b) In the Report Structure, move the trigger under the Page Root.
   
   c) Add a Word Box as a child of the trigger, and edit the text to read "Repeat Me". The report template should look like **Figure 429: Report template example** on page 1008.
Save the report template file as Triggers.4rt in your report template directory.

3. Create a text file in the same directory as the template files. Save it as Triggers.4rt.prop and enter the following text:

```plaintext
tags: Triggers
label: Trigger Template
description: Template with a single trigger
```

4. Create an image file in the same directory as the template files. Save it as Triggers.4rt.png.

   **Note:** You can take a screen capture of the existing template, or copy an image from elsewhere.

**Example 2: Use the "Triggers" template to create a report**

Create a report using the new template.

1. Go to **File > New > Report from Template**.
2. In the **Filters** drop-down menu, select **Triggers**, and then select the **Trigger Template** template.

   **Tip:** If you do not see the template in this list, select **Tools > Specific Setup > Reload**. If the template is still not visible, check that you have set the report template directory.
3. In the Schema Associations tab, a Schema Location and ensure that the trigger is repeated on every row.
4. Click Finish.

**Example 3: Template with a dynamic list**

In this example, you create a template with a dynamic list.

The dynamic list contains a dynamic number of fields, for example, columns in a table. The number and data type of the fields is not defined in the template, but in the report when it is created. This example uses the template field and
template field trigger to produce the dynamic list for the template. You specify template triggers in the schema, and template fields in the template.

**Example 3: Create the files required for an "Items" report template**

Create the files to support the new "Items" report template. The template will include template field triggers and a template field.

**About this task**

You will create four files:

- Template schema definition file: `Items.rsd`
- Report template file: `Items.4rt`
- Properties file: `Items.4rt.prop`
- Image file: `Items.4rt.png`

**Before you begin:**

- Ensure that you have a directory on your disk (for example, `MyReportTemplates`) to hold your report template files. You may need to configure Genero Studio to recognize the directory.
- In Genero Studio, create the project (for example, `MyTemplates.4pw`) to hold the template files.

1. Create a template schema definition file (**File > New > Reports > Template Schema Definition (.rsd)**) and enter the following text:

   ```xml
   <?xml version="1.0" encoding="utf-8"?>
   <ReportSchema fileVersion="30000" gstVersion="30000"
    rootElementName="model">
    <TemplateFieldsTrigger groupName="items"/>
    <TemplateFieldsTrigger groupName="items"/>
    <TemplateFieldsTrigger groupName="moreItems"/>
    <Trigger name="requiredButNotUsed" minOccurs="0" maxOccurs="unbounded"/>
   </ReportSchema>
   ``

   **Note:** The trigger is required for the template to work, but in this case is not used.

   Save the template schema definition file as `Items.rsd` in your report template directory.

   For every named group, a template trigger (shown as a blue dot) is created in the template report structure. In this example, three template triggers are created (two for "items" and one for "moreItems"). If you want to include the same data twice (for example, adding the titles to the heading and the data to the list), you must create two groups with the same name.

   For every uniquely named group, a tab is created in the **New Report from Template** wizard. In this example, two tabs are created: for "items" and "moreItems".

2. Create a report template file (**File > New > Reports > Empty Report Template (.4rt)**):
   a) In the Data View, open the `Items.rsd` schema.
   b) Add a Horizontal Box (Mini Page), a Word Box, and two Template Fields. In the Report Structure, move the items to look like **Figure 431: Report template example** on page 1012.
Figure 431: Report template example

Save the report template file as `Items.4rt` in your report template directory.

3. Create a text file in the same directory as the template files. Save it as `Items.4rt.prop` and enter the following text:

```plaintext
tags: Items
label: Dynamic List template
description: Template with dynamic list
```

4. Create an image file in the same directory as the template files. Save it as `Items.4rt.png`.

   **Note:** You can take a screen capture of the existing template, or copy an image from elsewhere.

**Example 3: Use the "Items" template to create a report**

Create a report using the new template.

1. Go to **File > New > Report from Template**.
2. Select the **Items** template.
   
   **Tip:** If you do not see the template in this list, select **Tools > Specific Setup > Reload**. If the template is still not visible, check that you have set the report template directory.

   The **New Report from Template** wizard opens.

3. In the **Schema Associations** tab, select the **Schema Location** you want to use.

   **Note:** You do not need to select the repetition. The trigger is required for the template to work, but is not used in this example.

4. In the **Add Fields [items]** and **Add Fields [moreItems]** tabs, add as many fields as you want.

   **Note:** Add fields of different types, for example, at least one String and at least one Numeric type.

5. Click **Finish**.

Select the fields that have been created in the report and look at their properties, in particular:

- **Text** matches the selected field, for example, `data_orderline.item_itemid`.
- **Type** is specific to the data type, for example a Word Box for a string value.
- **Class** is also specific to the data type, for example `grwStringValue`. 
Report Writer Deployment and Customization

The Genero Report Engine (GRE) and Genero Report Viewer for HTML5 have limited configuration and customization options.

- GRE environment variables on page 1013
- Genero Report Viewer for HTML5 customization on page 1014
- Distributed Mode on page 1015

GRE environment variables

These environment variables are relevant for the Genero Report Engine (GRE).

- GREDIR on page 1013
- GREOUTPUTDIR on page 1013
- GRE_MAX_CONCURRENT_JOBS on page 1013
- GREDEBUG on page 1013
- GRE_DEFAULT_IMAGE_URL on page 1014

GREDIR

The GREDIR environment variable specifies the installation directory of the Genero Report Engine (GRE).

GREOUTPUTDIR

The GREOUTPUTDIR environment variable specifies the directory where all writing operations that would otherwise be performed in the current working directory of the Genero Report Engine (GRE) will be performed.

The value can be an absolute or a relative file path. If the value is a relative file path, then it is internally prepended with the current working directory of GRE. When set, all writing operations that would otherwise be performed in the current working directory of GRE will be performed in the specified directory.

Examples include relative path specification in the functions fgl_report_setOutputFileName(), fgl_report_configureImageDevice() (the filePath parameter), fgl_report_setBrowserDocumentDirectory(), fgl_report_setBrowserFontDirectory() and the location of the GRE debug files (jdebug0.xml, jdebug1.xml, and so on.)

GRE_MAX_CONCURRENT_JOBS

The GRE_MAX_CONCURRENT_JOBS environment variable limits the number of worker threads in distributed mode.

By default, the Genero Report Engine runs 25 concurrent threads. As such, by default it can process no more than 25 concurrent jobs. Jobs that are started at a point in time when the maximum value has been reached are queued until another job completes. The limit on concurrent threads is to prevent memory exhaustion in times of critical load.

GRE_MAX_CONCURRENT_JOBS allows the modification of the number of concurrent threads. This variable takes an integer.

GREDEBUG

The GREDEBUG environment variable defines the debug level for the Genero Report Engine.

Valid values include:
- 0: nothing
- 1: fatal only
- 2: fatal and error
- 3: fatal, error, and warning
- 4: fatal, error, warning, and info
- 5: fatal, error, warning, info, and debug
- 6: fatal, error, warning, info, debug, and trace
- 6+: all debug information
The files are written according to the value of the GREOUTPUTDIR. If GREOUTPUTDIR is not set, they are written to the current working directory of the Genero Report Engine (GRE).

**GRE_DEFAULT_IMAGE_URL**
The **GRE_DEFAULT_IMAGE_URL** environment variable defines the URL of the fallback image for an Image Box in a report design document.

The *fallback image* is the image to display if the requested image for an Image Box is not found.

The URL can be an absolute or relative URL. If it is a relative URL, the URL is resolved relative to the location of the form design (.drp) document.

This environment variable is specific to Genero Report Writer.

**Genero Report Viewer for HTML5 customization**
When you select Browser as the output option, files are created for the report. These files are viewed using the Genero Report Viewer for HTML5.

All files needed to operate the Genero Report Viewer for HTML5 are included in the Genero Report Engine (GRE) package in the directory $GREDIR/viewer.

**Customizing the viewer**
The Genero Report Viewer for HTML5 is comprised of the files found within $GREDIR/viewer.

**Table 270: GRV for HTML5 viewer files and directories**

<table>
<thead>
<tr>
<th>File / Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>viewer.html</td>
<td>Main HTML file containing the toolbar and document view. Customizable via CSS. Can be replaced by custom version if the same classes, ids and event bindings are used. Loads viewer.js, model.js, bowser.js and styles.css.</td>
</tr>
<tr>
<td>print.html</td>
<td>Print preview page. Customizable via CSS. Can be replaced by a custom version if the same classes, ids and event bindings are used. Loads print.js, model.js, bowser.js and styles.css.</td>
</tr>
<tr>
<td>viewer.js</td>
<td>JavaScript file that connects the HTML elements in the viewer (viewer.html by default) with the model (model.js). Can be replaced for the case that a custom HTML viewer is used that does not use the same classes, ids and event bindings as the default viewer.</td>
</tr>
<tr>
<td>print.js</td>
<td>JavaScript file that connects the HTML elements in the print previewer (print.html by default) with the model (model.js). Can be replaced for the case that a custom HTML print previewer is used that does not use the same classes, ids and event bindings as the default print previewer.</td>
</tr>
<tr>
<td>model.js</td>
<td>A JavaScript object representing a report. Methods exist to query the document (for example, to get the total number of pages), to navigate in the document (for example, to make a certain page the current page) and to register for events (for example, to ask to be notified when a new page is created).</td>
</tr>
<tr>
<td>bowser.js</td>
<td>A third-party utility for managing browser-specific issues.</td>
</tr>
<tr>
<td>styles.css</td>
<td>The default styles used by viewer.html and print.html.</td>
</tr>
<tr>
<td>images</td>
<td>A directory containing the images referenced from viewer.html and print.html.</td>
</tr>
</tbody>
</table>

**Related concepts**
Genero Report Viewer for HTML5 on page 655
If you choose **Browser** as the output format and **Preview** as the output action, the report displays in the Genero Report Viewer for HTML5.

**Distributed Mode**

The report engine can be started as a daemon to which report applications can connect to process reports. One or more engines can be started on the same machine or on different machines, hence *distributed mode*.

Distributed mode offers the following advantages:

- Faster processing for short documents.
  
The startup time of the JVM that is incurred for every report in regular mode can exceed the processing time of the report, causing the overall performance to be poor. In distributed mode, the JVM is started and initialized only once. For report batches, the improvement in performance can be dramatic; for example, in one test a 500 file PDF batch completed in 17 seconds rather than in 7 minutes.

- Improved scalability

  Formatting graphical reports is CPU intensive. However CPU is usually expensive on the server optimized for IO that is running the DVM and/or the database. The distributed mode allows offloading the report processing entirely to another, very much cheaper, machine such as a standard PC. Such a dedicated report formatting PC could be installed with Windows™ which has the additional advantage of handling fonts and printers in a user friendlier way.

- Ability to have the GRE and application on separate machines

  Perhaps your IT policy requires the GRE sit on a separate machine, or perhaps the required font sits on a specific machine.

**Take advantage of distributed mode**

To take advantage of distributed mode:

1. Place your report design documents (.4rp) on the server where the GRE daemon runs. If your design document references images using "file: url", place these files on the same server. See [Place report design documents on the daemon server](#).
2. Edit the source code to reflect that distributed mode is being used. This includes providing remote connection details, handle errors, and specify the location for finished files. See [Connect to a daemon using the reporting API](#) and [Logging in distributed mode](#).
3. Start the GRE daemon. See [Start the Genero Report Engine daemon](#).

**Place report design documents on the daemon server**

Place your report design documents (.4rp) on the server running the Genero Report Engine daemon. In addition to the report design documents, ensure that the fonts, images, and printers required for the reports are available. See [Configure fonts](#).

**Start the Genero Report Engine daemon**

The daemon is invoked by calling the script `$GREDIR/bin/greportwriter` with the “–l” option and specifying the port it listens on. For example:

```
cd $GREDIR
./envgre
$GREDIR/bin/greportwriter -l 6500
```

If the Genero Report Engine daemon resides on a different machine than the DVM, you must start the daemon with the `-l portname` option. The `-l` option listens on the specified port and for each connection attempts to read an XML document from the socket. For example:

```
$GREDIR/bin/greportwriter -l 6500
```
If the Genero Report Engine daemon resides on the same machine as the DVM, or if you are using SSH port forwarding, you can optionally use the \(-o\) option. The \(-o\) option provides extra security by ensuring that the daemon accepts incoming connections only if they are local. It must be used in conjunction with the \(-l\) option, for example:

\[
\$\text{GREDIR}/\text{bin/greportwriter} \ -l \ 5900 \ -o
\]

**Connect to a daemon using the reporting API**

The API function `fgl_report_configureDistributedProcessing` is used to select and configure distributed processing. It takes two parameters that denote the server to use; a host name and a port. This code shows an example for a GRE daemon running on the local machine (host "localhost"):

\[
\begin{align*}
\text{IF NOT } & \text{fgl_report_loadCurrentSettings("OrderReport.4rp") THEN} \\
& \ldots \\
\text{END IF} \\
& \ldots \\
\text{CALL fgl_report_configureDistributedProcessing("localhost",6500)} \\
& \text{RETURN fgl_report_commitCurrentSettings()}
\end{align*}
\]

This example connects to a daemon running on the server “PrintServer”:

\[
\begin{align*}
\text{IF NOT } & \text{fgl_report_loadCurrentSettings("OrderReport.4rp") THEN} \\
& \ldots \\
\text{END IF} \\
& \ldots \\
\text{CALL fgl_report_configureDistributedProcessing("PrintServer",6500)} \\
& \text{RETURN fgl_report_commitCurrentSettings()}
\end{align*}
\]

**Note:** If the server runs on a different physical machine with different resource paths to the DVM, you might need to configure the environment variables using the `fgl_report_configureDistributedEnvironment` function.

**Logging in distributed mode**

If the `GREDEBUG` environment variable is set, the daemon logs messages of the specified level to the file `gre.log` in the home directory of the user invoking the daemon.

**Important:** Sensitive and personal data may be written to the output. Make sure that the log output is written to files that can only be read by application administrators.

The API function `fgl_report_setDistributedRequestingUserName` can set a user name in the log file in order to distinguish between log entries originating from different users.

**Related concepts**

Tips for installing fonts on page 658

The best results are achieved if the fonts on the server where Genero Report Engine (GRE) is installed and those used in the Genero Report Designer (GRD) are the same. If these are on different systems, you may need to install fonts.

**Preview reports in distributed mode**

If you are running the GRE daemon on a different machine than the DVM, and the output type is either "PDF", "RTF", "XLS", "XLX" or "HTML" (as specified by the `fgl_report_selectDevice` function), you must set up the environment before you can preview reports.

**Important:** The following instructions are not necessary if the GRE and the DVM are running on the same physical machine.

1. Start the Web server service by invoking the command `grehttpd`.

   Located at `GREDIR/bin/grehttpd`, the following options are available:

   - \(-p \ \text{port}\), where `port` is the port number. The default port is 8080.
• \(-q\) activates the Web server service in quiet mode.
• \(-d\ directory\), where \(directory\) specifies the Web root. There may be several. If nothing is specified, then the current working directory ("." ) is taken as the Web root.
• \(-license\) prints the NanoHTTPD license.

Note: The Genero Report Engine ships with this minimal Web server. However, any other web server capable of serving static files can be used.

2. Set the \(GREOUTPUTDIR\) environment variable to the specified Web root directory in the environment of the GRE daemon (greportwriter -l).

Troubleshooting

You can determine the source of issues by examining the error messages produced by the Genero Report Engine. You can also report issues to support.

Report Writer error messages

A list of configuration error messages. For messages that are not self-explanatory, additional information is provided.

**Table 271: Report Writer error messages**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-18001</td>
<td>file not found</td>
</tr>
<tr>
<td>GS-18002</td>
<td>Data schema is missing, specify (-data &lt;path&gt;) argument</td>
</tr>
<tr>
<td>GS-18003</td>
<td>Need one and only one root in data schema</td>
</tr>
</tbody>
</table>

Genero Report Engine error messages

A list of Genero Report Engine (GRE) error messages. For messages that are not self-explanatory, additional information is provided.

**Table 272: Error messages for the Package designtime.trigger**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
</table>
| GS-37400 | Missing "name" attribute on "input-variable" element  
Error message and resolution should be self-explanatory. |
| GS-37401 | Missing "type" attribute on "input-variable" element  
Error message and resolution should be self-explanatory. |
| GS-37402 | Unrecognized "type" value "%1" on "input-variable" element  
Error message and resolution should be self-explanatory. |
| GS-37403 | Missing "expectedLocation" attribute on "input-variable" element  
Error message and resolution should be self-explanatory. |
| GS-37404 | Unrecognized "expectedLocation" value "%1" on "input-variable" element  
Error message and resolution should be self-explanatory. |
| GS-37405 | Missing "name" attribute on "match" element  
Error message and resolution should be self-explanatory. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-37406</td>
<td>Missing &quot;nameConstraint&quot; attribute on &quot;match&quot; element</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37407</td>
<td>Missing &quot;name&quot; attribute on &quot;Collection&quot; element</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37409</td>
<td>Missing &quot;name&quot; attribute on &quot;Field&quot; element</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37410</td>
<td>Missing &quot;type&quot; attribute on &quot;Field&quot; element</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37411</td>
<td>Unrecognized &quot;type&quot; value &quot;%1&quot; on &quot;Field&quot; element</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
</tbody>
</table>

Table 273: Error messages for the Package designtime

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-37200</td>
<td>Failed to find PXML node in template</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37206</td>
<td>Failed to find RTL &quot;stylesheet&quot; element in template</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37207</td>
<td>Failed to find element named &quot;%1&quot; in schema</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37208</td>
<td>Failed to find match node named &quot;%1&quot; in template</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37209</td>
<td>Failed to find root element &quot;%1&quot; in schema file &quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37210</td>
<td>Failed to find RTL root match node in template</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37211</td>
<td>Failed to find root element &quot;%1&quot; in rdd schema file &quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37212</td>
<td>Missing &quot;schemaFileName&quot; attribute in ReportSchemaTransform element at %1</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37213</td>
<td>Missing &quot;collectionName&quot; attribute in CollectionWrapper element at %1</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-37214</td>
<td>Missing &quot;itemToShift&quot; attribute in SiblingShifter element at %1 Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37215</td>
<td>Missing &quot;insertPoints&quot; attribute in SiblingShifter element at %1 Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37216</td>
<td>Failed to find item &quot;%1&quot; for shifting specified in SiblingShifter element at %2 Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37217</td>
<td>Missing &quot;collectionGroupName&quot; attribute in CollectionGrouper element at %1 Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37218</td>
<td>Missing &quot;groupingExpression&quot; attribute in CollectionGrouper element at %1 Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37219</td>
<td>Failed to parse &quot;placementLocation&quot; value &quot;%1&quot; as integer in CollectionAggregator element at %2 Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37220</td>
<td>Failed to find trigger path &quot;%1&quot; For more information, see Triggers on page 791.</td>
</tr>
<tr>
<td>GS-37221</td>
<td>Failed to find trigger &quot;%1&quot; in parent trigger &quot;%2&quot; For more information, see Triggers on page 791.</td>
</tr>
<tr>
<td>GS-37222</td>
<td>Missing &quot;collectionItemName&quot; attribute in element at %1 Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-37223</td>
<td>Missing &quot;collectionGroupName&quot; attribute in CollectionGrouper element at %1 Error message and resolution should be self-explanatory.</td>
</tr>
</tbody>
</table>

Table 274: Error messages for the Package layoutnode

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-31200</td>
<td>Internal error: Invalid bar code type Contact support.</td>
</tr>
<tr>
<td>GS-31201</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. Value is not an integer Error message and resolution should be self-explanatory.</td>
</tr>
</tbody>
</table>

**Tip:** This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-31202</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The value needs to be %3 digits long</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31203</td>
<td>Bar code type &quot;%1&quot;: invalid checksum %2 in %3. Value should be %4</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31204</td>
<td>DocumentStructureException: failed to add text node (cause=%1).</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31205</td>
<td>DocumentStructureException: failed to finalize text node (cause= %1).</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31206</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. Value is not an integer</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31207</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The value needs to be %3 digits long</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31208</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. First digit must be either '0' or '1'</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31209</td>
<td>Internal error: Wrong initializer for digit &quot;%1&quot;</td>
</tr>
<tr>
<td>GS-31210</td>
<td>Bar code type &quot;Code128&quot;: invalid codeValue &quot;%1&quot;. Value cannot be parsed as comma separated character list</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31211</td>
<td>Bar code type &quot;Code128&quot;: invalid start codeValue &quot;%1&quot;. Sequence must start with STARTA, STARTB or STARTC</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31212</td>
<td>Bar code type &quot;%1&quot;: invalid checksum &quot;%2&quot; in %3. Value should be &quot;%4&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31213</td>
<td>Bar code type &quot;Code128&quot;: invalid character &quot;%1&quot; in codeValue &quot;%2&quot;. Code is not available in the character set %3.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31214</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. Value is not an integer</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31215</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The value needs to be %3 digits long</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31216</td>
<td>Bar code type &quot;%1&quot;: invalid checksum %2 in %3. Value should be %4</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31217</td>
<td>Bar code type &quot;%1&quot;: invalid number of digits %2. Value must be a multiple of 2.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31218</td>
<td>Bar code type &quot;%1&quot;: invalid value for noCheckDigits %2. Value must be either 1 or 2.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable inGenero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31219</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. Value may contain only the digits 0–9 and the ‘-’ character.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31220</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The value needs to be %3 digits long</td>
</tr>
<tr>
<td></td>
<td><strong>Error message and resolution should be self-explanatory.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31221</td>
<td>Bar code type &quot;%1&quot;: invalid K checksum %2 in %3. Value should be &quot;%4&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31222</td>
<td>Bar code type &quot;%1&quot;: invalid C checksum %2 in %3. Value should be '%4'</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in</td>
</tr>
<tr>
<td></td>
<td>Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31224</td>
<td>Bar code type &quot;%1&quot;: invalid character '%2' in codeValue &quot;%3&quot;.</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in</td>
</tr>
<tr>
<td></td>
<td>Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31225</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The value needs to be %3 digits long</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in</td>
</tr>
<tr>
<td></td>
<td>Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31226</td>
<td>Bar code type &quot;%1&quot;: invalid checksum %2 in %3. Value should be %4</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in</td>
</tr>
<tr>
<td></td>
<td>Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31227</td>
<td>Bar code type &quot;Code 39 Extended&quot;: invalid character &quot;%1&quot; in codeValue &quot;%2&quot;.</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in</td>
</tr>
<tr>
<td></td>
<td>Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31228</td>
<td>Bar code type &quot;Code 39 Extended&quot;: invalid codeValue &quot;%1&quot;. Value cannot be parsed as comma separated</td>
</tr>
<tr>
<td></td>
<td>character list</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in</td>
</tr>
<tr>
<td></td>
<td>Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31229</td>
<td>Bar code type &quot;Code 39 Extended&quot;: invalid character &quot;%1&quot; in codeValue &quot;%2&quot;.</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in</td>
</tr>
<tr>
<td></td>
<td>Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31230</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. Value is not an integer</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in</td>
</tr>
<tr>
<td></td>
<td>Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31231</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The value needs to be 9 digits long</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in</td>
</tr>
<tr>
<td></td>
<td>Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31232</td>
<td>Bar code type &quot;%1&quot;: invalid checksum %2 in %3. Value should be %4</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in</td>
</tr>
<tr>
<td></td>
<td>Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31233</td>
<td>Internal error: Wrong initializer for digit &quot;%1&quot;</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31234</td>
<td>Bar code type &quot;%1&quot;: invalid value for controlCharacters %2. Value must be a four character string.</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31235</td>
<td>Bar code type &quot;%1&quot;: invalid value for noCheckDigits %2. Value must be either 1 or 2.</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31236</td>
<td>Bar code type &quot;%1&quot;: invalid character '%2' in codeValue &quot;%3&quot;.</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31237</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The value needs to be %3 digits long</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31238</td>
<td>Bar code type &quot;%1&quot;: invalid K checksum %2 in %3. Value should be '%4'</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31239</td>
<td>Bar code type &quot;%1&quot;: invalid C checksum %2 in %3. Value should be '%4'</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31241</td>
<td>Bar code type &quot;Code 93 Extended&quot;: invalid codeValue &quot;%1&quot;. Value cannot be parsed as comma separated character list</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31242</td>
<td>Bar code type &quot;Code 93 Extended&quot;: invalid character &quot;%1&quot; in codeValue &quot;%2&quot;.</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31243</td>
<td>Bar code type &quot;%1&quot;: invalid character '%2' in codeValue &quot;%3&quot;.</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31244</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The value needs to be %3 digits long</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31245</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The first character of the value needs to be either 'a', 'b', 'c' or 'd'&lt;br&gt;&lt;br&gt;<strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31246</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The last character of the value needs to be either 't', 'n', '*' or 'e'&lt;br&gt;&lt;br&gt;<strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31247</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The character '%3' may not be used within the value&lt;br&gt;&lt;br&gt;<strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31248</td>
<td>Bar code type &quot;%1&quot;: invalid checksum %2 in %3. Value should be '%4'&lt;br&gt;&lt;br&gt;<strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31249</td>
<td>Internal error: Codabar 18: invalid check type</td>
</tr>
<tr>
<td>GS-31250</td>
<td>Bar code type &quot;%1&quot;: invalid codeValue &quot;%2&quot;. The last character of the value needs to be either 'a', 'b', 'c' or 'd'&lt;br&gt;&lt;br&gt;<strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31251</td>
<td>Internal error: Codabar 2: invalid check type</td>
</tr>
<tr>
<td>GS-31252</td>
<td>Bar code type &quot;%1&quot;: expected text to follow '\' at end of &quot;codeValue&quot; attribute&lt;br&gt;&lt;br&gt;<strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31253</td>
<td>Bar code type &quot;%1&quot;: expected three octal digits to follow '\' in &quot;codeValue&quot; attribute at position %2&lt;br&gt;&lt;br&gt;<strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31254</td>
<td>Bar code type &quot;%1&quot;: expected octal digit in &quot;codeValue&quot; attribute at position %2&lt;br&gt;&lt;br&gt;<strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31257</td>
<td>Bar code type &quot;%1&quot;: code data too long (val=%2), max=928&lt;br&gt;&lt;br&gt;<strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31258</td>
<td>Bar code type &quot;%1&quot;: invalid errorCorrectionDegree &quot;%2&quot;. Value must be between 0 and 8.&lt;br&gt;&lt;br&gt;<strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31259</td>
<td>Bar code type &quot;%1&quot;: invalid value for lines &quot;%2&quot;. Value must be a value between 1 and 90.</td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31260</td>
<td>Bar code type &quot;%1&quot;: invalid value for lines &quot;%2&quot;. Value causes illegal value for dataSymbolsPerLine of %3 which must be a value between 1 and 30.</td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31261</td>
<td>Bar code type &quot;%1&quot;: invalid dataSymbolsPerLine &quot;%2&quot;. Value must be between 1 and 30.</td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31262</td>
<td>Bar code type &quot;%1&quot;: invalid value for dataSymbolsPerLine &quot;%2&quot;. Value causes illegal value for lines of %3 which must be a value between 1 and 90.</td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31264</td>
<td>Bar code type &quot;%1&quot;: value for lines &quot;%2&quot; and dataSymbolsPerLine &quot;%3&quot; is too small to hold data of %4. The product must be at least %5</td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31265</td>
<td>Bar code type &quot;%1&quot;: value for additionalPaddingLines &quot;%2&quot; causes the data lines to exceed the maximum value of 90 lines</td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31266</td>
<td>Bar code type &quot;%1&quot;: encoding error: data needs to end with a latch to ASCII encoding</td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31267</td>
<td>Bar code type &quot;data-matrix&quot;: invalid rawCodeValue &quot;%1&quot;. Value cannot be parsed as comma separated character list</td>
</tr>
<tr>
<td></td>
<td>For more information, see data-matrix bar code type.</td>
</tr>
</tbody>
</table>

GS-31268  Character '%1' cannot be encoded since it is not available in the code page ISO-8859-1.                                                                                                           Error message and resolution should be self-explanatory.                                               |

GS-31269  Datamatrix "Shifted ASCII" encoding: Character '%1' not allowed.                                                                                                                               For more information, see data-matrix bar code type.                                                    |

GS-31270  Datamatrix "Basic ASCII" encoding: Character '%1' not allowed.                                                                                                                                For more information, see data-matrix bar code type.                                                    |
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-31271</td>
<td>Datamatrix &quot;Shifted ASCII&quot; encoding: Unknown encoding symbol &quot;%1&quot;. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31272</td>
<td>Datamatrix &quot;Basic ASCII&quot; encoding: Unknown encoding symbol &quot;%1&quot;. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31273</td>
<td>Datamatrix &quot;C40 (basic set)&quot; encoding: Character '%1' not allowed. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31274</td>
<td>Datamatrix &quot;C40 (basic set)&quot; encoding: Illegal position of &quot;STARTASCII&quot; symbol. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31275</td>
<td>Datamatrix &quot;C40 (basic set)&quot; encoding: Unknown encoding symbol &quot;%1&quot;. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31276</td>
<td>Datamatrix &quot;C40 (set 1)&quot; encoding: Unknown encoding symbol &quot;%1&quot;. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31277</td>
<td>Datamatrix &quot;C40 (set 2)&quot; encoding: Character '%1' not allowed. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31278</td>
<td>Datamatrix &quot;C40 (set 2)&quot; encoding: Unknown encoding symbol &quot;%1&quot;. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31279</td>
<td>Datamatrix &quot;C40 (set 3)&quot; encoding: Character '%1' not allowed. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31280</td>
<td>Datamatrix &quot;C40 (set 3)&quot; encoding: Unknown encoding symbol &quot;%1&quot;. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31281</td>
<td>Datamatrix &quot;C40&quot; encoding: Encountered non data character as last character at the end of symbol space. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31282</td>
<td>Datamatrix &quot;C40&quot; encoding: Encoded data exceeds maximum of 1558 bytes. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31284</td>
<td>Datamatrix &quot;C40&quot; encoding: Encountered non data character as last character at the end of symbol space. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31285</td>
<td>Datamatrix &quot;BASE 256&quot; encoding: failed to parse &quot;%1&quot; as byte value. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31286</td>
<td>Datamatrix &quot;BASE 256&quot; encoding: failed to parse &quot;%1&quot; as byte value (value is outside of the range 0-255). For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31287</td>
<td>Datamatrix &quot;ANSI X11&quot; encoding: Character '%1' not allowed. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31288</td>
<td>Datamatrix &quot;ANSI X11&quot; encoding: Illegal position of &quot;STARTASCII&quot; symbol. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31289</td>
<td>Datamatrix &quot;ANSI X12&quot; encoding: Unknown encoding symbol &quot;%1&quot;. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31290</td>
<td>Datamatrix &quot;ANSI X12&quot; encoding: Invalid number of encoded codewords (%1). The number must be a multiple of 3 For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31291</td>
<td>Datamatrix &quot;EDIFACT&quot; encoding: Character '%1' not allowed. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31292</td>
<td>Datamatrix &quot;EDIFACT&quot; encoding: Illegal position of &quot;STARTASCII&quot; symbol. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31293</td>
<td>Datamatrix &quot;EDIFACT&quot; encoding: Unknown encoding symbol &quot;%1&quot;. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31294</td>
<td>Datamatrix &quot;EDIFACT&quot; encoding: Invalid number of encoded codewords (%1). The number must either be a multiple of 4 or terminated by a STARTASCII latch. For more information, see data-matrix bar code type.</td>
</tr>
<tr>
<td>GS-31295</td>
<td>Code value too long</td>
</tr>
<tr>
<td>GS-31296</td>
<td>Invalid number of arguments in gradient paint specification (Required are x1, y1, color1, x2, y2, color2, cyclic)</td>
</tr>
<tr>
<td>GS-31297</td>
<td>Failed to parse PXML x expression value %1 from the gradientPaint initializer %2. Parse Error:%3</td>
</tr>
</tbody>
</table>

**Tip:** This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-31298</td>
<td>Failed to parse PXML y expression value %1 from the gradientPaint initializer %2. Parse Error:%3</td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31299</td>
<td>Failed to parse PXML color value %1 from the gradientPaint initializer %2. Parse Error:%3</td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31300</td>
<td>Failed to parse boolean value &quot;cyclic&quot; from the gradientPaint initializer %1. Parse Error:%2</td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31301</td>
<td>Invalid command &quot;%1&quot; in path &quot;%2&quot;</td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31302</td>
<td>Unknown command &quot;%1&quot; in path &quot;%2&quot;</td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31303</td>
<td>Number of items in xyList not a multiple of 2 (value=%1 count=%2</td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-31304</td>
<td>Must have same number of values as row titles multiplied by column titles. Have %1 values and %2 values</td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-31305</td>
<td>Internal error: Invalid chart type</td>
</tr>
<tr>
<td>GS-31306</td>
<td>Syntax error int PXML font size expression (%1)</td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31307</td>
<td>Failed to parse PXML fontsize expression &quot;%1&quot;</td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31308</td>
<td>Decimal format is not available for this input locale</td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31309</td>
<td>Decimal format is not available for this output locale</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31310</td>
<td>Failed to create HTML view (cause=%1;%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31311</td>
<td>Failed to load image from data (cause=%1)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31312</td>
<td>Unsupported image format in data URL &quot;%1&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31313</td>
<td>Unsupported image format loading image from URL &quot;%1&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31314</td>
<td>Failed to load SVG document from file &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31315</td>
<td>Failed to load image from file &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31316</td>
<td>Unsupported image format loading image from file &quot;%1&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31317</td>
<td>Failed to load image (cause=&quot;%1&quot;)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31318</td>
<td>Can't find resource &quot;%1&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31319</td>
<td>Failed to load SVG document from resource URL &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31320</td>
<td>Failed to load image from resource URL &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31321</td>
<td>Failed to load &quot;Base-64&quot; encoded SVG document from data URL &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31322</td>
<td>Failed to load SVG document from data URL &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31323</td>
<td>Syntax error in data URL &quot;%1&quot; (Failed to locate delimiter ',')</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31324</td>
<td>Unsupported encoding in data URL &quot;%1&quot; (Failed to find &quot;;base64&quot;)</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31325</td>
<td>Incomplete data URL &quot;%1&quot; (Data is missing)</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31326</td>
<td>Failed to load image from data URL &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31327</td>
<td>Failed to load image from URL &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31328</td>
<td>Internal error: GrabPrintable::getFontRenderContext() returned null</td>
</tr>
<tr>
<td>GS-31329</td>
<td>Invalid &quot;anchorX&quot; PXML expression value %1 computed from expression &quot;%1&quot; (must yield a value between 0 and 1))</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31330</td>
<td>Invalid &quot;anchorY&quot; PXML expression value %1 computed from expression &quot;%1&quot; (must yield a value between 0 and 1))</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31331</td>
<td>Invalid font style value &quot;%1&quot; in list &quot;%2&quot; (must be one of: &quot;plain&quot;, &quot;italic&quot; or &quot;bold&quot;)</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31332</td>
<td>Failed to parse font size percent expression %1 (cause=%2)</td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-3133</td>
<td>The empty string is not allowed as PXML color expression</td>
</tr>
<tr>
<td>GS-3134</td>
<td>Failed to parse PXML color expression &quot;%1&quot; (must be a '#' followed by an integer in hex notation) (cause=%2)</td>
</tr>
<tr>
<td>GS-3135</td>
<td>Invalid port &quot;%1&quot; value</td>
</tr>
<tr>
<td>GS-3136</td>
<td>invalid value for lowWaterMark (must be integer &gt;= 0) val=%1</td>
</tr>
<tr>
<td>GS-3137</td>
<td>invalid value for highWaterMark (must be negative or greater than lowWaterMark) lowWaterMark=%1 highWaterMark=%2</td>
</tr>
<tr>
<td>GS-3138</td>
<td>SAXException: failed to initialize (cause=%1).</td>
</tr>
<tr>
<td>GS-3139</td>
<td>Timeout: Page was not consumed within %1 milliseconds</td>
</tr>
<tr>
<td>GS-3140</td>
<td>SAXException: failed to create page (cause=%1).</td>
</tr>
<tr>
<td>GS-3141</td>
<td>SAXException: failed to finish document (cause=%1).</td>
</tr>
<tr>
<td>GS-3142</td>
<td>Internal error: Page not in map</td>
</tr>
<tr>
<td>GS-3143</td>
<td>IOException: failed to transform memory queue into disk queue (cause=%1).</td>
</tr>
<tr>
<td>GS-3144</td>
<td>IOException: failed to write page to disk queue (cause=%1).</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31345</td>
<td>IOException: failed to flush queue to disk (cause=%1). <strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31346</td>
<td>Internal error: %1::eventInitializedChild() called for a cloned LayoutNode</td>
</tr>
<tr>
<td>GS-31347</td>
<td>Nodes attached to named ports may not be self placing <strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
</tbody>
</table>
| GS-31348 | All nodes attached to named ports must be specified prior to specifying any nodes for the primary port  
A port is the same as a section.  
A node is said to be attached to a named port in its parent if it has the Section property set.  
A node is said to be attached to the primary port if it does not have the Section property set.  
Node lists are ordered. If you look at the Report Structure view where nodes of the same parent are stacked vertically, a node is prior to another node when it is higher up.  
To solve this issue, reorganize the nodes such that all named nodes sit higher in the parent node than nodes attached to the primary port. **Tip:** This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document. |
<p>| GS-31349 | Internal error: cannot propagate, parent==null||parent not instanceof LayoutNode                                                                                                                                                                                                                                                      |
| GS-31350 | Internal error: have uninitializedChild when cloning                                                                                                                                                                                                                                                                                     |
| GS-31351 | Internal error: %1::attachPortedChild() invalid port. Port=%2                                                                                                                                                                                                                                                                              |
| GS-31352 | The &quot;roundingMode&quot; attribute is not supported on this platform (requires Java version 1.6 or higher). <strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document. |
| GS-31353 | Failed to parse value &quot;%1&quot; using the specified input format (cause=%2) <strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                 |
| GS-31354 | Failed to format field value &quot;%1&quot; using format &quot;%2&quot; <strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                 |
| GS-31355 | Invalid page number format &quot;%1&quot; (must be one of: &quot;arabic&quot;,&quot;lowerroman&quot;,&quot;roman&quot; or &quot;upperroman&quot;) <strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document. |</p>
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-31356</td>
<td>Illegal argument value &quot;%1&quot; as second parameter in call to the PXML function format(). Valid values are ARABIC, LOWERROMAN and UPPERROMAN. For further information, see Using the PXML expression language on page 854.</td>
</tr>
<tr>
<td>GS-31357</td>
<td>Internal error: getStringReturnValue called on function format. The function returns a string value</td>
</tr>
<tr>
<td>GS-31358</td>
<td>Internal error: getParameterType called on function %1. The function takes no parameters</td>
</tr>
<tr>
<td>GS-31359</td>
<td>Internal error: pushString called on function %1. The function takes no parameters</td>
</tr>
<tr>
<td>GS-31360</td>
<td>Internal error: pushNumeric called on function %1. The function takes no parameters</td>
</tr>
<tr>
<td>GS-31361</td>
<td>Internal error: getStringReturnValue called on function %1. The function returns a numeric value</td>
</tr>
<tr>
<td>GS-31362</td>
<td>Internal error: pushString called on function %1. The function takes only one string parameter</td>
</tr>
<tr>
<td>GS-31363</td>
<td>Internal error: getStringReturnValue called on function %1. The function returns a numeric value</td>
</tr>
<tr>
<td>GS-31364</td>
<td>Number of sizes and titles differ (Have %1 sizes and %2 titles) Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31365</td>
<td>Internal error: Invalid pie graph type</td>
</tr>
<tr>
<td>GS-31366</td>
<td>Internal error: %1::eventNewChild() called for a cloned LayoutNode</td>
</tr>
<tr>
<td>GS-31367</td>
<td>Internal error: %1::eventInitializedChild() called for a cloned LayoutNode</td>
</tr>
<tr>
<td>GS-31368</td>
<td>Internal error: cannot propagate, parent==null</td>
</tr>
<tr>
<td>GS-31369</td>
<td>Internal error: have unitializedChild when cloning</td>
</tr>
<tr>
<td>GS-31370</td>
<td>Loading SVG document from url %1 failed (cause=%2)</td>
</tr>
<tr>
<td>GS-31371</td>
<td>Loading SVG document from reader failed (cause=%1)</td>
</tr>
<tr>
<td>GS-31372</td>
<td>Loading SVG document from input stream failed (cause=%1)</td>
</tr>
<tr>
<td>GS-31373</td>
<td>Invalid command in transformInstructions attribute (&quot;%1&quot;) Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31374</td>
<td>not enough arguments to %1 command in transform attribute. Required are %2</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31375</td>
<td>Wrong number of matrix values in %1 attribute (%2) required are 6</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31376</td>
<td>Internal error: %1 expression=%2</td>
</tr>
<tr>
<td>GS-31377</td>
<td>Indentation value exceeds width by %1</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31378</td>
<td>Internal error: No parent</td>
</tr>
<tr>
<td>GS-31379</td>
<td>Internal error: Parent not of type LayoutNode (type=%1)</td>
</tr>
<tr>
<td>GS-31382</td>
<td>The number of items in all series must be the same for the</td>
</tr>
<tr>
<td></td>
<td>&quot;StackedArea&quot; type.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31383</td>
<td>The x values of different series need to be the same for the &quot;StackedArea&quot;</td>
</tr>
<tr>
<td></td>
<td>type.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31384</td>
<td>The number of xValues and yValues differ (Have %1 xValues and %2 yValues)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31385</td>
<td>When seriesTitles are specified, they must have same number as xValues and</td>
</tr>
<tr>
<td></td>
<td>yValues. Have %1 xValues and %2 seriesTitles</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31386</td>
<td>Internal error: Invalid code type for XYGraph</td>
</tr>
<tr>
<td>GS-31388</td>
<td>Internal error: UnsupportedEncodingException &quot;%1&quot; for value &quot;%2&quot; using</td>
</tr>
<tr>
<td></td>
<td>encoding &quot;%3&quot;</td>
</tr>
<tr>
<td>GS-31389</td>
<td>QR-Code: encoding exception &quot;%1&quot; on encoding value &quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31391</td>
<td>QR-Code: Unsupported encoding &quot;%1&quot; (Run &quot;CharsetInfo&quot; for a list of</td>
</tr>
<tr>
<td></td>
<td>supported encodings)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31392</td>
<td>QR-Code: Invalid the error correction degree value %1 (must be a</td>
</tr>
<tr>
<td></td>
<td>value between 0 and 3)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31393</td>
<td>Invalid font size value %1 (value needs to be to be greater than 0)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31394</td>
<td>Error parsing hex digit (0) at position %2 in string &quot;)(2)&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31395</td>
<td>Parse error: Failed to parse tracking code and routing code from code</td>
</tr>
<tr>
<td></td>
<td>value &quot;/%1&quot; (Found more than one comma)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31396</td>
<td>Parse error: Failed to parse tracking code and routing code from code</td>
</tr>
<tr>
<td></td>
<td>value &quot;/%1&quot; (Encountered non digit character (1))</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31397</td>
<td>Parse error: routing code &quot;/%1&quot; exceeds 11 digits</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31398</td>
<td>Parse error: Tracking code &quot;/%1&quot; is too short (needs to have at least 2</td>
</tr>
<tr>
<td></td>
<td>digits)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31399</td>
<td>Parse error: The second digit of the tracking code (0) is outside the</td>
</tr>
<tr>
<td></td>
<td>allowed range 0 - 4 in tracking code &quot;/%2&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31400</td>
<td>Nodes attached to the last page footer port must be specified last</td>
</tr>
<tr>
<td></td>
<td>(After all other nodes including nodes for the primary port)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31401</td>
<td>Failed to load PDF document from file &quot;/%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31402</td>
<td>Failed to load PDF document from resource URL &quot;/%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-31403</td>
<td>Failed to load &quot;Base-64&quot; encoded PDF document from data URL &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31404</td>
<td>Loading PDF document from url %1 failed (cause=%2)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-31405</td>
<td>Loading PDF document failed (cause=%1)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-31406</td>
<td>Failed to load &quot;Base-64&quot; encoded PDF document from data URL &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-31407</td>
<td>Requested page index %1 is out of the range 0..%2.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
</tbody>
</table>

**Table 275: Error messages for the Package main**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-32100</td>
<td>Instantiating the Viewer failed with a TargetInvokationException (cause=%1:%2)</td>
</tr>
<tr>
<td></td>
<td>For further information, see What is packaging? on page 1173</td>
</tr>
</tbody>
</table>

**Table 276: Error messages for the Package shared**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-32700</td>
<td>attribute %1 must be specified</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32701</td>
<td>Invalid value &quot;%1&quot; (must be integer)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32702</td>
<td>Invalid value &quot;%1&quot; (must be a floating point value)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-32703</td>
<td>Invalid value &quot;%1&quot; (must be boolean)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32704</td>
<td>Invalid enumeration value &quot;%1&quot; (must be one of: %2)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32705</td>
<td>Invalid option value &quot;%1&quot; (must be one of: %2)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32706</td>
<td>Required option value (must be one of: %1)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32707</td>
<td>Invalid mandatory option value &quot;%1&quot; (must be one of: %2)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32708</td>
<td>Invalid boolean option value &quot;%1&quot; (must be one of: %2)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32709</td>
<td>Required boolean option value (must be one of: %1)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32710</td>
<td>Invalid mandatory boolean option value &quot;%1&quot; (must be one of: %2)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32711</td>
<td>Invalid value &quot;%1&quot; (must be a colon separated list of float point values)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-32712</td>
<td>Invalid value &quot;%1&quot; (must be a colon separated list of POSITIVE float point values)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32713</td>
<td>Invalid value &quot;%1&quot; (must be a colon separated list of integer values)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32714</td>
<td>Invalid object option value &quot;%1&quot; (must be one of: %2)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32715</td>
<td>Required object option value (must be one of: %1)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32716</td>
<td>Invalid mandatory object option value &quot;%1&quot; (must be one of: %2)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32717</td>
<td>Invalid value &quot;%1&quot; (must be a colon separated list of strings): unexpected character &quot;%2&quot; at position %3. Expected a &quot; &quot; or &quot;,&quot;, character</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32718</td>
<td>Invalid value &quot;%1&quot; (must be a colon separated list of strings): encountered unclosed string starting a position %2.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-32719</td>
<td>Missing lower bound in range specification &quot;%1&quot; in expression &quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-32720</td>
<td>Failed to parse &quot;%1&quot; as number in expression in expression &quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-32721</td>
<td>Failed to parse &quot;%1&quot; as upper bound number in expression in expression &quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
</tbody>
</table>

**Table 277: Error messages for the Package stylesheet.rtl**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-33900</td>
<td>Unknown RTL variable type &quot;%1&quot;</td>
</tr>
<tr>
<td></td>
<td>For further information, see <strong>Using the expression language</strong> on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <strong>attribute error</strong>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33901</td>
<td>Failed to initialize numeric RTL attribute variable from the input string</td>
</tr>
<tr>
<td></td>
<td>&quot;%1&quot;</td>
</tr>
<tr>
<td></td>
<td>For further information, see <strong>Using the expression language</strong> on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <strong>attribute error</strong>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33902</td>
<td>Missing &quot;url&quot; attribute in RTL call-report element at %1</td>
</tr>
<tr>
<td></td>
<td>For further information, see <strong>Using the expression language</strong> on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <strong>attribute error</strong>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33903</td>
<td>Missing RTL match element at %1</td>
</tr>
<tr>
<td></td>
<td>For further information, see <strong>Using the expression language</strong> on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <strong>attribute error</strong>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33904</td>
<td>Missing RTL &quot;name&quot; attribute in entity declaration at %1</td>
</tr>
<tr>
<td></td>
<td>For further information, see <strong>Using the expression language</strong> on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <strong>attribute error</strong>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33905</td>
<td>Missing &quot;url&quot; attribute in RTL load-entities element at %1</td>
</tr>
<tr>
<td></td>
<td>For further information, see <strong>Using the expression language</strong> on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <strong>attribute error</strong>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33906</td>
<td>Missing &quot;name&quot; attribute in RTL expand-entity element at %1</td>
</tr>
<tr>
<td></td>
<td>For further information, see <strong>Using the expression language</strong> on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <strong>attribute error</strong>. Attribute errors are displayed in</td>
</tr>
<tr>
<td></td>
<td>a way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-33907</td>
<td>Reference to undeclared RTL entity &quot;%1&quot; at %2</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a</td>
</tr>
<tr>
<td></td>
<td>way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33908</td>
<td>Failed to replace RTL entity parameter &quot;%1&quot;. Parameter was not specified at</td>
</tr>
<tr>
<td></td>
<td>%2</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a</td>
</tr>
<tr>
<td></td>
<td>way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33909</td>
<td>Missing &quot;name&quot; attribute in RTL entity parameter element at %1</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a</td>
</tr>
<tr>
<td></td>
<td>way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33910</td>
<td>Missing &quot;value&quot; attribute in RTL entity parameter element at %1</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a</td>
</tr>
<tr>
<td></td>
<td>way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33911</td>
<td>Failed to find RTL style sheet root element at %1</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a</td>
</tr>
<tr>
<td></td>
<td>way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33912</td>
<td>Unknown RTL element %1 at %2</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a</td>
</tr>
<tr>
<td></td>
<td>way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33913</td>
<td>Encountered style sheet element with more than one RTL element at %1</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a</td>
</tr>
<tr>
<td></td>
<td>way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33914</td>
<td>Missing RTL elements in a sequences block at %1</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a</td>
</tr>
<tr>
<td></td>
<td>way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33915</td>
<td>Missing RTL alternatives in a or block at %1</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a</td>
</tr>
<tr>
<td></td>
<td>way that makes them clickable in Genero Studio, as they refer to a</td>
</tr>
<tr>
<td></td>
<td>particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| GS-33916 | Value for attribute %1 cannot be parsed as integer (Value=%2) at RTL element %3  
For further information, see Using the expression language on page 849.  
**Tip:** This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document. |
| GS-33917 | Invalid RTL attribute value selector %1 in attribute constraint %2 at %3  
For further information, see Using the expression language on page 849.  
**Tip:** This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document. |
| GS-33918 | Failed to resolve namespace prefix %1 in RTL constraint for %2 at %3  
For further information, see Using the expression language on page 849.  
**Tip:** This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document. |
| GS-33919 | Failed to parse regular expression for RTL attribute %1. Error=%2 at %3  
For further information, see Using the expression language on page 849.  
**Tip:** This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document. |
| GS-33920 | RTL attribute "name" not specified at "%1"  
For further information, see Using the expression language on page 849.  
**Tip:** This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document. |
| GS-33921 | RTL attribute "type" not specified at "%1"  
For further information, see Using the expression language on page 849.  
**Tip:** This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document. |
| GS-33922 | Illegal RTL location value "%1" at %2 (must be one of "expectedHere", "expectedBefore", "expectedAhead" or "expectedWayAhead"  
For further information, see Using the expression language on page 849.  
**Tip:** This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document. |
| GS-33923 | Error in input to RTL style: got token opening tag %1 when expecting %2  
For further information, see Using the expression language on page 849. |
| GS-33924 | Error in input to RTL style: got closing tag %1 when expecting %2  
For further information, see Using the expression language on page 849. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-33925</td>
<td>Expression error: %1 evaluating RTL expression &quot;%2&quot; at attribute &quot;%3&quot; at %4</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33926</td>
<td>Expression error: No variable value for variable &quot;%1&quot; encountered in</td>
</tr>
<tr>
<td></td>
<td>input stream when evaluating RTL expression &quot;%2&quot; at attribute &quot;%3&quot; at %4.</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33927</td>
<td>Expression error: %1 evaluating RTL expression &quot;%2&quot; at attribute &quot;%3&quot; at %4</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33928</td>
<td>Type error: Expression %1 yielded null instead of a %2 value at</td>
</tr>
<tr>
<td></td>
<td>RTL attribute %3 at %4.</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33929</td>
<td>Type error: Expression %1 yielded %2 instead of a %3 value at RTL attribute</td>
</tr>
<tr>
<td></td>
<td>%4 at %5.</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way</td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33930</td>
<td>Reference to unknown global RTL method &quot;$1&quot;</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33931</td>
<td>Unknown RTL method %1 in the current context of type %2</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33932</td>
<td>Unknown RTL method %1 on object of type %2</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33933</td>
<td>Wrong number of arguments supplied to method &quot;%1.substring&quot; (Expected 2</td>
</tr>
<tr>
<td></td>
<td>or 3, got %2)</td>
</tr>
<tr>
<td></td>
<td>For further information, see The String Class on page 975.</td>
</tr>
<tr>
<td>GS-33934</td>
<td>%1 to %2 conversion error in RTL function &quot;%3.toNum()&quot; (cannot parse</td>
</tr>
<tr>
<td></td>
<td>value &quot;%4&quot; as %5)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-33935</td>
<td>Illegal access error. Not able to call RTL method &quot;%1&quot; on object of type %2 (cause=%3)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33936</td>
<td>Error executing RTL method &quot;%1&quot; on object of type %2 (cause=%3)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33937</td>
<td>Wrong number of args supplied to RTL method &quot;%1.%2&quot; (Expected %3, got %4)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33938</td>
<td>Invalid type for argument no %1 in call to RTL method &quot;%2. %3&quot; (Expected %3, got %4)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33939</td>
<td>Invalid null argument for argument no %1 in call to RTL method &quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33940</td>
<td>Illegal access error. Not able to obtain value of RTL variable %1 on object of type %2 (cause=%3)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33941</td>
<td>Internal error: Attempt to reference an RTL array element on a null pointer (key=&quot;%1&quot;)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33942</td>
<td>Attempt to reference an RTL array element on a non array object of type %1 (key=&quot;%2&quot;)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33943</td>
<td>Internal RTL error: Can't find character context</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33944</td>
<td>RTL attribute &quot;name&quot; not specified in rtl:array element</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33945</td>
<td>RTL attribute &quot;key&quot; not specified in rtl:array element</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33946</td>
<td>RTL attribute &quot;name&quot; not specified in rtl:let element at %1</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33947</td>
<td>RTL attribute &quot;value&quot; not specified in rtl:let element at %1</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| GS-33948 | Undeclared RTL variable %1 at %2  
**Tip:** This is an *attribute error*. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                                                                                                                                                                               |
| GS-33949 | RTL attribute "name" not specified at %1  
**Tip:** This is an *attribute error*. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                                                                                                                                                                               |
| GS-33950 | Expression error "%1" retrieving value of RTL variable "%2" at %2  
**Tip:** This is an *attribute error*. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                                                                                                                                                                   |
| GS-33951 | Unknown RTL variable %1 at %2  
**Tip:** This is an *attribute error*. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                                                                                                                                                                                           |
| GS-33952 | RTL variable %1 in rtl:array-let element is not of type array but of type %2 at %3  
**Tip:** This is an *attribute error*. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                                                                                                                                                                   |
| GS-33953 | RTL attribute "key" not specified at %1  
**Tip:** This is an *attribute error*. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                                                                                                                                                                                           |
| GS-33954 | RTL expression "%1" in attribute "key" evaluated to null at %2  
**Tip:** This is an *attribute error*. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                                                                                                                                                                   |
| GS-33955 | RTL attribute "name" not specified at %1  
**Tip:** This is an *attribute error*. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                                                                                                                                                                                           |
| GS-33956 | RTL expression "%1" in attribute "key" evaluated to null at %2  
**Tip:** This is an *attribute error*. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                                                                                                                                                                   |
| GS-33957 | RTL attribute "name" not specified at %1  
**Tip:** This is an *attribute error*. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                                                                                                                                                                                           |
| GS-33958 | Internal RTL error: Failed to find "toCharacter"  
For further information, see [Using the expression language](#) on page 849.                                                                                                                                                                                                                                                                                                                                                       |
| GS-33959 | Encountered RTL "%1" element containing an invalid element "%2" at %3  
**Tip:** This is an *attribute error*. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.                                                                                                                                                                                                                                           |
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-33960</td>
<td>missing &quot;enumeration&quot; attribute value in RTL &quot;for-each&quot; element encountered at %1</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33961</td>
<td>Invalid object type %1 returned for RTL attribute &quot;enumeration&quot; in a &quot;for-each&quot; element. Expected expression &quot;%2&quot; to return enumeration type at %3</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33962</td>
<td>Missing RTL &quot;condition&quot; attribute value in &quot;while&quot; element encountered at %1</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33963</td>
<td>Internal RTL error: Order of calls violated</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33964</td>
<td>Illegal trailing '.' in RTL identifier</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33965</td>
<td>Redefinition of RTL identifier %1 of type %2 as record</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33966</td>
<td>%1 to %2 conversion error in RTL function &quot;%3.toNumeric()&quot; (cannot parse value &quot;%4&quot; as %5)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-33967</td>
<td>Date.fromString(): Failed to parse date value &quot;%1&quot; with pattern &quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td>For further information, see The Date Class on page 951.</td>
</tr>
<tr>
<td>GS-33968</td>
<td>Date.parseString(): Failed to parse date value &quot;%1&quot; with pattern &quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td>For further information, see The Date Class on page 951.</td>
</tr>
<tr>
<td>GS-33970</td>
<td>Missing &quot;schemaFileName&quot; attribute in ReportSchemaTransform element at %1</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33973</td>
<td>Unknown transformation processor name %1 at %2</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
</tbody>
</table>
Table 278: Error messages for the Package stylesheet.standardpipe

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-33800</td>
<td>Cannot open file</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33802</td>
<td>Old file format. Use last version of Genero Report Designer to update the file</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33803</td>
<td>File format : &quot; + version + &quot; not supported. This executable supports :&quot; + SUPPORTED_VERSION_T</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33804</td>
<td>logical page mapping is set to &quot;labels&quot; but fgl_report_configureLabelOutput() was not called to configure the layout</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33805</td>
<td>cannot determine label width. Value was not set in call to fgl_report_configureLabelOutput() and it is not set in the .4rp template.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33806</td>
<td>cannot determine label height. Value was not set in call to fgl_report_configureLabelOutput() and it is not set in the .4rp template.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33807</td>
<td>environment variable GREDIR is not set</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33808</td>
<td>environment variable FGLDIR is not set</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
</tbody>
</table>

Table 279: Error messages for the Package stylesheet

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-33000</td>
<td>%1: not prepared to handle nested style declarations</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>GS-33001</td>
<td>Failed to parse style attribute &quot;%1&quot;. Parse error=%2</td>
</tr>
<tr>
<td>GS-33002</td>
<td>Error parsing CSS stylesheet: %1</td>
</tr>
<tr>
<td>GS-33003</td>
<td>Internal error: Node %1 (%2) getIdValue() failed</td>
</tr>
<tr>
<td>GS-33004</td>
<td>Internal error: Invalid operator %1</td>
</tr>
<tr>
<td>GS-33005</td>
<td>Internal error: Invalid Id %1</td>
</tr>
<tr>
<td>GS-33006</td>
<td>Internal error: Invalid Id %1</td>
</tr>
<tr>
<td>GS-33007</td>
<td>Internal error: failed to find simple selector index</td>
</tr>
<tr>
<td>GS-33008</td>
<td>Internal error: garbled selector</td>
</tr>
<tr>
<td>GS-33009</td>
<td>error during printing: Exception: %1</td>
</tr>
<tr>
<td>GS-33010</td>
<td>Executing process &quot;%1&quot; returned non zero exit code: %2, Process output:%3</td>
</tr>
<tr>
<td>GS-33011</td>
<td>Got an exception of type &quot;%1&quot; when attempting to start process &quot;%2&quot; (cause=%3)</td>
</tr>
<tr>
<td>GS-33012</td>
<td>IOException writing to file &quot;%1&quot; (cause=&quot;%2&quot;)</td>
</tr>
<tr>
<td>GS-33013</td>
<td>Internal error: closing tag=&quot;%1&quot;</td>
</tr>
<tr>
<td>GS-33014</td>
<td>Internal error: Expected closing '%1' tag. Got &quot;%2&quot; tag instead</td>
</tr>
<tr>
<td>GS-33015</td>
<td>Please contact your sales office regarding licensing.</td>
</tr>
<tr>
<td>GS-33016</td>
<td>Missing COLDEF section in TABLE</td>
</tr>
</tbody>
</table>

For further information, see Table on page 867.
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-33017</td>
<td>Found illegal tag &quot;%1&quot; within COLDEFS section</td>
</tr>
<tr>
<td></td>
<td>For further information, see Table on page 867.</td>
</tr>
<tr>
<td>GS-33018</td>
<td>COLDEF element requires &quot;fWidth&quot; or &quot;pWidth&quot; attribute</td>
</tr>
<tr>
<td></td>
<td>For further information, see Table on page 867.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33019</td>
<td>COL references undeclared column in COLDEFS section</td>
</tr>
<tr>
<td></td>
<td>For further information, see Table on page 867.</td>
</tr>
<tr>
<td>GS-33020</td>
<td>Illegal align value &quot;%1&quot; (must be one of &quot;left&quot;, &quot;right&quot;, &quot;center&quot; or &quot;baseline&quot;)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33021</td>
<td>invalid value %1 for align attribute (must be one of &quot;left&quot;, &quot;right&quot; or &quot;center&quot;)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33022</td>
<td>LI element is not nested within a OL or UL element</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-33023</td>
<td>Encountered DIMENSIONS element at unexpected location within a PIVOTTABLE. DIMENSIONS need to be declared before any VALUES.</td>
</tr>
<tr>
<td></td>
<td>For further information, see Working with pivot tables on page 836.</td>
</tr>
<tr>
<td>GS-33024</td>
<td>Encountered a DATACOLUMN element for an undeclared column.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-33025</td>
<td>%1 element encountered at unexpected position.%2</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-33026</td>
<td>Failed to parse value &quot;%1&quot; as floating point value</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33027</td>
<td>Not enough dimensions in pivot table to draw as a map chart (Table needs to have at least one dimension column)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Working with pivot tables on page 836.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33028</td>
<td>Not enough values in pivot table to draw as a map chart (Table needs to have at least one value column)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Working with pivot tables on page 836.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-33029</td>
<td>Not enough dimensions in pivot table to draw as a map chart (Table needs to have at least two dimension columns)</td>
</tr>
<tr>
<td></td>
<td>For further information, see <a href="#">Working with pivot tables</a> on page 836.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33030</td>
<td>Not enough values in pivot table to draw as a map chart (Table needs to have at least one value column)</td>
</tr>
<tr>
<td></td>
<td>For further information, see <a href="#">Working with pivot tables</a> on page 836.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33031</td>
<td>Not enough values in pivot table to draw as a xy chart (Table needs to have at least two value columns)</td>
</tr>
<tr>
<td></td>
<td>For further information, see <a href="#">Working with pivot tables</a> on page 836.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33032</td>
<td>First value is not numeric (The first two values need to be numeric in order to draw a XY chart)</td>
</tr>
<tr>
<td></td>
<td>For further information, see <a href="#">XY charts</a> on page 823.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33033</td>
<td>Second value is not numeric (The first two values need to be numeric in order to draw a XY chart)</td>
</tr>
<tr>
<td></td>
<td>For further information, see <a href="#">XY charts</a> on page 823.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33034</td>
<td>Internal error: closing tag=&quot;%1&quot;</td>
</tr>
<tr>
<td>GS-33035</td>
<td>Internal error: Expected closing &quot;%1&quot; tag. Got &quot;%2&quot; tag instead</td>
</tr>
<tr>
<td>GS-33036</td>
<td>Internal error: Reference to unregistered color &quot;%1&quot;</td>
</tr>
<tr>
<td>GS-33037</td>
<td>Internal error: Reference to unregistered font &quot;%1&quot;</td>
</tr>
<tr>
<td>GS-33038</td>
<td>Incomplete document</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-33039</td>
<td><strong>IllegalAccessException:</strong> Cannot load class &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33040</td>
<td><strong>InstantiationException:</strong> Cannot load class &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-33041</td>
<td>NoSuchMethodException: Cannot load class &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33042</td>
<td>InvocationTargetException: Cannot load class &quot;%1&quot; (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33043</td>
<td>Internal error: IOException (cause=&quot;%1&quot;)</td>
</tr>
<tr>
<td>GS-33044</td>
<td>Internal error: parse called</td>
</tr>
<tr>
<td>GS-33045</td>
<td>Syntax error in PXML font size expression (%1) (Missing comma)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33046</td>
<td>Syntax error in PXML font size expression (%1) (can't parse &quot;line&quot; and &quot;column&quot; values as integers)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33047</td>
<td>Failed to parse font size percent expression %1 (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33048</td>
<td>Internal error: Could not find element %1 with source id %2 (name=%3), path=%4</td>
</tr>
<tr>
<td>GS-33049</td>
<td>Internal error: Could not find element %1 with source id %2 (name=%3), path=%4</td>
</tr>
<tr>
<td>GS-33050</td>
<td>Internal error: Failed to retrieve parent font size: (cause=%1)</td>
</tr>
<tr>
<td>GS-33051</td>
<td>Invalid port %1</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33052</td>
<td>Invalid page number format &quot;%1&quot;</td>
</tr>
<tr>
<td></td>
<td>For further information, see pageNoFormat property.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33053</td>
<td>Internal error: Got exception of type &quot;%1&quot; during creation of an hex encoded bitmap (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33054</td>
<td>COLDEF element requires &quot;fWidth&quot; or &quot;pWidth&quot; attribute</td>
</tr>
<tr>
<td></td>
<td>For further information, see Table on page 867.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33055</td>
<td>Internal error: failed to find child %1 for parent %2</td>
</tr>
</tbody>
</table>

For further information, see **Table** on page 867.
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Tip</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-33056</td>
<td>Internal error: node is of unexpected type %1</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33057</td>
<td>COL references undeclared column in COLDEFS section in row %1</td>
<td>For further information, see Table on page 867.</td>
</tr>
<tr>
<td>GS-33058</td>
<td>Exception of type &quot;%1&quot; encountered when trying to load image from URL &quot;%2&quot; (cause=%3)</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33059</td>
<td>Value may not contain the file name separator character ' %1 '</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33060</td>
<td>Value &quot;%1&quot; is not supported on this platform (Run &quot;java ImageIOInfo&quot; to obtain a list of available formats)</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33061</td>
<td>Got an exception of type &quot;%1&quot; when attempting to start the PPM image processor &quot;%2&quot; (cause=%3)</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33062</td>
<td>Got an exception of type &quot;%1&quot; when attempting to write to the PPM image processor &quot;%2&quot; (cause=%3)</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33063</td>
<td>Executing PPM processor &quot;%1&quot; returned non zero exit code: %2 (Process output=%3)</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33064</td>
<td>Executing PPM processor &quot;%1&quot; returned non zero exit code: %2)</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33065</td>
<td>Failed to save image to file &quot;%1&quot; (cause=%2)</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33066</td>
<td>Encountered exception of type &quot;%1&quot; while attempting to save a page image to a stream (cause=%2)</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33067</td>
<td>Failed to save PPM image to file &quot;%1&quot; (cause=%2)</td>
<td>This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GS-33068</td>
<td>Invalid value %1 (Allowed values are between 0 and 1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33069</td>
<td>Failed to save PDF document to file &quot;%1&quot; (cause=%2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33070</td>
<td>Encountered exception of type &quot;%1&quot; when attempting to write PDF document to host &quot;%2&quot; at port %3 (cause=%4)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33071</td>
<td>Encountered '{' in list (no nested lists allowed) at character position %1 in data &quot;%2&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33072</td>
<td>Unexpected character '{' at character position %2 in data &quot;%3&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33076</td>
<td>Encountered unexpected '{' character at character position %1 in data &quot;%2&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33078</td>
<td>Missing '{' to close list at character position %1 in data &quot;%2&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33079</td>
<td>Encountered unfinished attribute declaration at character position %1 in data &quot;%2&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33080</td>
<td>Missing '=' character after attribute name at character position %1 in data &quot;%2&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33081</td>
<td>Missing attribute value after '=' character at character position %1 in data &quot;%2&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33082</td>
<td>%1: failed to write to host &quot;%2&quot; at port %3, reason: %4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GS-33083</td>
<td>Unknown encoding <code>%1</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33084</td>
<td>Invalid value '%1' for attribute '%2'</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33085</td>
<td>Internal error: got exception of type '%1' when creating a &quot;date time&quot; attribute from value '%2' (cause='%3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33086</td>
<td>Internal error: got exception of type '%1' when creating an integer attribute from value '%2' (cause='%3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33087</td>
<td>Internal error: got exception of type '%1' when creating a text attribute from value '%2' (cause='%3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33088</td>
<td>Attribute &quot;mediaTrayNumbers&quot; is only available on the Linux platform</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33090</td>
<td>No print service found</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33094</td>
<td>Cannot write to file '%1' msg='%2'</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33095</td>
<td>Internal error: Failed parse pattern value '%1'. Msg='%2'</td>
<td></td>
</tr>
<tr>
<td>GS-33096</td>
<td>Got an exception of type '%1' when attempting to launch the spooler command '%2' (cause='%3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33097</td>
<td>Got an exception of type '%1' when attempting to write to the spooler '%2' (cause='%3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33098</td>
<td>cannot find stream printer for mime type '%1'</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GS-33100</td>
<td>Internal error: reject called</td>
<td></td>
</tr>
<tr>
<td>GS-33101</td>
<td>Internal error: got non 2D Graphics</td>
<td></td>
</tr>
<tr>
<td>GS-33102</td>
<td>Internal error: pageIndex %1 out of range (%2,%3</td>
<td></td>
</tr>
<tr>
<td>GS-33103</td>
<td>Unexpected element &quot;%1&quot;. Expected document root element to be a &quot;PXML&quot; element. Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33104</td>
<td>%1 root element has wrong namespace &quot;%2&quot;. Expected namespace is &quot;%3&quot;. Try adding xmlns=&quot;%4&quot; to the root element. Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33105</td>
<td>Class %1 not found Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33106</td>
<td>Cannot load class %1 (%2) Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33108</td>
<td>Cannot load class %1. constructor not found (%2) Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33110</td>
<td>Internal error: %1::endElementEx() current==null</td>
<td></td>
</tr>
<tr>
<td>GS-33117</td>
<td>IOException on writing RTF document to file &quot;%1&quot; (cause=%2) Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33118</td>
<td>%1 element is not nested within a %2 element Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33120</td>
<td>setting up print service failed (%1) Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33121</td>
<td>IO error during printing (%1), cause=%2 Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33122</td>
<td>aborted printing (%1) Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33123</td>
<td>error during printing (%1:%2) Check your printer.</td>
<td></td>
</tr>
<tr>
<td>GS-33124</td>
<td>error during printing, printing aborted. Check your printer.</td>
<td></td>
</tr>
<tr>
<td>GS-33125</td>
<td>error during printing (%1) Check your printer.</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GS-33126</td>
<td><strong>internal error: got invalid value %1 for property orientationRequested.</strong> Setting to %2.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For further information, see <a href="#">fgl_report_setPrinterOrientationRequested</a> on page 757.</td>
<td></td>
</tr>
<tr>
<td>GS-33127</td>
<td><strong>InvocationTargetException: failed to instantiate style sheet &quot;%1&quot; (cause=%2).</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33128</td>
<td><strong>Exception: failed to instantiate style sheet &quot;%1&quot; (cause=%2).</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33129</td>
<td><strong>IOException reading compressed source (cause=%1)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33131</td>
<td><strong>Invalid command in transform attribute (&quot;%1&quot;)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33132</td>
<td><strong>Not enough arguments to %1 command in transform attribute. Required are %2</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33133</td>
<td><strong>Cannot parse argument &quot;%1&quot; as number for the %2 command in transform attribute</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33134</td>
<td><strong>Wrong number of matrix values in %1 attribute (%2) required are %3</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33135</td>
<td><strong>Uneven number of values in %1 attribute (%2)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33136</td>
<td><strong>Cannot resolve reference in use tag href=%1</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33137</td>
<td><strong>Internal error: Invalid end cap value %1</strong></td>
<td></td>
</tr>
<tr>
<td>GS-33138</td>
<td><strong>Internal error: Invalid line join %1</strong></td>
<td></td>
</tr>
<tr>
<td>GS-33139</td>
<td><strong>Attribute value error: %1 must be a value between %2 and %3 (val= %4)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GS-33140</td>
<td>Wr t error: failed to read from source</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33141</td>
<td>Wr t error: Got non zero status value:%1, msg=%2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33142</td>
<td>Wr t error: Exception=%1, msg=&quot;%2&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33143</td>
<td>IOException writing to file &quot;%1&quot;, msg=&quot;%2&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way</td>
<td></td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
<td></td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33144</td>
<td>IOException flushing to temp file. msg=&quot;%1&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way</td>
<td></td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
<td></td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33145</td>
<td>The TransformerFactory does not support SAX input and SAX output</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way</td>
<td></td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
<td></td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33146</td>
<td>Encountered &quot;TransformerConfigurationException&quot; trying to initialize for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>loading XSLT style from URL &quot;%1&quot; (cause=%2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way</td>
<td></td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
<td></td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33147</td>
<td>Encountered &quot;SAXException&quot; trying to initialize for loading XSLT style</td>
<td></td>
</tr>
<tr>
<td></td>
<td>from URL &quot;%1&quot; (cause=%2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way</td>
<td></td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
<td></td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33148</td>
<td>Encountered &quot;IOException&quot; trying to load XSLT style from URL &quot;%1&quot; (cause=</td>
<td></td>
</tr>
<tr>
<td></td>
<td>%2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way</td>
<td></td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
<td></td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33150</td>
<td>Encountered &quot;TransformerConfigurationException&quot; trying to load XSLT style</td>
<td></td>
</tr>
<tr>
<td></td>
<td>from URL &quot;%1&quot; (cause=%2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way</td>
<td></td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
<td></td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33151</td>
<td>Encountered &quot;SAXNotRecognizedException&quot; trying to load XSLT style from</td>
<td></td>
</tr>
<tr>
<td></td>
<td>URL &quot;%1&quot; (cause=%2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way</td>
<td></td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
<td></td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33152</td>
<td>Encountered &quot;SAXNotSupportedException&quot; trying to load XSLT style from</td>
<td></td>
</tr>
<tr>
<td></td>
<td>URL &quot;%1&quot; (cause=%2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tip: This is an attribute error. Attribute errors are displayed in a way</td>
<td></td>
</tr>
<tr>
<td></td>
<td>that makes them clickable in Genero Studio, as they refer to a particular</td>
<td></td>
</tr>
<tr>
<td></td>
<td>attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GS-33153</td>
<td>Encountered &quot;SAXException&quot; on issuing &quot;startDocument&quot; using XSLT style from URL &quot;%1&quot; (cause=%2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33154</td>
<td>Missing column value for column %1</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33155</td>
<td>Illegal character set name %1</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33156</td>
<td>Unsupported character set name %1</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33157</td>
<td>Failed to write page number template (cause=%1}</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33158</td>
<td>Path %1 is not a directory</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33159</td>
<td>Failed to create directory %1</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33160</td>
<td>Unexpected element &quot;%1&quot;. Expected document root element to be a &quot;SVG&quot; element.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33161</td>
<td>Encountered unreferenceable &quot;g&quot; element in &quot;defs&quot; element (&quot;id&quot; attribute missing).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33162</td>
<td>Encountered unreferenceable &quot;font&quot; element in &quot;defs&quot; element (&quot;id&quot; attribute missing).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33163</td>
<td>Encountered unreferenceable &quot;image&quot; element in &quot;defs&quot; element (&quot;id&quot; attribute missing).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>GS-33164</td>
<td>Encountered faulty &quot;image&quot; element in &quot;defs&quot; element (&quot;href&quot; attribute missing).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td>Tip:</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-33165</td>
<td>Failed to create image file in directory %1</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33166</td>
<td>Failed to create file &quot;index.txt&quot; in directory %1</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33167</td>
<td>font id &quot;%1&quot; does not end with a style digit.</td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-33168</td>
<td>Failed to create WOFF font file &quot;%1&quot;</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33169</td>
<td>Failed to create TTF font file &quot;%1&quot;</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33170</td>
<td>Failed to create EOT font file &quot;%1&quot;</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33171</td>
<td>Failed to rename file %1.part</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33172</td>
<td>Failed to create generic report design document from template %1</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33173</td>
<td>Failed to create temporary generic report design document file</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33174</td>
<td>Printer %1 not found</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33175</td>
<td>Failed to rename WOFF font file to &quot;%1&quot;</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33176</td>
<td>Failed to rename TTF font file to &quot;%1&quot;</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-33177</td>
<td>Failed to close image file &quot;%1&quot; (cause=%2)</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GS-33178</td>
<td>IOException closing file &quot;%1&quot;. msg=&quot;%2&quot;</td>
<td></td>
</tr>
<tr>
<td>GS-33179</td>
<td>Failed to rename EOR font file to &quot;%1&quot;</td>
<td></td>
</tr>
<tr>
<td>GS-33180</td>
<td>Failed to create file &quot;%1&quot;</td>
<td></td>
</tr>
<tr>
<td>GS-33182</td>
<td>Failed to parse %1 as URL</td>
<td></td>
</tr>
<tr>
<td>GS-33184</td>
<td>RTL attribute &quot;name&quot; not specified at &quot;%1&quot; For further information, see Using the expression language on page 849.</td>
<td></td>
</tr>
<tr>
<td>GS-33185</td>
<td>RTL attribute &quot;type&quot; not specified at &quot;%1&quot; For further information, see Using the expression language on page 849.</td>
<td></td>
</tr>
<tr>
<td>GS-33186</td>
<td>Expression error: %1 evaluating RTL expression &quot;%2&quot; at attribute &quot;%3&quot; For further information, see Using the expression language on page 849.</td>
<td></td>
</tr>
<tr>
<td>GS-33187</td>
<td>Computed columns are supported only for absolute paths</td>
<td></td>
</tr>
<tr>
<td>GS-33188</td>
<td>Attribute &quot;type&quot; not specified at &quot;%1&quot;</td>
<td></td>
</tr>
<tr>
<td>GS-33189</td>
<td>Invalid computed column type &quot;%1&quot; (must be one of: %2)</td>
<td></td>
</tr>
<tr>
<td>GS-33190</td>
<td>Attribute &quot;name&quot; not specified at &quot;%1&quot;</td>
<td></td>
</tr>
<tr>
<td>GS-33191</td>
<td>Attribute &quot;expression&quot; not specified at &quot;%1&quot;</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GS-33192</td>
<td>The value %1 for the attribute &quot;initializationLocation&quot; exceeds the value %2 of the path depth specified by the &quot;pathToContainingClass&quot; attribute at &quot;%3&quot;</td>
<td></td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33193</td>
<td>The value %1 for the attribute &quot;initializationLocation&quot; exceeds the value %2 of the path depth specified by the &quot;pathToContainingClass&quot; attribute at &quot;%3&quot;</td>
<td></td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33194</td>
<td>The value %1 for the attribute &quot;placementLocation&quot; exceeds the value %2 of the path depth specified by the &quot;pathToContainingClass&quot; attribute at &quot;%3&quot;</td>
<td></td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33195</td>
<td>The value %1 for the attribute &quot;placementLocation&quot; is not smaller or equal to the value %2 of the &quot;placementLocation&quot; at &quot;%3&quot;</td>
<td></td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33196</td>
<td>Path must be absolute</td>
<td></td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33197</td>
<td>No SortItems specified at &quot;%1&quot;</td>
<td></td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33198</td>
<td>Got an exception of type &quot;%1&quot; when attempting to connect to the PJL printer &quot;%2&quot; (cause=%3)</td>
<td></td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33199</td>
<td>Got an exception of type &quot;%1&quot; when attempting to initialize PJL job (cause=%2)</td>
<td></td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33200</td>
<td>Got an exception of type &quot;%1&quot; when attempting to initialize PJL from IPP attributes (cause=%2)</td>
<td></td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33201</td>
<td>Got an exception of type &quot;%1&quot; when attempting to instantiate Postscript optimizer (cause=%2)</td>
<td></td>
</tr>
<tr>
<td>Tip:</td>
<td>This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GS-33202</td>
<td>Got an exception of type &quot;%1&quot; when attempting to create destination file &quot;%2&quot; for writing (cause=%3)</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33203</td>
<td>Destination URI %1 doesnt denote a local file</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33204</td>
<td>lpr printer options are not available on this platform</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33205</td>
<td>Unknown transformation processor name %1 at %2</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33206</td>
<td>Unknown &quot;%1&quot; element in Aggregations element at %2</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33207</td>
<td>Unknown &quot;%1&quot; element in Grouping element at %2</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33208</td>
<td>Missing &quot;path&quot; attribute in Collection element at %1</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33209</td>
<td>Missing &quot;collectionItemName&quot; attribute in Collection element at %1</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33210</td>
<td>Unknown &quot;%1&quot; element in GroupBy element at %2</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33211</td>
<td>Missing &quot;expression&quot; attribute in GroupBy element at %1</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33212</td>
<td>Missing &quot;name&quot; attribute in GroupBy element at %1</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33213</td>
<td>Unknown &quot;%1&quot; element in GroupBy element at %2</td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>This is an <strong>attribute error</strong>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GS-33214</td>
<td>Missing &quot;name&quot; attribute in input-variable element at %1&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;strong&gt;Tip:&lt;/strong&gt; This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33215</td>
<td>Missing &quot;type&quot; attribute in input-variable element at %1&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;strong&gt;Tip:&lt;/strong&gt; This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33216</td>
<td>Missing &quot;path&quot; attribute in Aggregation element at %1&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;strong&gt;Tip:&lt;/strong&gt; This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33217</td>
<td>Missing &quot;name&quot; attribute in Aggregation element at %1&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;strong&gt;Tip:&lt;/strong&gt; This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33218</td>
<td>Missing &quot;rule&quot; attribute in Aggregation element at %1&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;strong&gt;Tip:&lt;/strong&gt; This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33219</td>
<td>Missing &quot;expression&quot; attribute in Aggregation element at %1&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;strong&gt;Tip:&lt;/strong&gt; This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33220</td>
<td>Unknown &quot;%1&quot; element in GroupBy element at %2&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;strong&gt;Tip:&lt;/strong&gt; This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33221</td>
<td>CError computing aggregate &quot;Min&quot; from the RTL expression &quot;%1&quot; since the computed value &quot;%2&quot; does not implement the Comparable interface&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;strong&gt;Tip:&lt;/strong&gt; This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33222</td>
<td>CError computing aggregate &quot;Max&quot; from the RTL expression &quot;%1&quot; since the computed value &quot;%2&quot; does not implement the Comparable interface&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;strong&gt;Tip:&lt;/strong&gt; This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33223</td>
<td>This version supports stand alone licenses only&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;strong&gt;Tip:&lt;/strong&gt; This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>GS-33224</td>
<td>Failed to highlight using a custom highlighter. Failed to invoke the static method &quot;CustomHighlighter.highlight(Graphics2D,String,Rectangle2D.Double,Color,Color)&quot; (cause=%1)&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;strong&gt;Tip:&lt;/strong&gt; This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GS-33225</td>
<td>PrintException during printing (%1), cause=%2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Check your printer.</td>
<td></td>
</tr>
</tbody>
</table>

Table 280: Error messages for the Package util.domutil.selection

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-34800</td>
<td>Internal error: Can't find element denoted by the path %1 in the document fragment %2</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
</tbody>
</table>

Table 281: Error messages for the Package util.domutil

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-34500</td>
<td>Internal error: Element.getLocalName() returned null. Apparently the document was modified using DOM Level 1 functions</td>
</tr>
<tr>
<td>GS-34501</td>
<td>Attribute type error: Expected integer value for attribute %1</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-34502</td>
<td>Attribute type error: Expected long value for attribute %1</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-34503</td>
<td>Attribute type error: Expected double value for attribute %1</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-34504</td>
<td>Attribute type error: Expected boolean value for attribute %1</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-34505</td>
<td>Failed to parse value &quot;%1&quot; as URL (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-34506</td>
<td>Cannot resolve relative URL %1. Have no system id to absolutize against.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-34507</td>
<td>Failed to parse &quot;systemId&quot; value &quot;%1&quot; as URI (cause=%2)</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-34508</td>
<td>Failed to create absolute URL from systemId=&quot;%1&quot; and URL=&quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an <em>attribute error</em>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
</tbody>
</table>
### Table 282: Error messages for the Package util.expressionparser

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-35400</td>
<td>Internal error: pushString called on function %1. The function takes only one numeric parameter</td>
</tr>
<tr>
<td>GS-35401</td>
<td>Internal error: getStringReturnValue called on function %1. The function returns a numeric value</td>
</tr>
<tr>
<td>GS-35402</td>
<td>Internal error: pushString called on function %1. The function takes only two numeric parameters</td>
</tr>
<tr>
<td>GS-35403</td>
<td>Internal error: getStringReturnValue called on function %1. The function returns a numeric value</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-35404</td>
<td>RTL expression error: Illegal access error. Not able to obtain value of variable %1 on object of type %2 (cause=%3)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35405</td>
<td>RTL expression error: Illegal array or array index. Not able to obtain array value for index %1 on List of type %2 (err=%3)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35406</td>
<td>RTL expression error: Illegal array index. Not able to obtain array value for index %1 on List of type %2 (cause=%3)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35407</td>
<td>RTL expression error: Unable to resolve non numeric key %1 on array of type %2 (Non numeric arguments can only be used on &quot;Map&quot; arrays)</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35408</td>
<td>Unknown media type &quot;%1&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-35409</td>
<td>Unknown media type &quot;%1&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-35410</td>
<td>Unexpected extra text %1 at end of PXML expression &quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35411</td>
<td>Unexpected extra text %1 at end of PXML string expression &quot;%2&quot;</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35412</td>
<td>Unexpected token %1 in PXML expression. Expected else operator ':=' in ternary conditional expression</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35413</td>
<td>Unexpected token %1 following '</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35414</td>
<td>Unexpected token %1 following '&amp;' in PXML expression</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35415</td>
<td>Unexpected token %1 following '%2' in PXML expression</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35416</td>
<td>Unexpected end of PXML expression</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>GS-35417</td>
<td>Type error: illegal call to PXML function &quot;%1&quot; returning %2 while expecting a function that returns a numeric value. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35418</td>
<td>Expected enumeration identifier for parameter number %1 in call to PXML function %2 but got %3 instead. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35419</td>
<td>Missing function arguments in call to PXML function %1. Expected %2 parameters but found %3 instead. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35420</td>
<td>Missing '}' in call to PXML function &quot;%1&quot;. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35421</td>
<td>Unmatched '(' in PXML expression. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35422</td>
<td>Internal error in PXML expression (unexpected token %1). Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35423</td>
<td>Unexpected end of PXML string expression. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35424</td>
<td>Type error: illegal call to PXML function &quot;%1&quot; returning %2 when expecting a function that returns a string value. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35425</td>
<td>Expected enumeration identifier for parameter number %1 in call to PXML string function &quot;%2&quot; but got %3 instead. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35426</td>
<td>Missing function arguments in call to PXML string function &quot;%1&quot; (Expected %2 parameters but found %3). Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35427</td>
<td>Missing '}' in call to PXML string function &quot;%1&quot;. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35428</td>
<td>Unmatched '(' in PXML string expression. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35429</td>
<td>Invalid operator in expression: '%1' (%2). For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35430</td>
<td>Unexpected extra token &quot;%1&quot; at end of RTL expression &quot;%2&quot;. For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-35431</td>
<td>Unexpected token %1 in RTL expression. Expected else operator ':' in ternary</td>
</tr>
<tr>
<td></td>
<td>conditional expression</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35432</td>
<td>Error in RTL conditional expression: Expected %1 value to precede '?' operator but got a %2 value</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35433</td>
<td>Error in RTL conditional or expression: Expected left operand to be of type %1 but got a %2 value</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35434</td>
<td>Error in RTL conditional or expression: Expected right operand to be of type %1 but got a %2 value</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35435</td>
<td>Error in RTL conditional and expression: Expected left operand to be of type %1 but got a %2 value</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35436</td>
<td>Error in RTL conditional and expression: Expected right operand to be of type %1 but got a %2 value</td>
</tr>
<tr>
<td></td>
<td>For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35437</td>
<td>RTL expression error: Relational operator &quot;&gt;=&quot; cannot be applied to types %1,%2</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35438</td>
<td>RTL expression error: Relational operator &quot;&gt;&quot; cannot be applied to types %1,%2</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35439</td>
<td>RTL expression error: Relational operator &quot;&lt;=&quot; cannot be applied to types %1,%2</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35440</td>
<td>RTL expression error: Relational operator &quot;&lt;&quot; cannot be applied to types %1,%2</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35441</td>
<td>RTL expression error: Operator '+' cannot be applied to %1,%2</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35442</td>
<td>RTL expression error: Operator '-' cannot be applied to %1,%2</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35443</td>
<td>RTL expression error: Operator '*' cannot be applied to %1,%2</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-35444</td>
<td>RTL expression, type mismatch: cannot multiply value of type %1 Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35445</td>
<td>RTL expression, division failed (cause=%1) Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35446</td>
<td>RTL expression: Operator '/' cannot be applied to %1,%2 For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35447</td>
<td>RTL expression, integer division failed (cause=%1) Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35448</td>
<td>RTL expression error: Operator '%%' cannot be applied to %1,%2 For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35449</td>
<td>RTL expression error: Illegal array index value (null) For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35450</td>
<td>RTL expression error: Unexpected token %1 (&quot;%2&quot; following array index expression. Expected ']' to terminate expression For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35451</td>
<td>RTL expression error: Cannot negate object of type %1 For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35452</td>
<td>RTL expression error: Cannot perform logical negation on an object of type %1 For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35453</td>
<td>Unexpected end of RTL expression For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35454</td>
<td>RTL expression error: Unexpected token %1 (&quot;%2&quot; in call to function %3. Expected either ',' or ')'. For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35455</td>
<td>RTL expression error: Unmatched '(' For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35456</td>
<td>RTL expression error: Unexpected token %1 (type=%2) For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35457</td>
<td>ExpressionTokenizer: Internal error: pushBack() called more than once in a row. For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35458</td>
<td>Invalid character '%1' in expression &quot;%2&quot; For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-35459</td>
<td>Unexpected end of expression in expression &quot;%1&quot;, was expecting '%2' For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35460</td>
<td>Unexpected character '%1' in expression '%2', was expecting '%3' For further information, see Using the expression language on page 849.</td>
</tr>
<tr>
<td>GS-35461</td>
<td>Failed to parse numeric value &quot;%1&quot; Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35462</td>
<td>Failed to parse &quot;%1&quot; as a numeric value for the RTL variable &quot;%2&quot; Tip: This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-35463</td>
<td>Unknown function &quot;%1&quot; Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35464</td>
<td>Unknown function &quot;%1()&quot; Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35465</td>
<td>Unknown function &quot;%1(2)&quot; Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35466</td>
<td>Unknown PXML unit &quot;%1&quot; For further information, see Unit Names on page 978.</td>
</tr>
<tr>
<td>GS-35467</td>
<td>Unknown variable &quot;%1&quot; Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-35468</td>
<td>Unexpected token &quot;&quot; + TOKEN_NAMES[tttype] + &quot;&quot; in PXML string expression Error message and resolution should be self-explanatory.</td>
</tr>
</tbody>
</table>

Table 283: Error messages for the Package util.getopt

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-36000</td>
<td>Invalid '=' character in option &quot;%1&quot; Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36001</td>
<td>Internal error: Invalid usage</td>
</tr>
<tr>
<td>GS-36002</td>
<td>Invalid option &quot;%1&quot;. Expected '-' to start option Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36003</td>
<td>Invalid option &quot;%1&quot;. Expected option name to follow '-' Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36004</td>
<td>Invalid option &quot;%1&quot;. Expected option name to follow &quot;--&quot; Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-36005</td>
<td>Unknown option &quot;%1&quot;</td>
</tr>
<tr>
<td>GS-36006</td>
<td>Missing required argument for option &quot;%1&quot;.</td>
</tr>
</tbody>
</table>

**Table 284: Error messages for the Package util.regexp.derivates**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-36600</td>
<td>Unknown external reference %1</td>
</tr>
<tr>
<td>GS-36601</td>
<td>Internal error: Tokenizer returned invalid values n=%1</td>
</tr>
<tr>
<td>GS-36603</td>
<td>Internal error: Tokenizer returned invalid values m=%1</td>
</tr>
<tr>
<td>GS-36604</td>
<td>Internal error: Tokenizer returned invalid values n=%1 m=%2</td>
</tr>
<tr>
<td>GS-36606</td>
<td>Internal error: got internal reference token allthough the stream claims not to produce these</td>
</tr>
<tr>
<td>GS-36607</td>
<td>Reference index %1 out of range (max=%2)</td>
</tr>
<tr>
<td>GS-36608</td>
<td>found recursive external reference %1. Stack=%2</td>
</tr>
<tr>
<td>GS-36609</td>
<td>Line does not consist of two space separated fields</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-36610</td>
<td>Expression file is empty</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> This is an attribute error. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</td>
</tr>
<tr>
<td>GS-36611</td>
<td>Unexpected end of expression</td>
</tr>
<tr>
<td>GS-36613</td>
<td>Unexpected EOF in control character specification</td>
</tr>
<tr>
<td>GS-36614</td>
<td>Illegal control character specification &quot;\c%1&quot;</td>
</tr>
<tr>
<td>GS-36615</td>
<td>Unexpected EOF in external reference</td>
</tr>
<tr>
<td>GS-36616</td>
<td>Expected '{' character to follow \R in external reference. Got '}' instead</td>
</tr>
<tr>
<td></td>
<td><strong>Error message and resolution should be self-explanatory.</strong></td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-36617</td>
<td>Unclosed external reference</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36618</td>
<td>Unexpected EOF in tracked external reference</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
</tbody>
</table>
| GS-36619 | Expected '{' character to follow \
R in a tracked external reference. Got  \\%1' instead |
|          | Error message and resolution should be self-explanatory.                    |
| GS-36620 | Unclosed tracked external reference                                          |
|          | Error message and resolution should be self-explanatory.                    |
| GS-36621 | Unexpected EOF in marker definition                                         |
|          | Error message and resolution should be self-explanatory.                    |
| GS-36622 | Expected '{' character to follow \
M in marker definition. Got  \\%1' instead |
|          | Error message and resolution should be self-explanatory.                    |
| GS-36623 | Unclosed marker definition                                                   |
|          | Error message and resolution should be self-explanatory.                    |
| GS-36627 | Expected '{' character to follow \
p in character class definition. Got  \\%1' instead |
<p>|          | Error message and resolution should be self-explanatory.                    |
| GS-36628 | Unclosed character class definition                                          |
|          | Error message and resolution should be self-explanatory.                    |
| GS-36629 | Unknown character class name %1                                             |
|          | Error message and resolution should be self-explanatory.                    |
| GS-36630 | Invalid escape &quot;\%1&quot;                                                      |
|          | Error message and resolution should be self-explanatory.                    |
| GS-36631 | Unclosed character class                                                    |
|          | Error message and resolution should be self-explanatory.                    |
| GS-36632 | Illegal digit '{' in range modifier                                         |
|          | Error message and resolution should be self-explanatory.                    |
| GS-36633 | Unclosed range modifier                                                     |
|          | Error message and resolution should be self-explanatory.                    |
| GS-36634 | Upper range value smaller than lower range value in range modifier           |
|          | Error message and resolution should be self-explanatory.                    |</p>
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-36635</td>
<td>Invalid range values (%1,%2) in range modifier</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36636</td>
<td>Invalid range value (%1, ) in range modifier</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36637</td>
<td>Invalid range value (, %1) in range modifier</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36638</td>
<td>Illegal range modifier</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36639</td>
<td>Empty range modifier</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36640</td>
<td>Failed to find an element definition</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36641</td>
<td>Failed to find referenced element</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36642</td>
<td>Failed to find complexType element</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36643</td>
<td>Failed to find simpleType element</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36644</td>
<td>Failed to find attribute declaration</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36645</td>
<td>Failed to find attribute group declaration</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36646</td>
<td>Failed to find group declaration</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36647</td>
<td>Neither &quot;ref&quot; nor &quot;name&quot; Attributes not set on attribute group</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36648</td>
<td>Neither &quot;ref&quot; nor &quot;name&quot; Attributes not set on attribute</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36649</td>
<td>Encountered empty simpleType</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-36650</td>
<td>Encountered extension without base declaration</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| GS-36651 | Encountered restriction without base declaration  
Error message and resolution should be self-explanatory. |
| GS-36652 | Neither "ref" nor "name" Attributes not set on element  
Error message and resolution should be self-explanatory. |
| GS-36653 | maxOccurs has value 0 in particle  
Error message and resolution should be self-explanatory. |
| GS-36654 | maxOccurs is smaller than minOccurs in particle  
Error message and resolution should be self-explanatory. |
| GS-36656 | Internal error: Unknown type %1 |

Table 285: Error messages for the Package util.regexp

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
</table>
| GS-36300 | Unexpected extra text at end of expression at %1 (c='%2')  
Error message and resolution should be self-explanatory. |
| GS-36301 | Unexpected end of expression  
Error message and resolution should be self-explanatory. |
| GS-36302 | Missing ']' at %1  
Error message and resolution should be self-explanatory. |
| GS-36303 | Missing ')' at %1  
Error message and resolution should be self-explanatory. |
| GS-36305 | Expected character to follow '-' at %1  
Error message and resolution should be self-explanatory. |
| GS-36306 | Expected character to follow '[' at %1  
Error message and resolution should be self-explanatory. |
| GS-36307 | Internal error: Called unread() more than once at %1 |
| GS-36308 | Internal error: Called unread() more than once at %1 |
| GS-36309 | Invalid escape "\%1" at %2  
Error message and resolution should be self-explanatory. |

Table 286: Error messages for the Package util

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-34200</td>
<td>Internal error: uri is not allowed to be null</td>
</tr>
<tr>
<td>GS-34201</td>
<td>Internal error: localName is not allowed to be null</td>
</tr>
<tr>
<td>GS-34202</td>
<td>Internal error: uri is not allowed to be null</td>
</tr>
<tr>
<td>GS-34203</td>
<td>Internal error: localName is not allowed to be null</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-34204</td>
<td>Internal error: uri is not allowed to be null</td>
</tr>
<tr>
<td>GS-34205</td>
<td>Internal error: localName is not allowed to be null</td>
</tr>
<tr>
<td>GS-34206</td>
<td>Internal error: uri is not allowed to be null</td>
</tr>
<tr>
<td>GS-34207</td>
<td>Internal error: localName is not allowed to be null</td>
</tr>
<tr>
<td>GS-34208</td>
<td>Internal error: uri is not allowed to be null</td>
</tr>
<tr>
<td>GS-34209</td>
<td>Internal error: '0'-'3' or 'u' expected after '':: %1</td>
</tr>
<tr>
<td>GS-34210</td>
<td>Internal error: '0'-'7' or 'u' expected after '':: %1</td>
</tr>
<tr>
<td>GS-34213</td>
<td>Unmatched quote at end of string &quot;%1&quot;)</td>
</tr>
<tr>
<td>GS-34214</td>
<td>Unmatched quote in string &quot;%1&quot; at position%2</td>
</tr>
<tr>
<td>GS-34215</td>
<td>failed to map file %1 into memory. Reason: %2</td>
</tr>
<tr>
<td>GS-34216</td>
<td>corrupted file %1. Corruption type: 1</td>
</tr>
<tr>
<td>GS-34217</td>
<td>corrupted file %1. Corruption type: 2</td>
</tr>
<tr>
<td>GS-34218</td>
<td>corrupted file %1. Corruption type: 3</td>
</tr>
<tr>
<td>GS-34219</td>
<td>corrupted file %1. Corruption type: 3 at position %2</td>
</tr>
<tr>
<td>GS-34220</td>
<td>Internal error: Encountered open Edge</td>
</tr>
<tr>
<td>GS-34221</td>
<td>Internal error: Encountered open CharacterEdge</td>
</tr>
</tbody>
</table>

**Redirect log messages to a file**

By default, GRE error and warning messages appear in the standard output. However, you can redirect these messages to a log file.

1. Create a new text file with the following entries:

   ```
   log4j.rootLogger = INFO, LOGFILE
   log4j.appenders.LOGFILE = org.apache.log4j.FileAppender
   log4j.appenders.LOGFILE.file = url/filename.log
   log4j.appenders.LOGFILE.layout = org.apache.log4j.PatternLayout
   log4j.appenders.LOGFILE.layout.ConversionPattern = %-4r %5p %c - %m - %X{requestingUserName}%n
   ```

   Replace `url` with the location and `filename` with the name of the log file.
2. Open the GREDIR/bin/greportwriter.bat file. To the java command, add the following option:

```
exec java -Dlog4j.configuration=url/filename -Djava.awt.headless=true GReportWriter $DEBUGOPTION "$@"
```

Where `url` is the location and `filename` is the name of the text file you created in step 1. For example:

```
exec java
-Dlog4j.configuration="file:///home/alex/log4j_configuration.txt"
-Djava.awt.headless=true GReportWriter $DEBUGOPTION "$@"
```

GRE logs are written to the `url/filename.log` file.

**Note:** If you want to have the same type of logging in non-distributed mode as in distributed mode, it is not necessary to specify an external file. You can instead specify a resource path to one of the configuration files that are shipped inside `gre.jar`. These files are dependent on the operating system:

- **Unix:** `log4j_configuration_unx_d.txt`
- **Windows:** `log4j_configuration_win_d.txt`
- **Other:** `log4j_configuration_oth_d.txt`

For example, in Linux™:

```
exec java
-Dlog4j.configuration="com/fourjs/report/resources/log4j_configuration_unx_d.txt"
-Djava.awt.headless=true GReportWriter $DEBUGOPTION "$@"
```

### Report an issue to support

Use the GREDATAFILE environment variable to generate debug information that you can use in a support ticket.

If you are encountering an issue with your Genero Report Writer application, you need to be able to gather the processing instructions and the XML data stream being sent to the Genero Report Engine.

Follow these instructions if you need to open a support ticket.

1. In the Genero Configuration Management dialog on page 197, create an environment variable called GREDATAFILE.
2. Set GREDATAFILE to the file name where you want to store the debug information, for example, `reportdatafile.xml`.
3. Run the application.

The application writes to the specified file. It will not create a report.

4. Provide your support person with the following files:
   - The data file specified by GREDATAFILE
   - The report definition document (.4rp)
   - All resources accessed by the report, for example, the images, HTML files, subreports, and data schemas (.rdd or .xsd)

Four Js Support will be able to reproduce the issue using the files provided.

## Web Services

Information about using Web Services in Genero Studio.

- [Create a Web Services program](#) on page 1076
- [Add Web Service](#) on page 1076
- [The Web Services wizard](#) on page 1076
• Build the application on page 1079
• Update the WSDL on page 1080
• Generating web services

See also the Web Services topic in the *Genero Business Development Language User Guide*.

### Create a Web Services program

You can use Genero Studio to write, compile, and execute a Genero Web Services (GWS) client or server application.

1. Add an application, library, or folder node to a project in Project Manager.
2. Right-click the node, and select Add Web Service. Select either Client Consuming WSDL or Server Implementing WSDL.
   
   This calls the The Web Services wizard on page 1076, to assist you in adding the Web Services client or server information to the application.
3. Use Code Editor to create or modify a Genero application that calls a Web Service (client) or defines a Web Service (server).
   
   Use the WSDL information to determine the service name and functions. Code Editor recognizes and parses the syntax of BDL functions related to Genero Web Services. See the *Genero Business Development Language User Guide* for details on coding your Web services clients and servers.
4. In Genero Studio Project Manager, add the WSHelper.42m file to the external dependency property of your Genero application; this compiled file contains functions used internally for Web Services.
5. Use the Genero Studio Build menu option to compile and link your new application.

If you must access the web service using a proxy, provide the proxy information using Tools > Preferences.

### Add Web Service

To access or define a Web service, the Web Services Description Language (WSDL) description must be created by the service provider. You can use the WSDL description to determine the operations of the Web Service in your BDL program, the required parameters, and the return values.

1. Right-click an Application, folder or library node in Project Manager and select Add Web Service.
2. In the Web Services Wizard, choose between:
   
   a) Client Consuming WSDL - to retrieve the WSDL information and generate a .4gl file containing the BDL functions for the WSDL operations, which can be called by a GWS client application.
   
   b) Server Implementing WSDL - to retrieve the WSDL information and generate a .4gl file (GWS Server Application) containing the BDL functions to create and publish a Web Service which can be accessed over the web.

### Related concepts

The Web Services wizard on page 1076

The Web Services Wizard guides you through the steps to add the necessary service information to your application.

### The Web Services wizard

The Web Services Wizard guides you through the steps to add the necessary service information to your application.

**Web Services selection Page**

Provide the Service details to get the WSDL description, using one of these:

- **URL** - Provide the URL of the web service. Example: http://servername:port/servicename?WSDL
- **File** - Provide the name and location of a WSDL file in your file system.
Click the View button. The description of the Web Service is retrieved from the WSDL and displayed.

**Figure 432: Add Web Services - Client consuming WSDL dialog**

**Copy to** - Use the Browse button to specify the directory where a copy of the WSDL file should be saved. If you used the File option to specify the WSDL file, this directory must be different.

**Advanced Options button** - If you select this button, a dialog displays that allows you to add an external schema file with the WSDL file.
Within the Advanced Options dialog:

- **Additional Schema File** - press the Add icon to add an external schema file. An Add dialog opens, allowing you to search for the file. Other icons allow you to delete a previously added file, or to edit it.
- **Additional Schema Folder** - press the Add icon to add the path and filename of the schema folder.
- **Https Options** - Set the options needed for the fglwsdl tool, which implements adding the external schema.

**Web Services Project Integration (Client only)**

**Add Web Services to Project** - if this radio button is selected, the WSDL file will be added to your project in Project Manager. Files that show the Web Services operations as functions in a Genero Program are generated and stored in the Target directory of the application:

- **inc file** - the globals file containing the definitions of the input and output records, and the prototypes of the operations.
- **4gl file** - contains the definitions of the functions for your GWS client or server application, and the code that manages the Web Service request or publishes a service.

**Advanced** - this radio button gives you the flexibility to write your own web client stub. The 4gl and inc files are generated and added to your project in Project manager. The WSDL file is created in the "copy to" path.

**Important**: Do not modify the generated 4gl files; they are regenerated on each re-build, so modifications will be lost.
Advanced Code Generation Options

Select the options for code generation and runtime behavior, if you wish to modify the default behavior. See The fglwsdl tool for a complete list of the command options.

- **Select service/binding** - Choose from the list of all Services or binding in the generated wsdl file.

- **Behavior and Functionality**
  - **Compatible with Genero 1.xx** - Implements the "-compatibility" option of the Genero tool fglwsdl.
  - **Force RPC convention**  (applies to RPC only) - Implements "-fRPC" option of fglwsdl.
  - **Generate choice records for inherited types** - Implements "-Inheritance" option of fglwsdl.
  - **Ignore SOAP faults**  - Implements "-ignoreFaults" option of fglwsdl.
  - **Use WS-addressing** - Allows support of WS-Addressing (only WS-Addressing 1.0 is supported.)
  - **Check data type constraints (facets) for simple types** - Implements "-noFacets" option of fglwsdl. By default facet is present, so this option is enabled.
  - **Generate XML list of arrays** - Implements "-fArray" option of fglwsdl.
  - **Generate DOM callback handlers** - Implements "-domHandler" option of fglwsdl.
  - **Generate FGLPROFILE aliases for server URL** Generates FGLPROFILE Logical names in place of URLs for the client stub - implements "-alias" option of fglwsdl.

- **Coding Conventions & Style**
  - **Include service documentation** - Implements "-comments" option of fglwsdl.
  - **Prefix functions, variables, and types with** - Implements "-prefix" option of fglwsdl.
  - **Prefix variables and types with namespace** - Implements "-autoNsPrefix" option of fglwsdl. Specify the number of words of the namespace that are to be excluded from the prefix. The default prefix is the full namespace, and you may exclude up to nine words. After the ninth word, you can only add the last word of the namespace as the prefix.
  - **Define prefix for variables and types per namespace** - Implements "-nsPrefix" option of fglwsdl. For each namespace, you can define a prefix for variables and types.

- **Command Line** - additional arguments for the execution of the fglwsdl tool.

  The command line is read-only, and is automatically updated when you select an option.

Code Generation Result - optional page

This page displays only if you have selected **Advanced** in the Web Services Project Integration page of the wizard.

Select **Finish** to generate the WSDL file and accompanying files.

**Note:** If you must access the web service using a proxy, provide the proxy information in General Preferences on page 128.

Build the application

The Genero library file **WSHelper.42m** must be linked into every application using Web Services.

**Build Rules** are defined to link the generated 4gl file into your application. The generated 4gl file in the Target Directory should not be modified, as your changes will be lost if the file is regenerated.

**Important:** The Genero library file **WSHelper.42m** must be linked into every application using Web Services. Add the library file to the application node in Project Manager as an external dependency property, prior to building the application.

When you build the Project, the appropriate files are compiled and linked.
**Update the WSDL**

Right-click the WSDL node in the project structure to update the WSDL file.

An internet connection is required, unless the Web Services server is local.

While updating the file, any new dependency files are also downloaded.

---

**Mobile applications**

Genero Studio provides the framework for creating robust Genero mobile applications.

---

**Mobile development environment**

A complete and inclusive development environment allows you to write and test your Genero mobile apps from your desktop prior to deploying them to your mobile devices.

The environment includes the runtime DVM, the development IDE (Genero Studio), and two development agents: Genero Mobile for iOS (GMI) and Genero Mobile for Android™ (GMA). During development, the DVM and the IDE sit on the desktop, and the GMI and GMA display clients sit on mobile devices. A mobile device can be a physical device, such as a phone or tablet, or it can be an emulator.

During development, you run the app in **developer mode**. In developer mode, the app runs on a development machine and displays the user interface (UI) to an Android™ or iOS device or emulator. The server app makes the initial connection. The Genero Mobile Development Client runs on the device, listening at port 6400 (by default) for the connection.

By running the app from your development environment, you can develop and debug your app from within the Genero Studio IDE. You do not have to continuously copy files to the device. You can quickly see the user interface and engage with the app. You can quickly switch between various devices.

---

**Figure 434: Developer mode**

**Options for development**

Three clients are available for use while developing Genero mobile apps.
• Genero Mobile for Android™ (GMA)
  This client allows you to test your app on an Android™ physical device or emulator during development.
  To put the GMA client on your Android™ device, you must connect your device to your computer via the USB port.
  To run your app in developer mode and view it on your Android™ device, you must connect your device to your development machine via the USB port.
  See Configure Genero Mobile for Android on page 142 for details on setting up the GMA for development.

• Genero Mobile for iOS (GMI)
  This client allows you to test your app on an iOS physical device or simulator during development. The development machine must use a macOS™.
  To put the GMI client on your iOS device, you must connect your device to your computer via the USB port.
  To run your app in developer mode and view it on your iOS device, connect to your development machine via the USB port.
  See Configure Genero Mobile for iOS on page 151 for details on setting up the GMI for development.

• Genero Development Client
  Using the Genero Development Client, you can view your app on an iOS device when developing with a Windows® or Linux® development machine. macOS™ is required on the mobile device.
  **Important:** While this client allows you to view and test your app on the device in developer mode, it does not give you the ability to create an iOS package or to deploy your app to an iOS device.
  To put the Genero Development Client on your iOS device, you must download the app from the Apple® store.
  To run your app in developer mode and view it on your iOS device using the Genero Development Client, you must use a wireless network (wifi).
  See Display to the Genero Mobile Development Client on page 162 for details on using the Genero Development Client.

**Tools to assist with development**

Genero Mobile has a variety of tools to aid in development.

• Preview the form on the mobile device.
• Debug your app using the graphical debugger.
• Use the profiler to gather statistics on where your program spends its time and to identify bottleneck functions.
• View the AUI tree in a browser on your development machine (or any browser on the local Wifi network).
• Display the program logs.
• Display the device logs.

The Genero Studio graphical debugger is part of Genero Studio. The command-line debugger (fgldb) is included with Genero BDL, and is described in the Genero Business Development Language User Guide.

See Debugging a mobile app on page 615.

**Using command-line tools**

Working from the desktop, you have the full suite of command line tools at your disposal: gsmake, fglrun, fglrun –d, and so on. See the Genero Business Development Language User Guide for information on the command line tools.
Deploy to the device for testing

The development clients are provided to ease your development efforts, allowing you to view your app without having to package and deploy your app to the device to view changes in the app code. Towards the end of the development cycle, you will want to test your app running fully on the device.

When a package is deployed on a device where the debug package has been activated, debugging tools available include:

- View the AUI tree in a browser.
- Display the program logs.
- Display the device logs.
- Command-line on-device debugging.

See Packaging, deploying, and distributing apps on page 1168.

Genero mobile app demos

If you are looking for code snippets and examples, you are encouraged to view the demo apps.

You can run the demos in developer mode or you can deploy each demo to your device.

OfficeStoreMobile

OfficeStoreMobile is provided as one of the default projects. Access the project from the Tutorials & Samples tab on the Welcome Page. This app was created using the Business Application Modeler.

This demo is the Office Store demo, where you can view a list of customers and their orders.

You can deploy the package for the app to your device or emulator. The package can be found under the Packages node in the project.

MobilePatterns

MobilePatterns is provided as one of the default projects. Access the project from the Tutorials & Samples tab on the Welcome Page. This app was created using the Business Application Modeler.

This demo shows how you can define relationships between two forms, and how you can use a common form with multiple relationships defined.

You can deploy the package for the app to your device or emulator. The package can be found under the Packages node in the project.

PhoneDemo

The PhoneDemo includes the Stores demo and additional demos that show how various forms, menus and widgets are displayed on your mobile device. You can access the project from the Tutorials & Samples tab on the Welcome Page.

The source code can be found at $GSTDIR/samples/PhoneDemo or, if you are not using Genero Studio, $FGLDIR/demo/MobileDemo/phoneMain.

In addition to running the demo in developer mode or deploying the demo to your device, you can start the PhoneDemo from the Genero Mobile client.
Table 287: Starting the PhoneDemo from your Genero Mobile client

<table>
<thead>
<tr>
<th>GMA</th>
<th>GMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Start Genero Mobile on your device.</td>
<td>1. Start Genero Mobile on your device.</td>
</tr>
<tr>
<td>2. Tap Start demo.</td>
<td>2. Tap Run Demo.</td>
</tr>
</tbody>
</table>

The Stores demo is a non-generated app that provides a simple order tracking app for a fictional sporting goods store.

The source code for the Stores Demo can be found at $GSTDIR/samples/PhoneDemo/stores or, if you are not using Genero Studio, $FGLDIR/demo/MobileDemo/stores.

stores2

The Stores2 demo is the Stores demo written for a tablet device. It demonstrates the implementation of the split view and the navigator pane. You can access the project from the Tutorials & Samples tab on the Welcome Page.

The source code can be found at $GSTDIR/samples/PhoneDemo/stores2 or, if you are not using Genero Studio, $FGLDIR/demo/MobileDemo/stores2.

Debugging a mobile app

You have several tools available for debugging your Genero Mobile application.

To use the debugging tools, you will either have:

- An app running in developer mode, with the Debug service enabled.
- A debug version of a deployed app.

Debug version of a deployed app

Debugging methods and tools are available for the debug versions of standalone apps.

- Create a debug version of a deployed app on page 615
- Run the debug version of a deployed app (Android) on page 616
- Run a debug version of a deployed app (iOS) on page 616

Create a debug version of a deployed app

By default, the mobile app is built in release mode, which cannot be debugged. To build an app in debug mode, you must check the Mobile Debug Package environment set, which sets the DEBUG_PACKAGE environment variable to 1.

Before you begin, your app project must be open in Genero Studio.

1. Select Tools > Genero Configurations.
2. Select a Configuration Name.
   The current configuration is selected by default.
Within this environment set, the DEBUG_PACKAGE environment variable is set to 1 (TRUE) to build a debug version of the package.

4. Package the app. See Package a mobile app on page 1175.
   The package for the debug version of the app is created and ready to be deployed.

**Note:** If you install the debug version of the app on an iOS device, you must then enable the debug port in the app settings. For further information, see the Genero Business Development Language User Guide.

**Run the debug version of a deployed app (Android™)**

After deploying the app to your device, you must start the app in order to use the debug tools.

**Before you begin:**
- You must deploy the debug version of the app to the device.
- You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

1. On the device, launch the debug version of the app.
2. If the debug service is running on the device, stop it.
   - If the debug service is running, the red Genero Mobile debug service icon appears in the status bar.
     a) Drag down from the top of the screen to display a menu.
     b) Below the Genero Mobile debug service entry, tap **Cancel**.
        The red Genero Mobile debug service icon no longer appears in the status bar.
3. Set **Debug service** to **ON**.
   The red Genero Mobile debug service icon appears in the status bar.
4. Tap **Browse bundled apps**.
5. Tap **InternalGeneroApps**.
6. In the list, find the two entries for your app.
   - The two entries are file names: *programname.42r* and *packagenodename.xcf*.
   **Note:** If the package node name and the program name are identical, only one entry (*programname.xcf*) displays.
7. Tap on either file name to start the app.
   The debugger starts and you can debug your application.

**Related tasks**

Create a debug version of a deployed app on page 615
By default, the mobile app is built in release mode, which cannot be debugged. To build an app in debug mode, you must check the **Mobile Debug Package** environment set, which sets the DEBUG_PACKAGE environment variable to 1.

Run a debug version of a deployed app (iOS)
After deploying the debug version of the app to your device, you must start the app in order to use the debug tools.

**Before you begin:**
- You must deploy the debug version of the app to the device.
- You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

With Genero Mobile for iOS, the debug version of a deployed app enables the debug preferences by enabling port 6400 for debugging.

1. On the device, launch the debug version of the app.
2. In Genero Studio, select **Debug > Attach to Mobile Process**.
   The debugger starts and you can debug your application.

If you have trouble connecting, these are things you can check in the application's settings:
• The debug port check box must be selected. You should not need to enable the debug port as it is checked by default.
• The debug port in the application’s settings must be set to 6400, otherwise Genero Studio will not be able to connect to it.

**Related tasks**
Create a debug version of a deployed app on page 615
By default, the mobile app is built in release mode, which cannot be debugged. To build an app in debug mode, you must check the Mobile Debug Package environment set, which sets the DEBUG_PACKAGE environment variable to 1.

**Debug tools for apps in developer mode**
Some debug tools are only available when running your app in developer mode.
These debug tools are only available for apps running in developer mode.

- Preview a form on a mobile device on page 617
- Run an app with the graphical debugger on page 617
- Run an app with the Profiler on page 618

These debug tools can be used for apps running in developer mode.

- View the AUI tree (Android) on page 619
- View program logs (Android) on page 621
- View device logs (Android) on page 622

**Preview a form on a mobile device**
When working in developer mode, you can preview your form on a mobile device.

**Before you begin:**
Select the appropriate Genero Studio configuration for displaying to your mobile device or emulator, and have your app project open in Genero Studio.

1. Open a form specification file (4fd, 4fdm, or per).
2. Select **Build > Preview**.

The form displays on your device.

**Run an app with the graphical debugger**
When working in developer mode, you can use the graphical debugger to step through your code as your app executes on your mobile device.

**Before you begin:**
Select the appropriate Genero Studio configuration for displaying to your mobile device or emulator, and have your app project open in Genero Studio.

1. Open an app source (4gl) file.
2. In the source (4gl) file, use **Debug > Add/Delete Breakpoint** to set your breakpoints.
3. Select **Debug > Debug**.
   
   The app starts on your device.
4. Use the graphical debugger icons to step through your code.
   
   For more information on using the graphical debugger, see the *Genero Studio User Guide*.

**Run an app with the Profiler**
The Profiler is a tool built in the runtime system that generates a report about where the program spends time, and which function calls which function. The Profiler can help to identify areas in the program that are slower than expected.

**Before you begin:**
Select the appropriate Genero Studio configuration for displaying to your mobile device or emulator, and have your app project open in Genero Studio.

For more information on using the Profiler, see the *Genero Studio User Guide*.

**Note:** When in developer mode, the DVM is running on the desktop. The Profiler is therefore providing a profile of the app as it runs on the desktop. It may not be a true representation of how it will perform when deployed to the device.

1. Set the desired app as the default app.
2. Select **Debug > Execute with Profiler**.
   The app starts on your device.
3. Use the app.
4. End the app.
   The Profiler report appears in the Output panel.

**Viewing the AUI tree**

You can view the AUI tree in a browser. The AUI tree is the tree of the program's objects and user interface elements, and can be used to see what is present in the AUI tree for the program and the values of each element's properties.

The app must be running in developer mode or be the debug version of a deployed app.

- View the AUI tree (Android) on page 619
- View the AUI tree (iOS) on page 620
- View the AUI tree (Development Client) on page 621

**View the AUI tree (Android™)**

You can view the AUI tree for your Android™ mobile app in a browser.

Before you begin:

- Your app must be able to run in development mode or be the deployed debug version of the app.
- You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

1. If you are going to run your app in developer mode, complete these steps.
   a) From Genero Studio, launch your application.
   b) Select **Tools > Android Tools > Show AUI Tree**.
      The AUI Tree displays in a browser.
   c) Click on a node in the tree to highlight the corresponding item on the device or emulator.

2. To view your app in a browser while running the deployed debug version of your app using Genero Studio, complete these steps.
   a) To run the app from your device, launch your app in debug mode. See Run the debug version of a deployed app (Android) on page 616.
      The application launches and the initial form displays.
   b) If you are using Genero Studio, select **Tools > Android Tools > Show AUI Tree**.
      The AUI Tree displays in a browser.
   c) Click on a node in the tree to highlight the corresponding item on the device or emulator.

3. To view your app in a browser while running the deployed debug version of your app, complete these steps.
   a) To run the app from your device, launch your app in debug mode. See Run the debug version of a deployed app (Android) on page 616.
      The application launches and the initial form displays.
   b) Swipe down on the icon bar and locate the entry for the Debug service. Make a note of the URI address provided.
      The URI takes the form of `http://<device_ip_address>:<port_number>`.
   c) Open a browser and enter in the URI from the previous step.
d) Click AUI tree. The AUI Tree displays in a browser.
e) Click on a node in the tree to highlight the corresponding item on the device or emulator.

**Important:** The AUI tree does not automatically update in your browser as you move around the app. As you navigate the app on your device, you must reload your browser to view the changes in the AUI tree.

**View the AUI tree (iOS)**
You can view the AUI tree for your iOS mobile app in a browser.

**Before you begin:**
- Your app must be able to run in development mode or be the deployed debug version of the app.
- You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

1. If you are going to run your app in developer mode, complete these steps.
   a) From Genero Studio, launch your application.
   b) Select **Tools > iOS Tools > Show AUI Tree**. The AUI Tree displays in a browser.
   c) Click on a node in the tree to highlight the corresponding item on the device or simulator.

2. If you are going to run the deployed debug version of your app, complete these steps.
   a) On the iOS device, open Genero Mobile.
   b) In the **INFORMATION** section, note the **HTTP URL** provided.
       For example, **HTTP URL=http://192.168.0.160:6400**
   c) Launch your application in debug mode. See [Run a debug version of a deployed app (iOS)](page 616) on page 616.
   d) Swipe down to view details about the icons in the status bar.
       The description for the Genero Mobile debug service displays the URL you must use to access debug details. It takes the form **http://device-ip-address:port-number**.
   e) Enter the URL in a browser.
   f) Click AUI tree. The AUI Tree displays in a browser.
   g) Click on a node in the tree to highlight the corresponding item on the device or simulator.

3. If you are going to run the deployed debug version of your app, and your machine (running Genero Mobile) and device (running the app) are on the same wireless network, you can use Genero Studio to view the AUI tree.
   a) On the iOS device, open Genero Mobile.
   b) Select **Tools > Genero Configurations**.
   c) Select your iOS configuration from the list.
   d) Click the Edit icon next to the **Use Display Client** field.
       The Display Client management window opens.
   e) In the **Host** field, enter the ip address of your device. Click OK until all configuration management windows close.
   f) Launch your application in debug mode. See [Run a debug version of a deployed app (iOS)](page 616) on page 616.
   g) From Genero Studio, select **Tools > iOS Tools > Show AUI Tree**. The AUI Tree displays in a browser.
   h) Click on a node in the tree to highlight the corresponding item on the device or simulator.

**Important:** The AUI tree does not automatically update in your browser as you move around the app. As you navigate the app on your device, you must reload your browser to view the changes in the AUI tree.
**View the AUI tree (Development Client)**

While viewing your app in the Genero Development Client, you can view the AUI tree for your mobile app in a browser.

Before you begin, you should have verified your configuration within Genero Studio to display to the Genero Development Client.

The Genero Development Client allows the developer to view the AUI tree on the default port 6400.

1. On your iOS device, launch the Genero Development Client.
2. Find the **HTTP VIEWING URL**. Make a note of the URL.
   
   **Note:** The **HTTP VIEWING URL** displays in the Development window. If you do not see the development window, press the **Develop** button.
3. In Genero Studio, set your configuration for the **iOS Dev Client** and start your app.
4. In the Genero Development Client interface on your iOS device, click **Connect**.
5. Open a browser and enter in the **HTTP VIEWING URL**.
   The GMI Information page displays.
6. Click **AUI Tree**.
   The AUI Tree displays.

As you navigate the app on your device, you must reload your browser to view the changes in the AUI tree.

**Viewing the program logs**

You can view the program logs in a browser.

- [View program logs (Android)](page 621)
- [View program logs (iOS)](page 622)
- [View program logs (Development Client)](page 622)

**View program logs (Android™)**

You can view the program logs for your Android™ mobile app in a browser. The output displays VM messages (standard output and standard error).

Before you begin:

- Your app must be the deployed debug version of the app. You cannot view the standard output and error logs when running an app in developer mode.
- You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

1. To view the standard output and error logs in a web browser:
   a) To run the app from your device, launch your app in debug mode. See [Run the debug version of a deployed app (Android)](page 616).
      The application launches and the initial form displays.
   b) Swipe down on the icon bar and locate the entry for the Debug service. Make a note of the URI address provided.
      The URI takes the form of `http://<device_ip_address>:<port_number>`.
   c) Open a browser and enter in the URI from the previous step.
   d) Click **VM Output**.
      The standard output and standard error messages display.
2. To view using the **Display Standard output and error** menu option:
   a) To run the app from your device, launch your app in debug mode. See [Run the debug version of a deployed app (Android)](page 616).
      The application launches and the initial form displays.
b) Select **Tools > Android Tools > Display Standard output and error**.
   The program logs (standard output and standard error messages) are written to the Output view.

c) To stop the program logs from appearing in the Output view, select **Tools > Android Tools > Stop display Standard output and error**.

**View program logs (iOS)**

You can view the program logs for your iOS mobile app in a browser.

Before you begin:

- Your app must be able to run in development mode or be the deployed debug version of the app.
- You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

**Important:** Sensitive and personal data may be written to the output. Make sure that the log output is written to files that can only be read by application administrators.

1. On the iOS device, select **Settings > Genero Mobile**.
2. Enable GUI LOGGING.
3. Open Genero Mobile (GMI).
4. In the **INFORMATION** section, note the **HTTP URL** provided.
   For example, **HTTP URL=http://192.168.0.160:6400**
5. On your development machine, open your project in Genero Studio.
6. Start your app.
7. Open a browser and enter the **HTTP URL** identified in a previous step.
8. Click **Logs**.
   The program logs display in the browser.

**View program logs (Development Client)**

While viewing your app in the Genero Development Client, you cannot view the program logs in a browser.

The Genero Development Client acts as a remote GUI client, and therefore does not have logging options for the client. In order to get logs for an app displayed in the Genero Development Client, you need to enable the virtual machine's **--start-guilog** option when starting the app.

**Important:** Sensitive and personal data may be written to the output. Make sure that the log output is written to files that can only be read by application administrators.

See **Front-end protocol logging** in the *Genero Business Development Language User Guide* for details regarding the **--start-guilog** option.

**Viewing the device logs**

You can view the device logs in a browser.

The app must be running in developer mode or be the debug version of a deployed app.

**Note:** At this time, only Genero Mobile for Android™ offers this feature.

- **View device logs (Android)** on page 622

**View device logs (Android)**

You can view the Android device logs in a browser on your development machine.

Before you begin:

- Your app must be able to run in development mode or be the deployed debug version of the app.
- You must physically connect the device to your computer using a USB cable. Wireless debugging from within Genero Studio is not supported.

1. If you are going to run your app in developer mode, complete these steps.
a) From Genero Studio, launch your app.
b) Select Tools > Android Tools > Show Device Logs.
The device logs display in the Output view. New log entries will be appended in real time.

2. If you are going to run the deployed debug version of your app, complete these steps.
a) To run the app from your device, launch your app in debug mode. See Run the debug version of a deployed app (Android) on page 616.
The application launches and the initial form displays.
b) Swipe down on the icon bar and locate the entry for the Debug service. Make a note of the URI address provided.
The URI takes the form of http://<device_ip_address>:<port_number>.
c) Open a browser and enter in the URI from the previous step.
d) Click Android Logcat.
The device logs display.

Debugging a Web Component
You can debug a web component on a Genero Mobile application.

- Debug a web component on an Android device on page 623
- Debug a web component on an iOS device on page 623

Debug a web component on an Android device
You can enable the debugging of a Web component through Chrome.

This procedure requires:
- GMA 1.1 or greater.
- Android 4.4 or greater.

You need to enable the GMA debug mode and follow the instructions provided at http://developer.chrome.com/devtools/docs/remote-debugging#configure-webview

1. Connect your device to your desktop with a USB cable.
2. If GMA is running on your device, stop GMA on your device.
3. Enable USB debugging on the device.
4. Run your application with a web component.
   Tip: Please be patient as the application may take longer to start.
   GMA starts with debug service enabled.
5. Open Chrome on your desktop.
6. In the Chrome address bar, enter chrome://inspect/#devices.
7. Check Discover USB devices.
8. Accept the USB debugging on your device.

A list of URLs appear. You can inspect the individual pages.

Debug a web component on an iOS device
This configuration allows you to debug a web component on an iOS device.

For GMI, you can only debug with the iOS simulator. You cannot debug on the physical device.

1. Start the app.
2. Navigate to the webview you want to debug.
3. Open Safari.
4. Make sure the Show Develop menu in toolbar is enabled in Preferences > Advanced.
5. Go to Develop > iOS Simulator and choose the html to debug.
Localize a mobile app

Follow this procedure to localize your mobile app.

Before completing this procedure, you have an app that works but is not yet localized.

You can use localized string files to provide different languages for your mobile app. Before you complete the steps in this procedure, it would benefit you to read on the use of localized strings in the *Genero Business Development Language User Guide.*

**Important:** An app must be deployed to see localization take place. Localization does not work in developer mode.

1. Modify your app source and resource files for localized string use.
   a) For your text-based source files (4gl and per), see the article *Localized strings in program sources* in the *Genero Business Development Language User Guide.*
   b) For your graphical form files (4fd), see *Localizing your form* on page 527.
   c) For your resource files (such as your action defaults (4ad) and presentation styles (4st) files), see the article *Localized strings in XML resource files* in the *Genero Business Development Language User Guide.*

2. Create the localized string files.
   a) For your text-based source files (4gl and per), see the article *Extracting strings from sources* in the *Genero Business Development Language User Guide.*
   b) For your graphical form files (4fd), see *Localizing your form* on page 527.
   c) For your resource files (such as your action defaults (4ad) and presentation styles (4st) files), an extraction tool is not provided. You must create the localized string file by hand.

You now have a localized string file for each of your source files. These files serve as the starting point for language-specific localized string files; you will create a copy these files for each language you wish to provide.

**Tip:** Save these base localized string files in your Resources folder in your project. You do not, however, want them to be compiled. Select each of the base localized string files and check the Exclude from compilation property.

3. Update your fglprofile file to list the required string files.
   See *Using localized strings at runtime* in the *Genero Business Development Language User Guide.*

4. Identify the languages and their corresponding locale codes needed for your localization effort.
   The selected language is identified by a locale code following the ISO 639 standard. See *Using localized strings at runtime* in the *Genero Business Development Language User Guide.*

5. Create a sub-directory for each language you wish to localize.
   For the directory name, use the locale code identified in the previous step.

6. Copy all the localized string files into each locale sub-directory.

7. Edit the localized string files within each locale sub-directory to provide the translations. Save your changes.

8. Compile each localized string file.
   If you are using the command line to compile the localized string files, see *Compiling string files* in the *Genero Business Development Language User Guide.* If you are using Genero Studio to compile the localized string files, follow these steps.
   a) Create a library node for each language.
      For example, you could create a library named en to hold your generic English localized string files, and you could create a library named en_US to hold your English (United States) localized string files.
   b) Select the library node and change the **Target directory** property to $(ProjectDir)/bin/xx, where xx is the language code.
      For example, you would specify $(ProjectDir)/bin/en for the generic English library node, or $(ProjectDir)/bin/en_US for the English (United States) library node. The target directory specifies where the compiled string files will be placed after compilation. These examples assume that the compiled program files are being placed in $(ProjectDir)/bin.
c) Right-click on the library node and select Add Files to add your localized string files (.str) from the corresponding language directory on disk.
d) Right-click on the library node and select Build.
   The compiled localized string files (.42s) appear in the specified Target directory.
e) Right-click on the Application node and select Advanced Properties.
f) Select the Dependencies page and select each language library in the Dependencies property page.

9. For each package node, add a Directory node for each language library node.
   The Directory node places the compiled string files into the correct directory. For example, if the Source directory is set to $(ProjectDir)\bin\en_US, then the Destination directory must be set to $(ProjectDir)\bin\en_US. Set the Included files filter to *.42s.

10. Deploy your package.
11. Test each language.
   a) Change the language in the device's settings to one of the languages you have set for localization.
   b) Launch your app.

**Genero Mobile error messages**

A list of Genero Mobile error messages. For messages that are not self-explanatory, additional information is provided.

**Table 288: Genero Mobile error messages**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-09002</td>
<td>File path issue. Path is incorrect or file is missing.</td>
</tr>
<tr>
<td></td>
<td>Cannot package file : $filePath source and destination relative path don't match</td>
</tr>
<tr>
<td></td>
<td>Destination directory not in rootDir</td>
</tr>
<tr>
<td></td>
<td>Cannot compute zip directory from source dir</td>
</tr>
<tr>
<td></td>
<td>Error updating application archive '$appZip'</td>
</tr>
<tr>
<td></td>
<td>Error compressing application archive '$appZip'</td>
</tr>
<tr>
<td></td>
<td>Error adding $packageName\xcf to application archive '$appZip'</td>
</tr>
<tr>
<td>GS-09003</td>
<td>Bad script arguments such as an incorrect build/package/deploy/run rule.</td>
</tr>
<tr>
<td></td>
<td>Missing argument apkFilePath</td>
</tr>
<tr>
<td></td>
<td>Missing argument deployData</td>
</tr>
<tr>
<td></td>
<td>Missing argument projectDir</td>
</tr>
<tr>
<td></td>
<td>Missing argument architecture</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-09004</td>
<td>Genero configuration settings errors.</td>
</tr>
<tr>
<td></td>
<td>Missing dynamic property '$propertyName'</td>
</tr>
<tr>
<td></td>
<td>Missing environment variable GMADIR</td>
</tr>
<tr>
<td></td>
<td>Cannot find Genero Mobile for Android package, check GMADIR environment variable</td>
</tr>
<tr>
<td></td>
<td>Missing environment variable ANDROID_HOME</td>
</tr>
<tr>
<td></td>
<td>Provisioning profile '$provisioningProfile' doesn't exist: The path contained in PROVISIONING_PROFILE variable is not valid.</td>
</tr>
<tr>
<td></td>
<td>Provisioning profile '$provisioningProfile' is not a provisioning profile: The path contained in PROVISIONING_PROFILE variable is not a provisioning profile (mobileprovision) file.</td>
</tr>
<tr>
<td></td>
<td>Copy of provisioning profile '$provisioningProfile' failed: The copy of provisioning profile to package has failed.</td>
</tr>
<tr>
<td></td>
<td>Invalid value for DEBUG_PACKAGE variable (should be an integer superior or equal to 0). DEBUG_PACKAGE value is invalid. It should contains the debug level (a positive integer).</td>
</tr>
<tr>
<td></td>
<td>Cannot find Genero Mobile for iOS package, check GMIDIR environment variable. Path contained in GMIDIR is not valid.</td>
</tr>
<tr>
<td>GS-09005</td>
<td>Cannot read deployment data file '$deployData'</td>
</tr>
<tr>
<td>GS-09006</td>
<td>Error parsing deployment data file '$deployData'</td>
</tr>
<tr>
<td>GS-09007</td>
<td>Informational messages.</td>
</tr>
<tr>
<td></td>
<td>Deploy to $deviceId started</td>
</tr>
<tr>
<td></td>
<td>Deploy started</td>
</tr>
<tr>
<td></td>
<td>Deploy finished</td>
</tr>
<tr>
<td></td>
<td>Starting emulator</td>
</tr>
<tr>
<td></td>
<td>Waiting for $deviceId to respond</td>
</tr>
<tr>
<td></td>
<td>Emulator started</td>
</tr>
<tr>
<td></td>
<td>Starting Genero Mobile for Android on $deviceId</td>
</tr>
<tr>
<td></td>
<td>Forwarding android port 6400 to localhost:$displayClientPort</td>
</tr>
<tr>
<td></td>
<td>Starting Genero Mobile for Android Genero Mobile for Android started</td>
</tr>
<tr>
<td>GS-09008</td>
<td>Errors in package node in project file (4pw).</td>
</tr>
<tr>
<td></td>
<td>Invalid package name '$packageName': Use alpha numeric characters only.</td>
</tr>
<tr>
<td></td>
<td>Invalid package ID '$packageId': Follows java package naming: com.company.product</td>
</tr>
<tr>
<td></td>
<td>Invalid packageLabel '$packageLabel': Package label cannot be an empty string.</td>
</tr>
<tr>
<td></td>
<td>Invalid package version '$packageVersion': Package version must be integer value.</td>
</tr>
<tr>
<td>GS-09009</td>
<td>No module (.42r) in package</td>
</tr>
</tbody>
</table>
BAM Template Developer Guide

Genero Studio comes with a set of templates used with the Business Application Modeler (BAM) for the code generation. You are free to customize these templates or create your own templates so that the generated application meets your requirements.

You can customize and configure Genero Studio and the Business Application Modeler to generate the application you want to build, the way you want to build it.

For example you may find that instead of adding POINT or BLOCK code repeatedly for each application, you want to use a template that generates your customized code. Or, perhaps you want to add custom properties to a modeling diagram.

Options for modifying templates or creating new templates can be categorized in this order of complexity.

**Table 289: Options for modifying templates**

<table>
<thead>
<tr>
<th>Option</th>
<th>Types of modifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify a generated application without touching templates.</td>
<td>• Modify diagram property values.</td>
</tr>
<tr>
<td></td>
<td>• Use POINT/BLOCKs to modify generated code. See Using POINTs and BLOCKs on page 313.</td>
</tr>
<tr>
<td></td>
<td>• Use resource files (styles, action defaults). See Modify action defaults on page 318, Modify styles on page 318, Modify the Topmenu on page 319, Modify the Toolbar on page 320.</td>
</tr>
</tbody>
</table>
Option | Types of modifications
--- | ---
Modify the default template files. | • Add, remove, or change diagram properties
• Change generated 4gl code
Customize template files. | • Change build rules
• Create new file types
• Create new File > New options
• Customize Genero Studio diagrams. See Example 1: Adding a new property on page 1095, Example 3: Adding an entity to the BA Diagram on page 1101
• File templates, code templates, user action

Quick Start: Customizing templates

This Quick Start uses examples to illustrate how to customize template files.

Modifying the default templates requires a basic knowledge of Genero Business Development Language and the Tcl language or another template language. Genero Application Generator uses the Tcl Generator by default; it can be replaced by a generator of your choice such as XSLT or perl.

Example 1: Adding a new property

In this example, the BA diagram is modified to include a new property. When a value is set in the new property, the modified tcl script generates the code to include the information from the property.

Copy the default template set

Create a new template set from which applications can be generated.

1. Copy the entire content of the template directory (for example, GSTDIR/gst/bin/src/ag/tpl/dbapp) to a new directory. The new template directory could be a common one on the server or it could be part of the entire versioned project.

   You may also create a new empty directory to which you can add the template files one at a time as you create them.

   Note: Ensure that you have write access to the new directory.

2. Determine which configuration you want to use to run your generated applications. Select the configuration. Then, select Tools > Genero Configurations.

3. Find and select the Template Environment Set. Deselect the check box to deactivate the old template.

4. Make a copy of the Template environment set using the Duplicate action in the integrated Toolbar.

5. Rename your template environment if you wish by right-clicking on it and selecting Rename.

6. Select your template environment set. Notice that there are environment variables set.

7. Modify GSTSETUPDIR. Double-click GSTSETUPDIR in the list and modify the Value to point to the location of your template files.

8. Select OK.

9. In the Environment Sets list, select the check box for your environment set.
Figure 435: Select new environment set

Now you can customize the files that you copied into the new directory. Keep the same file names as the default template set. All your programs will be generated with these files; the default files will be ignored.
Add a new property to the BA diagram
This example illustrates how to add a new property to the Business Application diagram by adding a new icon property to the diagram template.

Figure 436: New property in Program entity properties

1. From Genero Studio, open the settings.agconf file in your template directory.
3. In the <Items> child element, modify the Program item to include a dynamicProperties attribute with a value of `icon`.

```xml
<Program>
  <dynamicProperties>
    <icon>smiley</icon>
  </dynamicProperties>
</Program>
```
4. In the <DynamicProperties> section of the <BusinessApplication> element, add a new <DynamicProperty> element to define the new icon dynamic property.

```xml
<DynamicProperty name="icon"
    type="TEXT"
    label="Icon"
    description="Defines icon to be used in window title bar of running application."/>
```

5. Save your changes.
7. Open the OfficeStore.4pw sample project from your My Genero Files directory.
8. Open the OfficestoreAppFlow.4ba diagram and select the Account or Orders program entity on the diagram.
9. Note the new Icon property in the Properties view.

Related concepts
Interpreting settings.agconf on page 1111
The settings.agconf file is an XML document with four sections: Form, Business Record, Business Application, and Database. It is used to manage the properties available to each of the diagrams used by the Business Application Modeler.

Modify tcl script to generate code
Modify the tcl script used to generate the code.
1. Open the main.tcl file in the tpl subdirectory of your template set directory.
2. Find the # Set XPath set of code. On the line following set ROOTFILELocator, add:

```plaintext
set icon [[getUniqueNode $ROOTNode {/AG/File/DynamicProperties/ DynamicProperty [@name='icon']} @value]]
```

3. Find the instruction `[gen_MAIN $DBName $outFormRelNode]` and add the `$icon` parameter:

```plaintext
[gen_MAIN $DBName $outFormRelNode $icon]
```

4. Find the instruction `proc gen_MAIN {DBName outFormRelNode} {` and add the icon parameter:

```plaintext
proc gen_MAIN {DBName outFormRelNode icon} {
```

5. Find the instruction `CALL ui.Interface.loadStyles("dbapp")` and on the following line add the call to set your image:

```plaintext
CALL ui.Interface.setImage("$icon")
```

6. Save the changes.
7. Select Tools > Specific setup > Reload.
8. In the OfficeStoreAppFlow.4ba, select the Account program entity and set the value of the Icon property to a valid image name. See Image directory structure on page 1121.
9. Save the files.
10. Rebuild and run the Account program and see the icon used in the window title of the program.

![Figure 438: Custom icon in window title](image-url)

### Example 2: Adding a File > New item

In this example, the File > New dialog is modified to include a new, custom item.

**Add a new file type definition to settings.agconf**

Modify the settings.agconf file to include a new file type based on the 4dbx database schema file type.

1. Open the settings.agconf file in your template directory.
2. Find the Database section. Notice that there is a File element defining the 4dbx file type. Copy and paste this File element so that you have an additional File element in your Database element.
3. Modify the extension attribute to use your own database extension, for example 4dbz.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<AGSettings version="5">
  <Form> </Form>
</AGSettings>
```
4. Save your files.

**Define an action in File > New for the new item**

Modify the creatables.conf file to define the new item in File > New.

1. Open the creatables.conf file in your template directory.
2. Notice that the Category elements organize the creatables into groups. Find the Category element with a name attribute of Database. Modify the File element to represent the newly defined database file by changing the label, extension, and source attributes.

   `<Category index="30" label="Database" name="Database">
   <File index="30" name="DBXNewERD" action="DBOpen"
     label="DB Schema (.4dbz)"
     description="Create a new DB schema"
     icon="document_newSchema"
     extension="4dbz"
     source="creatables/dbapp.4dbz"
     isTemplate="true"/>
   </Category>`

3. Save the changes.
4. Since the source element was changed to creatables/dbapp.4dbz, create that file in the creatables subdirectory in your template directory. Copy and rename dbapp.4dbx to dbapp.4dbz.
**Test new action**
View the new addition to the New dialog.

1. Launch Genero Studio.
2. Select **Tools** > **Specific setup** > **Reload**.
3. Select **File** > **New** and select your new database file type from the **Database** category.

![Select an item](image)

**Figure 439: New item**

**Example 3: Adding an entity to the BA Diagram**
In this example, the BA diagram is modified to include a new, custom item.

**Add a new item to the BA diagram**
Create a new Program entity called Start in the BA diagram.

1. Open the **settings.agconf** file in your template directory.
2. Find the **BusinessApplication** section. Add a new **DynamicProperty** and **Item**. In the **Item** element, define a new Program item with a **name**, **extension**, **label**, **icon** and **dynamicProperties** attribute values of your choice. The value of the **dynamicProperties** attribute corresponds with the **name** attribute in the **DynamicProperty** element.

```
<BusinessApplication>
  <DynamicProperties>
```

3. Save the changes.
4. Select Tools > Specific setup > Reload to reload the modified template file.
5. Select the Library node in your project and right-click. Select New > Business Application Diagram (4ba) to create a new 4ba.
6. Right-click on the BA diagram to see and select a new Start item. Save the 4ba file to your project directory (not the template directory).

The item cannot yet be implemented because the new file type (4srt) has not yet been defined.

Define the template file association
Add a new file type for the new program type.

1. Select Tools > Specific setup > Edit File Associations.
2. Define the file type and associated action. The mimeType and extensions attributes must match what you defined in your settings.agconf file.
3. Select OK to save the changes.

Add the new element to the program entity
Add the new element to the program entity.
1. Open the `creatables.conf` file in your template directory.
2. Find the Managed Code category and add a new `New` element to represent your new program entity.

```xml
<New
    index="900"
    name="CustomProgram"
    action="BANewProgram"
    label="Custom Program (.4srt)"
    actionLabel="Implement Custom Program"
    icon="document_4prg"
    description="Create Program"
    extension="4srt" />
```

3. Save the changes.
4. Select Tools > Specific setup > Reload to reload the modified template file.
5. Close and re-open the new Business Application diagram (4ba) in the Library node of your project.
6. Right-click on the Start entity you added to your program (or add one). Notice the action to implement your program now exists in the menu. Select this action.

![Implement Custom Program](image.png)

**Figure 441: Implement new program entity**

7. Save the `.4srt` file to the Application node in your project.

**Define the build rules**

Define the build rules for how the new file type is to be compiled.

1. Select Tools > Specific setup > Edit Build Rules.
2. Use the duplicate button to make a copy of the 4PRG build rule.
3. Edit the build rule to match your new program definition (.4srt).
Figure 442: Creating a new build rule

4. Save.

5. If your modifications include an image other than the default, copy the image file to the folders in the GSTDIR/images directory. See Image directory structure on page 1121.

6. You can now use your new 4SRT entity in your BA diagrams.

Example 4: Configuring a template to use an old FGL version

In this example, your application was created using an old version of FGL. The BAM templates need to be rebuilt to avoid incompatibility of code.

Change FGL version
Create a configuration that uses a different version of FGL.

1. Select Tools > Genero Configurations.
2. In the Configuration Name section, duplicate and rename an existing configuration.
3. Right-click on the new configuration, and select Set Genero configuration active.
4. In the Compiler/Runtime section, click the icon to edit the installation.
5. In the Genero install management dialog box, duplicate and rename an existing installation.

   Tip: Give the installation a meaningful name, for example, "FGL 2.50".

6. Edit the FGLDIR variable to point to the location of the FGL version, for example, C:\Program Files\FourJs\Genero Studio\2.50\fgl.
7. Click OK to exit the Genero install management dialog box.
8. Associate the new configuration name with the new genero installation.
Copy the default template set
Create a new template set from which applications can be generated.

1. Copy the entire content of the template directory (for example, GSTDIR /gst/bin/src/ag/tpl/dbapp) to a new directory. The new template directory could be a common one on the server or it could be part of the entire versioned project.

   You may also create a new empty directory to which you can add the template files one at a time as you create them.

   *Note:* Ensure that you have write access to the new directory.

2. Determine which configuration you want to use to run your generated applications. Select the configuration. Then, select Tools > Genero Configurations.

3. Find and select the Template Environment Set. Deselect the check box to deactivate the old template.

4. Make a copy of the Template environment set using the Duplicate action in the integrated Toolbar.

5. Rename your template environment if you wish by right-clicking on it and selecting Rename.

6. Select your template environment set. Notice that there are environment variables set.

7. Modify GSTSETUPDIR. Double-click GSTSETUPDIR in the list and modify the Value to point to the location of your template files.

8. Select OK.

9. In the Environment Sets list, select the check box for your environment set.

![Figure 443: Select new environment set](image)

Now you can customize the files that you copied into the new directory. Keep the same file names as the default template set. All your programs will be generated with these files; the default files will be ignored.

Copy the studio libraries
Create a copy of the studio libraries.

1. Copy the entire content of the studio libraries directory (for example, GSTDIR /gst/bin/src/studio/lib) directory to a new directory.

   *Note:* Ensure that you have write access to the new directory.

2. Select Tools > Genero Configurations, and select the configuration name.

3. Find and select the Studio Libraries Environment Set. Deselect the check box to deactivate the old studio libraries.


5. In the new studio libraries environment set, double-click the GSTLIBRARYDIR environment variable.
6. Modify the value of GSTLIBRARYDIR to point to the location of your new studio library files, and click **OK**.
7. In the Environment Sets list, select the check box for your studio libraries.

![Environment Sets](image)

**Figure 444: Select new studio libraries environment set**

**Compile template and libraries**

Rebuild the template and studio libraries to use the selected FGL version.

1. Go to the template directory and open the project file, for example, libdbapp.4pw.
2. Right-click on the application node and select **Rebuild All**.
3. Go to the studio library directory and open the project file, for example, libgst.4pw.
4. Right-click on the application node and select **Rebuild All**.

You can now open and build projects using the selected FGL version.

---

**How code is generated**

When you build an application from a Business Application diagram, the build rules define the various files that are input into the Code Generation Engine and the application code files that are output.

![Code Generation flow](image)

**Diagram:**
- DB, BA, Forms, Reports...

**Code Generation flow:**
- Code (Pass 1)
- Code (Final)
- Custom code

**Figure 445: Code Generation flow**
**BAM Consolidates and Generates**

**Data Consolidation**

The input from the BA diagram and related entities is gathered into a single XML file, which consolidates all the inputs into one package. This file is used when processing and generating the application code. This file could also be used to provide input to create the application models.

**Code Generation**

The XML file and a code template are used to generate the application code. The default Tcl template produces Genero 4gl files, but another tool could be used to generate the code (XSL translator for example). Custom code is preserved; if any custom code was created earlier, it is automatically restored in the newly generated application code.

**Example**

The build rules define the series of commands used to build and generate the code. In general, the Build rule for code generation:

- Saves custom code added by the user to the generated source files
- Generates the new source files without user code
- Restores the user code in the generated files
- Compiles the written and generated source files
- Links the compiled files

View default build rules selecting **Projects > Edit Build Rules**. This example shows the build rules used to generate the code for a Program entity in the Business Application diagram.
Table 290: Default build rule example

<table>
<thead>
<tr>
<th>Build rule command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(generate)</td>
<td>The $(generate) command creates an intermediary XML file from modeled entities.</td>
</tr>
<tr>
<td>$(blockpoint) -code</td>
<td>BLOCK/POINT is extracted from previously generated and modified code.</td>
</tr>
<tr>
<td>tclsh on page 322</td>
<td>The tclsh executable generates the final file by using both a Tcl template file and the intermediary XML file created by the $(generate) command.</td>
</tr>
<tr>
<td>$(blockpoint) -storeGenerated</td>
<td>Extracted BLOCK/POINT code is put back into the generated code.</td>
</tr>
<tr>
<td>$(fglcomp)</td>
<td>The fglcomp tool compiles BDL program sources files into a p-code version.</td>
</tr>
</tbody>
</table>
Reviewing the Build

To better understand what is happening during the build of the program, turn on verbose mode using Tools > Preferences > Compiler and Runtime > Compilation Configuration. Compile a diagram file, program, or application and view the results in the output.

Related concepts

The code generation template set on page 1110
Genero Studio provides a standard template set of files that are used for code generation. The files are written in XML and Tcl.

The code generation template set

Genero Studio provides a standard template set of files that are used for code generation. The files are written in XML and Tcl.

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbapp4.1</td>
<td>This is the default and recommended template for Genero 3.10.</td>
</tr>
<tr>
<td>dbapp4.0</td>
<td>This is the default and recommended template for Genero 3.00.</td>
</tr>
<tr>
<td>dbapp3.1</td>
<td>This is the default and recommended template for Genero 2.51.</td>
</tr>
<tr>
<td>dbapp3.0</td>
<td>This is the default and recommended template for Genero 2.50.</td>
</tr>
<tr>
<td>dbapp2.0</td>
<td>This is the default and recommended template for Genero 2.41.</td>
</tr>
</tbody>
</table>

Figure 447: Code Generation flow

Default templates

It is recommended that you use the most current template set for your Genero installation.

Table 291: Application Generation Templates
<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbapp1.0</td>
<td>The dbapp template in Genero Studio 2.40 was renamed in Genero Studio 2.41 to dbapp1.0. The dbapp template set is the same template set as dbapp1.0. This is the default and recommended template for 2.40. Using the DIALOG instruction, sub-dialog statements that display data, query a database, and edit single records and record lists can execute in parallel, allowing the application to handle different parts of a form simultaneously.</td>
</tr>
<tr>
<td>2.3x</td>
<td>This is a legacy template for compatibility with applications developed with Genero Studio 2.3x.</td>
</tr>
</tbody>
</table>

The GSTSETUPDIR environment variable specifies which template set is used.

**Template directories**

The template sets are located at GSTDIR/bin/src/ag/tpl/. Each template set directory name starts with "dbapp" and ends with the template set version number.

The contents of template sets have changed as the template sets have developed. Most template sets include the files settings.agconf and creatables.conf.

**Tip:** When modifying a template set, do not modify the original template files. Instead, make a copy of the directory. Copy the entire content of the template directory (for example, GSTDIR/gst/bin/src/ag/tpl/dbapp) to a new directory. The new template directory could be a common one on the server or it could be part of the entire versioned project.

**Related concepts**

- Environment sets on page 164
  Define sets of environment variables to add to or overwrite the default environment. These environment settings are also usable by each Genero Studio client that adds a remote Genero Studio Server to its list.

- GSTSETUPDIR on page 168
  The GSTSETUPDIR environment variable defines the BAM application generator template directory. Changing this variable launches synchronization from the server and reloads the templates.

**Related tasks**

- Copy the default template set on page 1095
  Create a new template set from which applications can be generated.

**Interpreting settings.agconf**

The settings.agconf file is an XML document with four sections: Form, Business Record, Business Application, and Database. It is used to manage the properties available to each of the diagrams used by the Business Application Modeler.

There are four main sections corresponding to the four diagrams that can be customized:

- Form on page 1123
- BusinessRecord on page 1133
- BusinessApplication on page 1143
- Database on page 1153

In the default settings.agconf, the Database section, for example, shows the information about the dynamicProperties defined for the Meta-schema file type (4dbx). The dynamicProperties found in the Meta-schema diagram for a 4dbx file are widget and label. When an item in the diagram, specified by nodeName, is selected, the properties appear as assigned by dynamicProperty. These properties are grouped according
to the dynamicPropertyGroup element. These properties have a label as assigned by the label attribute and data type as specified by type attribute. The widget/wizard used is specified by elements such as editorInfo.

---

```xml
<Database>
    <File extension="4db">
        <DynamicProperties>
            <DynamicProperty name="widget" type="ENUM" label="Widget"
                initialValue="Edit" description="Associated widget"
                editorInfo="contains:ButtonEdit|CheckBox|ComboBox|DateEdit|Edit|
                FFImage|FFLabel|Field|ProgressBar|RadioGroup|Slider|SpinEdit|TextEdit|TimeEdit"/>
            <DynamicProperty name="label" type="TEXT" label="Label"
                initialValue="" description="Associated label when generating form"/>
            <DynamicPropertyGroup name="formfieldGroup" label="Formfield"
                description="Formfield properties" properties="label;widget"/>
        </DynamicProperties>
        <Items>
            <Item nodeName="column" dynamicProperties="widget;label"/>
        </Items>
    </File>
</Database>
```

---

**Figure 448: Label and Widget properties in Meta-schema diagram (4db)**

**Related concepts**

*settings.agconf elements* on page 1122

The settings.agconf file is an XML document with four sections: Form, Business Record, Business Application, and Database. It is used to manage the properties available to each of the diagrams used by the Business Application Modeler.

*DynamicProperty* on page 1126
The DynamicProperty element is a child element of DynamicProperties.

**The WEB custom editor**
Custom property editors are defined by the customer. The WEB custom editor is an HTML property editor.

**Property info definition**
The property info definition in settings.agconf:

- `type = "WEB"`
- `editorInfo:
  - htmlEditor: path to the html file relative to the template directory
  - isDebug: false: activates the webkit debug
  - title: title of the dialog (banner + window title)

**Example**

```xml
<DynamicProperty name="expression" type="WEB" label="expression"
  initialValue=""
  editorInfo="htmlEditor:expression.html;isDynamic:true;isDebug:false;title:Expression editor"/>
```

**HTML and JavaScript**

`property.value` implicitly contains the property value when the editor is opened. The value present in `property.value` is taken as a new property value when the user has validated the dialog (and `onEditorAccept()` is called).

The function `onDocumentSource(source)` is called once the html document is loaded in the webkit. The source contains the source of the GeneroStudio current document (the document which contains the property).

The function `onEditorAccept()` is called when the user presses the OK button to accept the value.

The function `onEditorCancel()` is called when the user refuses the value (by pressing cancel or the x window button).

**expression.html file**

This is an example of an expression HTML file, referenced by the earlier DynamicProperty editor definition example.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  </head>
  <script language="JavaScript" type="text/javascript">
    // onDocumentSource event is called when the custom editor is loaded
    // source : the complete document source in xml
    function onDocumentSource(source) {
      // find record fields list and populate the ui
      var list = document.getElementById("fieldlist")
      var parser = new DOMParser();
      var xmlDoc = parser.parseFromString (source, "text/xml");
      var recordNodes = xmlDoc.getElementsByTagName("RecordField");
      for (var i = 0 ; i < recordNodes.length ; i++) {
        var recordNode = recordNodes[i];
        var recordName = recordNode.getAttribute("name");
        // create one item for the recordNode field
        var option=document.createElement("OPTION");
      }
    }
  </script>
</html>
```
```javascript
option.text = recordName;
option.value = recordName;
list.add(option);

// populate the input with the expression
var elem = document.getElementById("expression");
elem.value = property.value;

function pressed(character)
{
    var elem = document.getElementById("expression");
    elem.value += character;
    property.value = elem.value;
}

function clearExpression()
{
    var elem = document.getElementById("expression");
    elem.value = ";
    property.value = elem.value;
}

function selectField(fieldname)
{
    var elem = document.getElementById("expression");
    elem.value += fieldname;
    property.value = elem.value;
}

// onEditorAccept event is called before validating the custom editor
function onEditorAccept()
{
    alert("editor accepted !");
}

// onEditorCancel event is called before validating the custom editor
function onEditorCancel()
{
    alert("editor cancelled !");
}
```

```html
<body width="200" height="50">
<center>
<form name="calc" action="">
<table>

<table>
<thead>
<tr>
<th>button</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>&quot;7&quot;</td>
</tr>
<tr>
<td>8</td>
<td>&quot;8&quot;</td>
</tr>
<tr>
<td>9</td>
<td>&quot;9&quot;</td>
</tr>
<tr>
<td>+</td>
<td>&quot;+&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;4&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;4&quot;</td>
</tr>
</tbody>
</table>
```
The PROCESS custom editor
Custom property editors are defined by the customer. The PROCESS custom editor is a standalone executable property editor.

Property info definition
The property info definition in settings.agconf:

- type="PROCESS"
- editorInfo:
  - processEditor: path to the process editor executable relative to the template directory.

Example

```xml
<DynamicProperty name="process" type="PROCESS" label="process" 
description="process" 
editorInfo="processEditor:userPropertyEditor.exe;isDynamic:true" 
dynamicContent="masterTable" />
```
**Executable**

The program’s first argument is the property value when the editor opens.

If the program outputs a window id to the standard output, Genero Studio will ensure the corresponding window remains on top of Genero Studio (similar to a modal dialog). The syntax is a single line:

```
WindowId: <Window Identifier>
```

Once the property editing is done, the program should output the property value, on a single line as well:

```
Value: <property value>
```

**Note:** Non-printable characters must be escaped.

**Interpreting creatables.conf**

File > New menu options can be defined in the `creatables.conf` file. Added items are called creatables.

Within the `creatables.conf` file there are Category elements. Category elements correspond to a group of actions (creatables). Each New, File, Directory, Wizard element corresponds to a file type that can be created using the File > New menu or context menu.

In the default `creatables.conf` file, one of the defined Creatables is the Module Form from Database (4fdm), specifying its category (Design), subcategory (Managed Code), and type of Creatable (Wizard).

**Example: Managed Form File Type**

```xml
<Creatables version="1.0">
  <Category index="5" label="Design" name="MDA" icon="document_4ba" >
    <Category index="40" label="Managed Code" name="ManagedCode">
      <Wizard index="10"
        name="FDModuleForm"
        action="FDNew"
        label="Module Form from Database (4fdm)"
        icon="document_4fdm"
        description="Create an empty module in Form Designer"
        extension="4fdm" />
    </Category>
  </Category>
</Creatables>
```
Figure 449: New Items: Creatables

Related concepts
creatables.conf elements on page 1163
A listing of all available elements in the creatables.conf file.

Related tasks
Define an action in File New for the new item on page 1100
Modify the creatables.conf file to define the new item in File > New.

Tcl basics and samples
A Tcl script is used to generate the final file using .tcl files and the intermediary .xml file generated during the build of the application. The tcl files can be modified to produce different 4gl code or to reference newly added properties.

Tcl (pronounced "tickle") is a scripting language commonly used for scripted applications and GUIs. You can use a scripting language of your choice or modify the included tcl files.

To determine which tcl file to modify turn on verbose mode for build, link, execution rules using Tools > Preferences > Compiler and Runtime. Then, compile the diagram file that you want to modify. Find tclsh in the resulting output to see which tcl files are used.

Add POINT and BLOCK sections to template
You can add new POINT and BLOCK areas to a template.

1. To add a POINT or BLOCK section, determine where the POINT or BLOCK should be in the generated code and modify the (Tcl) file that corresponds to the generated file (4gl).
To better understand what is happening during the build of the program, turn on verbose mode using Tools > Preferences > Compiler and Runtime > Compilation Configuration. You will be able to see what tcl files are being used and when. This can help you identify which tcl file you need to modify.

2. Add a new POINT or BLOCK section. Be sure to give the POINT or BLOCK a unique name that is not already used by another POINT or BLOCK.

3. Surround your added POINT or BLOCK area using the comment character of the targeted source code (such as "(" and ")" characters in Genero). Otherwise, the generated source code will not compile.

4. Open your project and rebuild.

5. To revert a change made in the source code, see Revert a change to a POINT or BLOCK on page 315.

Related concepts
POINT and BLOCK reference on page 1168
POINT and BLOCK sections that appear in the generated source code are defined within the template (Tcl) files. You can add additional POINT and BLOCK sections to the generated code.

Example: Using XSLT instead of TcI
The code for your application can be generated using XSLT instead of Tcl.
This example demonstrates how to generate a file with AG using XSL Translator.

Generating a form
Instead of generating 4gl as the template does, we'll generate a per file (Genero form definition file). This example uses a 4rd (report data) file and generate a per form file with formfields corresponding to the 4rd record. We'll add an additional property frmLabel on each record field, which will be displayed to the left of each per file formfield.

- The new property is added to the 4rd definition in settings.agconf (changes are in bold):

```
<File extension="4rd">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS" label="foreignFields"
      initialValue="" dynamicContent="srcFieldsContent"
      editorInfo="isDynamic:true" />
    <DynamicProperty name="primaryFields" type="FIELDS" label="primaryFields"
      initialValue="" dynamicContent="dstFieldsContent"
      editorInfo="isDynamic:true" />
    <DynamicProperty name="frmlabel" type="TEXT" label="Label FRM"
      initialValue="" />
  </DynamicProperties>
  <Items>
    <Item nodeName="Relation" srcProperty="foreignFields"
      dstProperty="primaryFields"
      dynamicProperties="primaryFields;foreignFields"/>
    <Item nodeName="RecordField" dynamicProperties="frmlabel" />
  </Items>
</File>
```

- The file managedtextform.xsl is created, to describe what is to be generated. This file should be saved to the template directory. The GASDIR environment variable should be defined in the GAS configuration and added to PATH.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text" />
  <xsl:template match="/">
    --This is generated by the XSL
  </xsl:template>
</xsl:stylesheet>
```
• A new project is created and a new A Business Application diagram (4ba) is created and saved to the project.
• A new 4rd file is added to the project and a record created. The properties for frmlabel are added to the record fields.
• The 4rd Build Rule is updated to generate a per using xslt.

```xml
<BuildRule
    additionalDependencies=""
    commands="${generate} -depth 0 -ba "$BAFilePath" &quot;$commands\" &quot;
    description="4RD Report Data compiler"
    enabled="true"
    fileType="application/generostudio-4rd"
    id="1319189142826"
    intermediateFiles=""
    outputFiles="$($(InputDir)/$(InputBaseName)per "/>
```

The same Build Rule displayed in a Genero Studio dialog:
Figure 450: Build Rule

- The 4rd is compiled and a per is generated.

The generated form

When the per file is opened and previewed, it looks like this.
Figure 451: per form

**Image directory structure**

Genero Studio searches the `GSTDIR/images/` directories for images referenced in the templates and associated files.

A copy of the image, of increasing size, should be stored in the images directory in S1 through S5 size subdirectories.

- **s1**: 16x16 pixels: structure view, Toolbar, file type icon (File Browser, Project Manager)
- **s2**: 22x22 pixels
- **s3**: 32x32 pixels
- **s4**: 48x48 pixels: Business Application diagram
- **s5**: 64x64 pixels

Depending on the DPI set on the system, other sizes can be chosen. Existing icons can be found in `GSTDIR/images/s*`. These can be used in Business Application settings as well.

**Template Reference**

Reference topics for the default template set.

**XML reference**

Reference information for the XML template files.

- [settings.agconf elements](#) on page 1122
The settings.agconf file is an XML document with four sections: Form, Business Record, Business Application, and Database. It is used to manage the properties available to each of the diagrams used by the Business Application Modeler.

The file is located in the template directory `GSTDIR/gst/bin/src/ag/tpl/dbapp<version_number>`.

The XML schema for `settings.agconf` is `GSTDIR/gst/conf/schema/agconf.xsd`.

**settings.agconf elements**

- **AGSettings** on page 1123
  - **Form** on page 1123
    - **File** on page 1124
      - **Messages** on page 1125
      - DynamicProperties
        - **DynamicProperty** on page 1126
        - Items
          - **BusinessRecord / Database** on page 1133
    - **BusinessRecord** on page 1133
      - **File** on page 1124
        - **Messages** on page 1125
        - DynamicProperties
          - **DynamicProperty** on page 1126
          - Items
            - **BusinessRecord / Database** on page 1133
    - **BusinessApplication** on page 1143
      - DynamicProperties
        - **DynamicProperty** on page 1126
      - Items
        - **Relation (BusinessApplication)** on page 1149
      - Constraints
        - **Constraint** on page 1151
    - **Database** on page 1153
      - **File** on page 1124
        - DynamicProperties
          - **DynamicProperty** on page 1126
        - Items
          - **BusinessRecord / Database** on page 1133

**Related concepts**

- Interpreting `settings.agconf` on page 1111
The settings.agconf file is an XML document with four sections: Form, Business Record, Business Application, and Database. It is used to manage the properties available to each of the diagrams used by the Business Application Modeler.

**AGSettings**
The AGSettings element is the root element of the settings.agconf file.

The AGSettings element takes an optional attribute `version`.

**Syntax**

```xml
<AGSettings version="5">
  <Form></Form>
  <BusinessRecord></BusinessRecord>
  <BusinessApplication></BusinessApplication>
  <Database></Database>
</AGSettings>
```

**Child elements**
The AGSettings element may contain the following child elements:

1. Zero or one Form element.
2. Zero or one BusinessRecord element.
3. Zero or one BusinessApplication element.
4. Zero or one Database element.

**Example**
See GSTDIR/bin/src/ag/tpl/dbapp/settings.agconf

**Form**
The Form section in the settings.agconf file manages the properties available in Form Designer.

The Form element defines the format of a form type.

**Syntax**

```xml
<Form>
  <File attributes>
    <DynamicProperties> dynamic property list </DynamicProperties>
    <Items> item list </Items>
  </File>
</Form>
```

**Child elements**
The Form element may contain the following child elements:

1. One or more File elements.

**Example**

```xml
<Form extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty>
      name="T_QUERY"
      type="BOOLEAN"
      label="canSearch"
      initialValue="1"
    </DynamicProperty>
  </DynamicProperties>
</Form>
```
File

The File element defines the file type. Add one File element for each new file type.

Syntax

```
<File>
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicPropertyGroup/>
  </DynamicProperties>
  <Items>
    <Item/>
  </Items>
</File>
```

Attributes

Table 292: File element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>extension</td>
<td>File extension.</td>
</tr>
<tr>
<td>isManaged</td>
<td>Form element only. True or False (Default is False). Set to True if the type corresponds to managed form (to generate application code using records features like queries). If managed, the records in the form are enabled by default, otherwise they cannot be used for Application Modeling.</td>
</tr>
</tbody>
</table>

Child elements

The File element may contain the following child elements:

1. One or more DynamicProperty on page 1126 and DynamicPropertyGroup on page 1132 elements, within one DynamicProperties element.
2. One or more Item elements, within one Items element.

Example - File elements representing a file type.

```
<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS" label="foreignFields" initialValue="" editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/
    <DynamicProperty name="primaryFields" type="FIELDS"
```

```
<DynamicPropertyGroup name="relations" label="Relation" description="Relation properties group" properties="foreignFields;primaryFields"/>
<DynamicProperty name="canSearch" type="BOOLEAN" label="canSearch" initialValue="true" description="Allow search using Query By Example"/>
<DynamicProperty name="canAdd" type="BOOLEAN" label="canAdd" initialValue="true" description="Allow adding items"/>
<DynamicProperty name="canModify" type="BOOLEAN" label="canModify" initialValue="true" description="Allow modifying existing items"/>
<DynamicProperty name="canDelete" type="BOOLEAN" label="canDelete" initialValue="true" description="Allow deleting items"/>
<DynamicPropertyGroup name="functionality" label="Functionality" description="Functionality properties group" properties="canAdd;canModify;canDelete;canSearch"/>
</DynamicProperties>
<Items>
  <Item nodeName="Relation" srcProperty="foreignFields" dstProperty="primaryFields" dynamicProperties="primaryFields;foreignFields"/>
  <Item nodeName="Record" dynamicProperties="canAdd;canModify;canDelete;canSearch"/>
</Items>
</File>

Messages

The Messages element is used to identify, for a specific item, error messages to disable during code generation and/or to reduce from an error to a warning.

The Messages element has attributes.

Attributes

Table 293: Messages element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>warningAsError</td>
<td>A semicolon separated list of error identifiers. If an error identifier is added to an item, the message severity is increased from warning to error for this specific error for this specific item.</td>
</tr>
<tr>
<td>ignoreWarning</td>
<td>A semicolon separated list of error identifiers. If an error identifier is added to an item, the specified error is no longer generated for this specific item. Disables the message if its severity is warning.</td>
</tr>
</tbody>
</table>

Example

```xml
<Database>
  <File extension="4db">
    <Message warningAsError="GS-11000" ignoreWarning ="GS-11360;GS-11361">
      <DynamicProperties>
```
DynamicProperty

The DynamicProperty element is a child element of DynamicProperties.

DynamicProperties can have one or more DynamicProperty children.

The DynamicProperty element has attributes. Valid attributes depend on the parent element of Form, BusinessRecord, BusinessApplication, or Database.

Syntax

```xml
<File>
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicProperty/>
    <DynamicProperty/>
  </DynamicProperties>
</File>
```

Attributes

**Table 294: DynamicProperty element attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Property identifier, name that appears in the xml. Must be unique among all the dynamic properties. The property gets associated with the Item / Relation through name.</td>
</tr>
<tr>
<td>description</td>
<td>Textual description of the property. Displayed in property view tooltip.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>type</td>
<td>Property type. Defines how the property editor behaves.</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>The editor is a checkbox. Valid values are true, false.</td>
</tr>
<tr>
<td>MULTIPLEBOOLEAN</td>
<td>The editor is the multiple checkbox dialog, editorInfo can have contains(list of checkable values), title, description and icon</td>
</tr>
<tr>
<td>ENUM</td>
<td>The editor is combobox, values are in editorInfo contains: or dynamic</td>
</tr>
<tr>
<td>INTEGER</td>
<td>The editor is spinbox, range is in editorInfo: range</td>
</tr>
<tr>
<td>TEXT</td>
<td>The editor is for text.</td>
</tr>
<tr>
<td>FILE</td>
<td>The editor is the browse dialog.</td>
</tr>
<tr>
<td>FIELDS</td>
<td>The editor is the field selection dialog. The fields are dynamic.</td>
</tr>
<tr>
<td>BRQUERY</td>
<td>The editor is the record query dialog. This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDTEXT</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDSTYLE</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDINCLUDE</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FD COLORCONDITION</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>MULTIPLELINES</td>
<td>This editor is for text with a dialog for multiple line editing.</td>
</tr>
<tr>
<td>MULTIPLEPATH</td>
<td>This editor is the multiple path dialog. The hasBrowseButton specifies if the editor should have browse button. editPathMode can be file or directory. selectionMode can be single or multiple.</td>
</tr>
</tbody>
</table>

A custom property is a property which uses a property editor defined by the customer. We provide only 2 types of custom editors:

<p>| WEB      | The editor is a input field with a browse button which opens a dialog box displaying a html document. See The WEB custom editor on page 1113. |
| PROCESS  | The editor is a input field with a browse button which launches an external process. See The PROCESS custom editor on page 1115. |</p>
<table>
<thead>
<tr>
<th><strong>Attribute</strong></th>
<th><strong>Options</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>initialValue</code></td>
<td>Default value used when no user value is set. This is the value set when the restore button is used. The default value is not saved in the file.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dynamicContent (Form)</td>
<td>Name of the dynamic content source if the property is dynamic. The available dynamic contents in Form Designer are:</td>
</tr>
<tr>
<td></td>
<td><strong>databaseName</strong> List of available databases in the project and preferences.</td>
</tr>
<tr>
<td></td>
<td><strong>sqlTabName</strong> List of available database tables in the current database.</td>
</tr>
<tr>
<td></td>
<td><strong>colName</strong> List of available columns available with the sqlTabName table in the current database.</td>
</tr>
<tr>
<td></td>
<td><strong>aggregateColName</strong> List of available columns available with the aggregateTableName table in the current database.</td>
</tr>
<tr>
<td></td>
<td><strong>displayColName</strong> List of available columns available with the displayLikeTableName table in the current database.</td>
</tr>
<tr>
<td></td>
<td><strong>validateColName</strong> List of available columns available with the validateLikeTableName table in the current database.</td>
</tr>
<tr>
<td></td>
<td><strong>fieldType</strong> List of available field type (only formonly if there is no database).</td>
</tr>
<tr>
<td></td>
<td><strong>hidden</strong> Adds USER to true, false if the selection contains only table columns.</td>
</tr>
<tr>
<td></td>
<td><strong>posX, posY, gridwith, gridHeight</strong> Returns the geometry limits depending on parent geometry.</td>
</tr>
<tr>
<td></td>
<td><strong>rowCount, colCount, stepX, stepY</strong> Corresponding min and max (for matrices).</td>
</tr>
<tr>
<td></td>
<td><strong>totalRows</strong> Min and max rows for table elements.</td>
</tr>
<tr>
<td></td>
<td><strong>expandedColumns</strong> Deprecated.</td>
</tr>
<tr>
<td></td>
<td><strong>masterTable</strong> List of record implicit tables.</td>
</tr>
<tr>
<td></td>
<td><strong>componentType</strong> List of available web components.</td>
</tr>
<tr>
<td></td>
<td><strong>lookup</strong> List of lookups in the parent record.</td>
</tr>
<tr>
<td></td>
<td><strong>srcFieldsContent</strong> List of available record fields in relation source.</td>
</tr>
<tr>
<td></td>
<td><strong>dstFieldsContent</strong> List of available record fields in relation destination.</td>
</tr>
<tr>
<td></td>
<td><strong>precision</strong> Available precision for current qual1 qualifier.</td>
</tr>
<tr>
<td></td>
<td><strong>scale</strong> Available scale for current qual2 qualifier.</td>
</tr>
<tr>
<td></td>
<td><strong>aggregatePrecision</strong> Available aggregate precision for current aggregateQual1 qualifier.</td>
</tr>
<tr>
<td></td>
<td><strong>children</strong> List of available children names.</td>
</tr>
<tr>
<td></td>
<td><strong>sibling</strong> List of available sibling names.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dynamicContent</td>
<td>Name of the dynamic content source if the property is dynamic.</td>
</tr>
<tr>
<td>(BusinessApplication)</td>
<td>The available dynamic contents in BusinessApplication are:</td>
</tr>
<tr>
<td></td>
<td>srcFieldsContent</td>
</tr>
<tr>
<td></td>
<td>List of available fields in the relation source or the current item if not a relation.</td>
</tr>
<tr>
<td></td>
<td>dstFieldsContent</td>
</tr>
<tr>
<td></td>
<td>List of available fields in the destination item or the current item if not a relation.</td>
</tr>
<tr>
<td></td>
<td>actionContent</td>
</tr>
<tr>
<td></td>
<td>List of available actions in the source item or the current item if not a relation.</td>
</tr>
<tr>
<td>type</td>
<td>List of available types for the current element.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>editorInfo</td>
<td>Semicolon separated list of \texttt{attribute:value} pair containing the property editor information. For example:</td>
</tr>
<tr>
<td>editor</td>
<td>Use another editor than the default one associated to the property type.</td>
</tr>
<tr>
<td>alwaysUpdate</td>
<td>If true, the model is modified immediately on each user action.</td>
</tr>
<tr>
<td>contains</td>
<td>List of values for ENUM type if not dynamic.</td>
</tr>
<tr>
<td>editMode</td>
<td>If true, the property editor is editable (combobox, BRQuery).</td>
</tr>
<tr>
<td>filters</td>
<td>For FILE type, set the browse dialog extension filter.</td>
</tr>
<tr>
<td>isDynamic</td>
<td>If true, the values (contains property) is computed.</td>
</tr>
<tr>
<td>range</td>
<td>Range of values for the spinbox editor (similar to contains), two values separated with '</td>
</tr>
<tr>
<td>icon</td>
<td>Contains the banner icon for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
<tr>
<td>title</td>
<td>Contains the window title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
<tr>
<td>description</td>
<td>Contains the banner title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
<tr>
<td>label</td>
<td>Text displayed in the Properties view.</td>
</tr>
<tr>
<td>readOnly</td>
<td>Sets property to read only. Options are \texttt{true} or \texttt{false}.</td>
</tr>
<tr>
<td>isHidden</td>
<td>Sets visibility of the property. Options are \texttt{true} or \texttt{false}.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
                      label="foreignFields" initialValue=""
                      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/>
  </DynamicProperties>
</File>
```
DynamicPropertyGroup

The DynamicPropertyGroup element is a child element of DynamicProperties and is used to organize a group of properties together.

The DynamicPropertyGroup element has attributes.

Syntax

```
<DynamicProperties>
    <DynamicPropertyGroup>
        <Items>
            <Item nodeNam...canDelete;canSearch/>
        </Items>
    </DynamicPropertyGroup>
</DynamicProperties>
```

Table 295: DynamicPropertyGroup element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Property identifier, name that appears in the xml.</td>
</tr>
<tr>
<td>description</td>
<td>Textual description of the property. Displayed in Property view tooltip.</td>
</tr>
<tr>
<td>label</td>
<td>Text displayed in the Properties view.</td>
</tr>
<tr>
<td>properties</td>
<td>List of dynamicProperty names to display in the group, separated by semicolons.</td>
</tr>
</tbody>
</table>
Example

```xml
<DynamicPropertyGroup name="functionality" label="Functionality"
    description="Functionality properties group"
    properties="canAdd;canModify;canDelete;canSearch;canDisplay;canEmpty"/>
```

Item (Form / BusinessRecord / Database)

The `Item` element adds new dynamic properties to some existing node(s) in the Form or Business Record or Metaschema, depending on `nodeName`.

The `Item` element has attributes.

**Table 296: Item element attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nodeName</code></td>
<td>Corresponds to the node type name, such as &quot;Relation&quot;.</td>
</tr>
<tr>
<td><code>srcProperty</code></td>
<td>Name of the property used as the source fields list, if the item is a Relation on page 1150. Defines the source field while creating the query in application generation. The value of this property is the name of the dynamic property which holds source field. This dynamic property must be associated with relation through the <code>dynamicProperty</code> attribute.</td>
</tr>
<tr>
<td><code>dstProperty</code></td>
<td>Name of the property used as the destination fields list, if the item is a Relation on page 1150. Similar to <code>srcProperty</code> it defines the destination field.</td>
</tr>
<tr>
<td><code>dynamicProperties</code></td>
<td>List of dynamic properties that apply to the item node.</td>
</tr>
</tbody>
</table>

**Syntax**

```xml
<BusinessRecord>
    <File extension="4rd">
        <DynamicProperties>
            <DynamicProperty name="foreignFields" type="FIELDS"
                label="foreignFields"
                initialValue="" dynamicContent="srcFieldsContent"
                editorInfo="isDynamic:true" />
            <DynamicProperty name="primaryFields" type="FIELDS"
                label="primaryFields"
                initialValue="" dynamicContent="dstFieldsContent"
                editorInfo="isDynamic:true" />
        </DynamicProperties>
        <Items>
            <Item
                nodeName="Relation"
                srcProperty="foreignFields"
                dstProperty="primaryFields"
                dynamicProperties="primaryFields;foreignFields" />
        </Items>
    </File>
</BusinessRecord>
```

**BusinessRecord**

Similar to `Form`, the `BusinessRecord` element defines the settings for a Business Record entity, which describes the database columns for a report. The only difference is that the `isManaged` attribute does not exist (all Business Record files are managed).

In this default format, the only item that differs from the Record associated with a form is a Relation, and the Dynamic Properties apply to that item.
Syntax

```xml
<BusinessRecord>
  <File attributes>
    <DynamicProperties> dynamic property list </DynamicProperties>
    <Items> item list </Items>
  </File>
</BusinessRecord>
```

Child elements

The BusinessRecord element may contain the following child elements:

1. One or more File elements.

Example

```xml
<BusinessRecord>
  <File extension="4rd">
    <DynamicProperties>
      <DynamicProperty name="foreignFields" type="FIELDS"
                        label="foreignFields"
                        initialValue="" dynamicContent="srcFieldsContent"
                        editorInfo="isDynamic:true" />
      <DynamicProperty name="primaryFields" type="FIELDS"
                        label="primaryFields"
                        initialValue="" dynamicContent="dstFieldsContent"
                        editorInfo="isDynamic:true" />
    </DynamicProperties>
    <Items>
      <Item nodeName="Relation" srcProperty="foreignFields"
             dstProperty="primaryFields"
             dynamicProperties="primaryFields;foreignFields" />
    </Items>
  </File>
</BusinessRecord>
```

File

The File element defines the file type. Add one File element for each new file type.

Syntax

```xml
<File>
  <DynamicProperties>
    <DynamicProperty>
    <DynamicPropertyGroup/>
  </DynamicProperties>
  <Items>
    <Item/>
  </Items>
</File>
```

Attributes

**Table 297: File element attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>extension</td>
<td>File extension.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>isManaged</td>
<td>Form element only. True or False (Default is False). Set to True if the type corresponds to managed form (to generate application code using records features like queries). If managed, the records in the form are enabled by default, otherwise they cannot be used for Application Modeling.</td>
</tr>
</tbody>
</table>

**Child elements**

The File element may contain the following child elements:

1. One or more DynamicProperty on page 1126 and DynamicPropertyGroup on page 1132 elements, within one DynamicProperties element.
2. One or more Item elements, within one Items element.

**Example - File elements representing a file type.**

```xml
<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/>
    <DynamicProperty name="primaryFields" type="FIELDS"
      label="primaryFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="dstFieldsContent"/>
    <DynamicProperty name="relations" label="Relation"
      description="Relation properties group"
      properties="foreignFields;primaryFields"/>
    <DynamicProperty name="canSearch" type="BOOLEAN"
      label="canSearch" initialValue="true"
      description="Allow search using Query By Example"/>
    <DynamicProperty name="canAdd" type="BOOLEAN"
      label="canAdd" initialValue="true"
      description="Allow adding items"/>
    <DynamicProperty name="canModify" type="BOOLEAN"
      label="canModify" initialValue="true"
      description="Allow modifying existing items"/>
    <DynamicProperty name="canDelete" type="BOOLEAN"
      label="canDelete" initialValue="true"
      description="Allow deleting items"/>
    <DynamicPropertyGroup name="functionality" label="Functionality"
      description="Functionality properties group"
      properties="canAdd;canModify;canDelete;canSearch"/>
  </DynamicProperties>
  <Items>
    <Item nodeName="Relation" srcProperty="foreignFields" dstProperty="primaryFields"
      dynamicProperties="primaryFields;foreignFields"/>
    <Item nodeName="Record" dynamicProperties="canAdd;canModify;canDelete;canSearch"/>
  </Items>
</File>
```

**Messages**

The Messages element is used to identify, for a specific item, error messages to disable during code generation and/or to reduce from an error to a warning.

The Messages element has attributes.
Attributes

Table 298: Messages element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>warningAsError</td>
<td>A semicolon separated list of error identifiers. If an error identifier is added to an item, the message severity is increased from warning to error for this specific error for this specific item.</td>
</tr>
<tr>
<td>ignoreWarning</td>
<td>A semicolon separated list of error identifiers. If an error identifier is added to an item, the specified error is no longer generated for this specific item. Disables the message if its severity is warning.</td>
</tr>
</tbody>
</table>

Example

```xml
<Database>
  <File extension="4db">
    <Message warningAsError="GS-11000" ignoreWarning="GS-11360;GS-11361">
      <DynamicProperties/>
    </Message>
  </File>
</Database>
```

DynamicProperty

The DynamicProperty element is a child element of DynamicProperties.

DynamicProperties can have one or more DynamicProperty children.

The DynamicProperty element has attributes. Valid attributes depend on the parent element of Form, BusinessRecord, BusinessApplication, or Database.

Syntax

```xml
<File>
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicProperty/>
    <DynamicProperty/>
    <DynamicProperty/>
  </DynamicProperties>
</File>
```

Attributes

Table 299: DynamicProperty element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Property identifier, name that appears in the xml. Must be unique among all the dynamic properties. The property gets associated with the Item/Relation through name.</td>
</tr>
<tr>
<td>description</td>
<td>Textual description of the property. Displayed in property view tooltip.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>type</td>
<td>Property type. Defines how the property editor behaves.</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>The editor is a checkbox. Valid values are true, false.</td>
</tr>
<tr>
<td>MULTIPLEBOOLEAN</td>
<td>The editor is the multiple checkbox dialog, editorInfo can have contains(list of checkable values), title, description and icon</td>
</tr>
<tr>
<td>ENUM</td>
<td>The editor is combobox, values are in editorInfo contains: or dynamic</td>
</tr>
<tr>
<td>INTEGER</td>
<td>The editor is spinbox, range is in editorInfo: range</td>
</tr>
<tr>
<td>TEXT</td>
<td>The editor is for text.</td>
</tr>
<tr>
<td>FILE</td>
<td>The editor is the browse dialog.</td>
</tr>
<tr>
<td>FIELDS</td>
<td>The editor is the field selection dialog. The fields are dynamic.</td>
</tr>
<tr>
<td>BRQUERY</td>
<td>The editor is the record query dialog. This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDTEXT</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDSTYLE</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDINCLUDE</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDCOLORCONDITION</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>MULTIPLELINES</td>
<td>This editor is for text with a dialog for multiple line editing.</td>
</tr>
<tr>
<td>MULTIPLEPATH</td>
<td>This editor is the multiple path dialog. The hasBrowseButton specifies if the editor should have browse button. editPathMode can be file or directory. selectionMode can be single or multiple.</td>
</tr>
</tbody>
</table>

A custom property is a property which uses a property editor defined by the customer. We provide only 2 types of custom editors:

<p>| WEB | The editor is a input field with a browse button which opens a dialog box displaying a html document. See The WEB custom editor on page 1113. |
| PROCESS | The editor is a input field with a browse button which launches an external process. See The PROCESS custom editor on page 1115. |</p>
<table>
<thead>
<tr>
<th><strong>Attribute</strong></th>
<th><strong>Options</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>initialValue</code></td>
<td>Default value used when no user value is set. This is the value set when</td>
</tr>
<tr>
<td></td>
<td>the restore button is used. The default value is not saved in the file.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dynamicContent (Form)</td>
<td>Name of the dynamic content source if the property is dynamic. The available dynamic contents in Form Designer are:</td>
</tr>
<tr>
<td></td>
<td><strong>databaseName</strong> List of available databases in the project and preferences.</td>
</tr>
<tr>
<td></td>
<td><strong>sqlTabName</strong> List of available database tables in the current database.</td>
</tr>
<tr>
<td></td>
<td><strong>colName</strong> List of available columns available with the $sqlTabName$ table in the current database</td>
</tr>
<tr>
<td></td>
<td><strong>aggregateColName</strong> List of available columns available with the $aggregateTableName$ table in the current database</td>
</tr>
<tr>
<td></td>
<td><strong>displayColName</strong> List of available columns available with the $displayLikeTableName$ table in the current database</td>
</tr>
<tr>
<td></td>
<td><strong>validateColName</strong> List of available columns available with the $validateLikeTableName$ table in the current database</td>
</tr>
<tr>
<td></td>
<td><strong>fieldType</strong> List of available field type (only formonly if there is no database).</td>
</tr>
<tr>
<td></td>
<td><strong>hidden</strong> Adds USER to true, false if the selection contains only table columns.</td>
</tr>
<tr>
<td></td>
<td><strong>posX, posY, gridwith, gridHeight</strong> Returns the geometry limits depending on parent geometry.</td>
</tr>
<tr>
<td></td>
<td><strong>rowCount, colCount, stepX, stepY</strong> Corresponding min and max (for matrices).</td>
</tr>
<tr>
<td></td>
<td><strong>totalRows</strong> Min and max rows for table elements.</td>
</tr>
<tr>
<td></td>
<td><strong>expandedColumns</strong> Deprecated.</td>
</tr>
<tr>
<td></td>
<td><strong>masterTable</strong> List of record implicit tables.</td>
</tr>
<tr>
<td></td>
<td><strong>componentType</strong> List of available web components.</td>
</tr>
<tr>
<td></td>
<td><strong>lookup</strong> List of lookups in the parent record.</td>
</tr>
<tr>
<td></td>
<td><strong>srcFieldsContent</strong> List of available record fields in relation source.</td>
</tr>
<tr>
<td></td>
<td><strong>dstFieldsContent</strong> List of available record fields in relation destination.</td>
</tr>
<tr>
<td></td>
<td><strong>precision</strong> Available precision for current $qual1$ qualifier.</td>
</tr>
<tr>
<td></td>
<td><strong>scale</strong> Available scale for current $qual2$ qualifier.</td>
</tr>
<tr>
<td></td>
<td><strong>aggregatePrecision</strong> Available aggregate precision for current $aggregateQual1$ qualifier.</td>
</tr>
<tr>
<td></td>
<td><strong>children</strong> List of available children names.</td>
</tr>
<tr>
<td></td>
<td><strong>sibling</strong> List of available sibling names.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dynamicContent (BusinessApplication)</td>
<td>Name of the dynamic content source if the property is dynamic.</td>
</tr>
<tr>
<td></td>
<td>The available dynamic contents in BusinessApplication are:</td>
</tr>
<tr>
<td>srcFieldsContent</td>
<td>List of available fields in the relation source or the current item if not a relation.</td>
</tr>
<tr>
<td>dstFieldsContent</td>
<td>List of available fields in the destination item or the current item if not a relation.</td>
</tr>
<tr>
<td>actionContent</td>
<td>List of available actions in the source item or the current item if not a relation.</td>
</tr>
<tr>
<td>type</td>
<td>List of available types for the current element.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>editorInfo</td>
<td>Semicolon separated list of attribute:value pair containing the property editor information. For example: editorInfo=&quot;contains:ButtonEdit</td>
</tr>
<tr>
<td>editor</td>
<td>Use another editor than the default one associated to the property type.</td>
</tr>
<tr>
<td>alwaysUpdate</td>
<td>If true, the model is modified immediately on each user action.</td>
</tr>
<tr>
<td>contains</td>
<td>List of values for ENUM type if not dynamic.</td>
</tr>
<tr>
<td>editMode</td>
<td>If true, the property editor is editable (combobox, BRQuery).</td>
</tr>
<tr>
<td>filters</td>
<td>For FILE type, set the browse dialog extension filter.</td>
</tr>
<tr>
<td>isDynamic</td>
<td>If true, the values (contains property) is computed.</td>
</tr>
<tr>
<td>range</td>
<td>Range of values for the spinbox editor (similar to contains), two values separated with '</td>
</tr>
<tr>
<td>icon</td>
<td>Contains the banner icon for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
<tr>
<td>title</td>
<td>Contains the window title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
<tr>
<td>description</td>
<td>Contains the banner title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
<tr>
<td>label</td>
<td>Text displayed in the Properties view.</td>
</tr>
<tr>
<td>readOnly</td>
<td>Sets property to read only. Options are true or false.</td>
</tr>
<tr>
<td>isHidden</td>
<td>Sets visibility of the property. Options are true or false.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/>
  </DynamicProperties>
</File>
```
Item (Form / BusinessRecord / Database)

The Item element adds new dynamic properties to some existing node(s) in the Form or Business Record or Meta-schema, depending on nodeName.

The Item element has attributes.

Table 300: Item element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodeName</td>
<td>Corresponds to the node type name, such as “Relation”.</td>
</tr>
<tr>
<td>srcProperty</td>
<td>Name of the property used as the source fields list, if the item is a Relation on page 1150. Defines the source field while creating the query in application generation. The value of this property is the name of the dynamic property which holds source field. This dynamic property must be associated with relation through the dynamicProperty attribute.</td>
</tr>
<tr>
<td>dstProperty</td>
<td>Name of the property used as the destination fields list, if the item is a Relation on page 1150. Similar to srcProperty it defines the destination field.</td>
</tr>
<tr>
<td>dynamicProperties</td>
<td>List of dynamic properties that apply to the item node.</td>
</tr>
</tbody>
</table>

Syntax

```xml
<BusinessRecord>
  <File extension="4rd">
    <DynamicProperties>
      <DynamicProperty name="primaryFields" type="FIELDS"
        label="primaryFields" initialValue=""
        editorInfo="isDynamic:true" dynamicContent="dstFieldsContent"/>
      <DynamicPropertyGroup name="relations" label="Relation"
        description="Relation properties group"
        properties="foreignFields;primaryFields"/>
      <DynamicProperty name="canSearch" type="BOOLEAN"
        label="canSearch" initialValue="true"
        description="Allow search using Query By Example"/>
      <DynamicProperty name="canAdd" type="BOOLEAN"
        label="canAdd" initialValue="true"
        description="Allow adding items"/>
      <DynamicProperty name="canModify" type="BOOLEAN"
        label="canModify" initialValue="true"
        description="Allow modifying existing items"/>
      <DynamicProperty name="canDelete" type="BOOLEAN"
        label="canDelete" initialValue="true"
        description="Allow deleting items"/>
      <DynamicPropertyGroup name="functionality" label="Functionality"
        description="Functionality properties group"
        properties="canAdd;canModify;canDelete;canSearch"/>
    </DynamicProperties>
  </File>
</BusinessRecord>
```
The BusinessApplication element defines the format of all the entities (items) that a BA diagram can contain, and their dynamic properties.

Unlike the Form and Business Record formats, the Item elements in this file do not modify the behavior of an existing item, they create new item definitions with new names, inheriting from a default item. The items are Program, Form, Zoom, ReportData, WebService and WebServiceServer. This format is similar to the BusinessRecord, except there is no File element.

DynamicProperty

The DynamicProperty element is a child element of DynamicProperties.

DynamicProperties can have one or more DynamicProperty children.

The DynamicProperty element has attributes. Valid attributes depend on the parent element of Form, BusinessRecord, BusinessApplication, or Database.

Syntax

```xml
<File>
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicProperty/>
    <DynamicProperty/>
  </DynamicProperties>
</File>
```

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Property identifier, name that appears in the xml. Must be unique among all the dynamic properties. The property gets associated with the Item / Relation through name.</td>
</tr>
<tr>
<td>description</td>
<td>Textual description of the property. Displayed in property view tooltip.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>Property type. Defines how the property editor behaves.</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>The editor is a checkbox. Valid values are true, false.</td>
</tr>
<tr>
<td>MULTIPLEBOOLEAN</td>
<td>The editor is the multiple checkbox dialog, editorInfo can have contains(list of checkable values), title, description and icon</td>
</tr>
<tr>
<td>ENUM</td>
<td>The editor is combobox, values are in editorInfo contains: or dynamic</td>
</tr>
<tr>
<td>INTEGER</td>
<td>The editor is spinbox, range is in editorInfo: range</td>
</tr>
<tr>
<td>TEXT</td>
<td>The editor is for text.</td>
</tr>
<tr>
<td>FILE</td>
<td>The editor is the browse dialog.</td>
</tr>
<tr>
<td>FIELDS</td>
<td>The editor is the field selection dialog. The fields are dynamic.</td>
</tr>
<tr>
<td>BRQUERY</td>
<td>The editor is the record query dialog. This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDTEXT</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDSTYLE</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDMINCLUDE</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDVALUE</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>MULTIPLELINES</td>
<td>This editor is for text with a dialog for multiple line editing.</td>
</tr>
<tr>
<td>MULTIPLEPATH</td>
<td>This editor is the multiple path dialog. The hasBrowseButton specifies if the editor should have browse button. editPathMode can be file or directory. selectionMode can be single or multiple.</td>
</tr>
</tbody>
</table>

A custom property is a property which uses a property editor defined by the customer. We provide only 2 types of custom editors:

**WEB**

The editor is a input field with a browse button which opens a dialog box displaying a html document. See The WEB custom editor on page 1113.

**PROCESS**

The editor is a input field with a browse button which launches an external process. See The PROCESS custom editor on page 1115.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialValue</td>
<td>Default value used when no user value is set. This is the value set when the restore button is used. The default value is not saved in the file.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dynamicContent (Form)</td>
<td>Name of the dynamic content source if the property is dynamic. The available dynamic contents in Form Designer are:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>databaseName</td>
<td>List of available databases in the project and preferences.</td>
</tr>
<tr>
<td>sqlTabName</td>
<td>List of available database tables in the current database.</td>
</tr>
<tr>
<td>colName</td>
<td>List of available columns available with the sqlTabName table in the current database.</td>
</tr>
<tr>
<td>aggregateColName</td>
<td>List of available columns available with the aggregateTable Name table in the current database.</td>
</tr>
<tr>
<td>displayColName</td>
<td>List of available columns available with the displayLikeTableName table in the current database.</td>
</tr>
<tr>
<td>validateColName</td>
<td>List of available columns available with the validateLikeTable Name table in the current database.</td>
</tr>
<tr>
<td>fieldType</td>
<td>List of available field type (only form only if there is no database).</td>
</tr>
<tr>
<td>hidden</td>
<td>Adds USER to true, false if the selection contains only table columns.</td>
</tr>
<tr>
<td>posX, posY, gridwith, gridHeight</td>
<td>Returns the geometry limits depending on parent geometry.</td>
</tr>
<tr>
<td>rowCount, colCount, stepX, stepY</td>
<td>Corresponding min and max (for matrices).</td>
</tr>
<tr>
<td>totalRows</td>
<td>Min and max rows for table elements.</td>
</tr>
<tr>
<td>expandedColumns</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>masterTable</td>
<td>List of record implicit tables.</td>
</tr>
<tr>
<td>componentType</td>
<td>List of available web components.</td>
</tr>
<tr>
<td>lookup</td>
<td>List of lookups in the parent record.</td>
</tr>
<tr>
<td>srcFieldsContent</td>
<td>List of available record fields in relation source.</td>
</tr>
<tr>
<td>dstFieldsContent</td>
<td>List of available record fields in relation destination.</td>
</tr>
<tr>
<td>precision</td>
<td>Available precision for current qual1 qualifier.</td>
</tr>
<tr>
<td>scale</td>
<td>Available scale for current qual2 qualifier.</td>
</tr>
<tr>
<td>aggregatePrecision</td>
<td>Available aggregate precision for current aggregateQual1 qualifier.</td>
</tr>
<tr>
<td>children</td>
<td>List of available children names.</td>
</tr>
<tr>
<td>sibling</td>
<td>List of available sibling names.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dynamicContent</td>
<td>Name of the dynamic content source if the property is dynamic.</td>
</tr>
<tr>
<td>(BusinessApplication)</td>
<td>The available dynamic contents in BusinessApplication are:</td>
</tr>
<tr>
<td>srcFieldsContent</td>
<td>List of available fields in the relation source or the current item if not a relation.</td>
</tr>
<tr>
<td>dstFieldsContent</td>
<td>List of available fields in the destination item or the current item if not a relation.</td>
</tr>
<tr>
<td>actionContent</td>
<td>List of available actions in the source item or the current item if not a relation.</td>
</tr>
<tr>
<td>type</td>
<td>List of available types for the current element.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>editorInfo</td>
<td>Semicolon separated list of attribute:value pair containing the property editor information.</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td>editorInfo=&quot;contains:ButtonEdit</td>
</tr>
<tr>
<td>editor</td>
<td>Use another editor than the default one associated to the property type.</td>
</tr>
<tr>
<td>alwaysUpdate</td>
<td>If true, the model is modified immediately on each user action.</td>
</tr>
<tr>
<td>contains</td>
<td>List of values for ENUM type if not dynamic.</td>
</tr>
<tr>
<td>editMode</td>
<td>If true, the property editor is editable (combobox, BRQuery).</td>
</tr>
<tr>
<td>filters</td>
<td>For FILE type, set the browse dialog extension filter.</td>
</tr>
<tr>
<td>isDynamic</td>
<td>If true, the values (contains property) is computed.</td>
</tr>
<tr>
<td>range</td>
<td>Range of values for the spinbox editor (similar to contains), two values separated with ''.</td>
</tr>
<tr>
<td>icon</td>
<td>Contains the banner icon for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
<tr>
<td>title</td>
<td>Contains the window title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
<tr>
<td>description</td>
<td>Contains the banner title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
<tr>
<td>label</td>
<td>Text displayed in the Properties view.</td>
</tr>
<tr>
<td>readOnly</td>
<td>Sets property to read only. Options are true or false.</td>
</tr>
<tr>
<td>isHidden</td>
<td>Sets visibility of the property. Options are true or false.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/>
  </DynamicProperties>
</File>`
<DynamicProperty name="primaryFields" type="FIELDS" label="primaryFields" initialValue="" editorInfo="isDynamic:true" dynamicContent="dstFieldsContent"/>
<DynamicPropertyGroup name="relations" label="Relation" description="Relation properties group" properties="foreignFields;primaryFields"/>
<DynamicProperty name="canSearch" type="BOOLEAN" label="canSearch" initialValue="true" description="Allow search using Query By Example"/>
<DynamicProperty name="canAdd" type="BOOLEAN" label="canAdd" initialValue="true" description="Allow adding items"/>
<DynamicProperty name="canModify" type="BOOLEAN" label="canModify" initialValue="true" description="Allow modifying existing items"/>
<DynamicProperty name="canDelete" type="BOOLEAN" label="canDelete" initialValue="true" description="Allow deleting items"/>
<DynamicPropertyGroup name="functionality" label="Functionality" description="Functionality properties group" properties="canAdd;canModify;canDelete;canSearch"/>
</DynamicProperties>
<Items>
  <Item nodeName="Relation" srcProperty="foreignFields" dstProperty="primaryFields" dynamicProperties="primaryFields;foreignFields"/>
  <Item nodeName="Record" dynamicProperties="canAdd;canModify;canDelete;canSearch"/>
</Items>
</File>

Items

The Items element defines the items on the BA diagram and the relations between them.

Item (BusinessApplication)

Unlike Form and BusinessRecord, the Item element of BusinessApplication does not modify the behavior of an existing item, it defines a new item, with a new name, inheriting from a default item.

The Item element takes attributes.

Table 302: Item element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The new item type name.</td>
</tr>
<tr>
<td>label</td>
<td>Label for item as it will appear in the BA diagram New &gt;&gt; context menu.</td>
</tr>
<tr>
<td>extension</td>
<td>File extension.</td>
</tr>
<tr>
<td>icon</td>
<td>The icon used for the item in the Business Application diagram. The file must be present in the image directory of the Application Generator template directory. See Image directory structure on page 1121.</td>
</tr>
<tr>
<td>dynamicProperties</td>
<td>List of dynamic properties dynamic properties that apply to the item node.</td>
</tr>
</tbody>
</table>
Table 303: BasicItem element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The new basic item type name.</td>
</tr>
<tr>
<td>label</td>
<td>Label for item as it will appear in the BA diagram New &gt;&gt; context menu.</td>
</tr>
<tr>
<td>icon</td>
<td>The icon used for the item in the Business Application diagram. The file must be present in the image directory of the Application Generator template directory. See Image directory structure on page 1121.</td>
</tr>
<tr>
<td>depth</td>
<td>For example, in a BA diagram specify the number of relations to traverse while generating the model. Outgoing relations are traversed. The model always contains the incoming and outgoing relation for the current item, but if the depth limit is reached, the target item definition is not generated. The depth is an integer starting from 0, or unlimited to traverse the complete application.</td>
</tr>
</tbody>
</table>

Syntax

```xml
<BusinessApplication>
  <DynamicProperties> dynamic property list </DynamicProperties>
  <Items> item list </Items>
  <BasicItem> </BasicItem>
  <Relation> </Relation>
</BusinessApplication>
```

Table 304: Relation element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Defines the type of relation. There can be multiple relations of the type defined by name in BA diagram. The name must be unique in both item and relation definition.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dynamicProperties</td>
<td>List of dynamic properties that apply to the item node.</td>
</tr>
</tbody>
</table>

**Syntax**

```xml
<BusinessApplication>
  <DynamicProperties> dynamic property list </DynamicProperties>
  <Items>
    <Item> </Item>
    <BasicItem> </BasicItem>
    <Relation> </Relation>
  </Items>
</BusinessApplication>
```

**Example**

```xml
<Item name="Program" label="Program" extension="4prg" icon="bullet_class"/>
<Item name="Form" label="CRUD Form" extension="4fdm" icon="document_4fdm" dynamicProperties=""/> <!--dynamic properties here -->
<Item name="Zoom" label="Zoom Form" extension="4fdz" icon="document_4fdz" dynamicProperties=""/> <!--dynamic properties here -->
<BasicItem name="Gallery" label="Gallery" icon="ba_choosePhoto"/>
<BasicItem name="Phone" label="Phone" icon="ba_callPhone"/>
<BasicItem name="Mail" label="Mail" icon="ba_composeMail"/>
<BasicItem name="SMS" label="SMS" icon="ba_composeSms"/>
<BasicItem name="Contact" label="Contact" icon="ba_chooseContact"/>
<BasicItem name="Maps" label="Maps" icon="ba_mapsTo"/>
<Relation name="Relation" dynamicProperties=""/> <!--dynamic properties here -->
<Relation name="PhoneRelation" dynamicProperties="action;basicItemPhoneSrcField"/>
<Relation name="MailRelation" dynamicProperties=""/> <!--dynamic properties here -->
<Relation name="SMSRelation" dynamicProperties="action;basicItemSMSToSrcField;basicItemSMSContentSrcField"/>
<Relation name="MapsRelation" dynamicProperties="action;basicItemMapsSrcField"/>
<Relation name="PhotoRelation" dynamicProperties="action;basicItemImagePathSrcField"/>
<Relation name="GalleryRelation" dynamicProperties="action;basicItemImagePathSrcField"/>
<Relation name="ContactRelation" dynamicProperties="action;basicItemVcardSrcField"/>
</Items>
```

**Constraint**

Constraints are rules which apply to the relations between items on the BA diagram. If a constraint is not respected on an item, the corresponding error is displayed in the document errors view. During code generation, broken constraints generate errors and stop the generate process with an error status.

Constraints are used in the BA diagram to regulate what relations are valid between entities. For example, with the dbapp template set, if you create a Program and a Form entity on the BA diagram, a valid relation can be set between the Program and Form.
Figure 452: Valid relation between Program and CRUD Form

However, if you try to set a relation from a Form entity to a Program, you will see the constraint icon indicating that this relation is not valid.

Each Constraint element manages a constraint on a specific type of relation: Relation, ReportRelation, or WebServiceRelation.

Table 305: Constraint element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Any string</td>
<td>Name of the constraint. Use to display the name in case of error.</td>
</tr>
<tr>
<td>reference</td>
<td>Semicolon separated list such as Relation;ReportRelation;WebServiceRelation</td>
<td>Possible relation types, or * for any.</td>
</tr>
<tr>
<td>description</td>
<td>Any string</td>
<td>Textual description of the error.</td>
</tr>
<tr>
<td>source</td>
<td>Semicolon separated list of source items from which the relation begins, such as ReportData;WebService;Program;Form.</td>
<td>Possible source item types, or * for any.</td>
</tr>
<tr>
<td>destination</td>
<td>Semicolon separated list of destination items to which the relation ends, such as Form;Zoom;FormWebService;ZoomWebService.</td>
<td>Possible destination item types, or * for any.</td>
</tr>
<tr>
<td>minSource</td>
<td>int &gt;= 0 or * for unbounded</td>
<td>The minimum number of relation sources required for this relation type. Minimum source relation cardinality.</td>
</tr>
<tr>
<td>maxSource</td>
<td>int &gt;= 0 or * for unbounded</td>
<td>The maximum number of relation sources allowed for this relation type. Maximum source relation cardinality.</td>
</tr>
<tr>
<td>minDestination</td>
<td>int &gt;= 0 or * for unbounded</td>
<td>Minimum destination relation cardinality.</td>
</tr>
</tbody>
</table>
### Attribute, Value, Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxDestination</td>
<td>int &gt;= 0</td>
<td>The maximum number of relation destinations allowed for this relation type.</td>
</tr>
<tr>
<td></td>
<td>or '*' for unbounded</td>
<td>Maximum destination relation cardinality. If set to 0, this means a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>relation that meets source and destination constraints cannot be used.</td>
</tr>
</tbody>
</table>

#### Example

```xml
<Constraint
    name="Constraint_010"
    reference="WebServiceRelation"
    description="Define at least one outgoing relationship from WebServiceServer"
    source="WebServiceServer"
    destination="WebService;FormWebService;ZoomWebService"
    minSource="1"/>
```

In this example, Constraint_10 specifies that a WebServiceServer source entity must have at least one WebServiceRelation to a destination of WebService, FormWebService or ZoomWebService.

#### Database

This Database section defines the default format for the items that have been added to the Genero Studio meta-schema file.

The widget and label properties are added to the column item in this section.

```xml
<Database>
  <File extension="4db">
    <DynamicProperties>
      <DynamicProperty name="widget" type="ENUM" label="Widget"
        initialValue="Edit" description="Associated widget"
        editorInfo="contains:ButtonEdit|CheckBox|ComboBox|DateEdit|Edit|FFImage|FFLabel|Field|ProgressBar|RadioGroup|Slider|SpinEdit|TextEdit|TimeEdit"/>
      <DynamicProperty name="label" type="TEXT" label="Label"
        initialValue="" description="Associated label when generating form" />
      <DynamicPropertyGroup name="formfieldGroup" label="Formfield"
        description="Formfield properties"
        properties="label;widget" />
    </DynamicProperties>
    <Items>
      <Item nodeName="column" dynamicProperties="widget;label"/>
    </Items>
  </File>
</Database>
```

**File**

The File element defines the file type. Add one File element for each new file type.

#### Syntax

```xml
<File>
  <DynamicProperties/>
  <DynamicProperty/>
</File>
```
Attributes

Table 306: File element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>extension</td>
<td>File extension.</td>
</tr>
<tr>
<td>isManaged</td>
<td>Form element only. True or False (Default is False). Set to True if the type corresponds to managed form (to generate application code using records features like queries). If managed, the records in the form are enabled by default, otherwise they cannot be used for Application Modeling.</td>
</tr>
</tbody>
</table>

Child elements

The File element may contain the following child elements:

1. One or more DynamicProperty on page 1126 and DynamicPropertyGroup on page 1132 elements, within one DynamicProperties element.
2. One or more Item elements, within one Items element.

Example - File elements representing a file type.

```xml
<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue="" editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/>
    <DynamicProperty name="primaryFields" type="FIELDS"
      label="primaryFields" initialValue="" editorInfo="isDynamic:true" dynamicContent="dstFieldsContent"/>
    <DynamicPropertyGroup name="relations"
      label="Relation"
      description="Relation properties group"
      properties="foreignFields;primaryFields"/>
    <DynamicProperty name="canSearch" type="BOOLEAN"
      label="canSearch" initialValue="true"
      description="Allow search using Query By Example"/>
    <DynamicProperty name="canAdd" type="BOOLEAN"
      label="canAdd" initialValue="true"
      description="Allow adding items"/>
    <DynamicProperty name="canModify" type="BOOLEAN"
      label="canModify" initialValue="true"
      description="Allow modifying existing items"/>
    <DynamicProperty name="canDelete" type="BOOLEAN"
      label="canDelete" initialValue="true"
      description="Allow deleting items"/>
    <DynamicPropertyGroup name="functionality"
      label="Functionality"
      description="Functionality properties group"
      properties="canAdd;canModify;canDelete;canSearch"/>
  </DynamicProperties>
  <Items>
    <Item nodeName="Relation" srcProperty="foreignFields">
```
Messages

The Messages element is used to identify, for a specific item, error messages to disable during code generation and/or to reduce from an error to a warning.

The Messages element has attributes.

Attributes

Table 307: Messages element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>warningAsError</td>
<td>A semicolon separated list of error identifiers. If an error identifier is added to an item, the message severity is increased from warning to error for this specific error for this specific item.</td>
</tr>
<tr>
<td>ignoreWarning</td>
<td>A semicolon separated list of error identifiers. If an error identifier is added to an item, the specified error is no longer generated for this specific item. Disables the message if its severity is warning.</td>
</tr>
</tbody>
</table>

Example

```xml
<Message warningAsError="GS-11000" ignoreWarning="GS-11360;GS-11361">
  <DynamicProperties/>
</Message>
```

DynamicProperty

The DynamicProperty element is a child element of DynamicProperties. DynamicProperties can have one or more DynamicProperty children.

The DynamicProperty element has attributes. Valid attributes depend on the parent element of Form, BusinessRecord, BusinessApplication, or Database.

Syntax

```xml
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicProperty/>
    <DynamicProperty/>
  </DynamicProperties>
```
Attributes

Table 308: DynamicProperty element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Property identifier, name that appears in the xml. Must be unique among all the dynamic properties. The property gets associated with the Item / Relation through name.</td>
</tr>
<tr>
<td>description</td>
<td>Textual description of the property. Displayed in property view tooltip.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>type</td>
<td>Property type. Defines how the property editor behaves.</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>The editor is a checkbox. Valid values are true, false.</td>
</tr>
<tr>
<td>MULTIPLEBOOLEAN</td>
<td>The editor is the multiple checkbox dialog, editorInfo can have contains(list of checkable values), title, description and icon</td>
</tr>
<tr>
<td>ENUM</td>
<td>The editor is combobox, values are in editorInfo contains: or dynamic</td>
</tr>
<tr>
<td>INTEGER</td>
<td>The editor is spinbox, range is in editorInfo: range</td>
</tr>
<tr>
<td>TEXT</td>
<td>The editor is for text.</td>
</tr>
<tr>
<td>FILE</td>
<td>The editor is the browse dialog.</td>
</tr>
<tr>
<td>FIELDS</td>
<td>The editor is the field selection dialog. The fields are dynamic.</td>
</tr>
<tr>
<td>BRQUERY</td>
<td>The editor is the record query dialog. This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDTEXT</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDSTYLE</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDINCLUDE</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>FDCOLORCONDITION</td>
<td>This property type is reserved, do not use.</td>
</tr>
<tr>
<td>MULTIPLELINES</td>
<td>This editor is for text with a dialog for multiple line editing.</td>
</tr>
<tr>
<td>MULTIPLEPATH</td>
<td>This editor is the multiple path dialog. The hasBrowseButton specifies if the editor should have browse button. editPathMode can be file or directory. selectionMode can be single or multiple.</td>
</tr>
</tbody>
</table>

A custom property is a property which uses a property editor defined by the customer. We provide only 2 types of custom editors:

<p>| WEB      | The editor is a input field with a browse button which opens a dialog box displaying a html document. See The WEB custom editor on page 1113. |
| PROCESS | The editor is a input field with a browse button which launches an external process. See The PROCESS custom editor on page 1115. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialValue</td>
<td>Default value used when no user value is set. This is the value set when the restore button is used. The default value is not saved in the file.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dynamicContent (Form)</td>
<td>Name of the dynamic content source if the property is dynamic. The available dynamic contents in Form Designer are:</td>
</tr>
<tr>
<td>databaseName</td>
<td>List of available databases in the project and preferences.</td>
</tr>
<tr>
<td>sQITabName</td>
<td>List of available database tables in the current database.</td>
</tr>
<tr>
<td>colName</td>
<td>List of available columns available with the sQITabName table in the current database.</td>
</tr>
<tr>
<td>aggregateColName</td>
<td>List of available columns available with the aggregateTableName table in the current database.</td>
</tr>
<tr>
<td>displayColName</td>
<td>List of available columns available with the displayLikeTableName table in the current database.</td>
</tr>
<tr>
<td>validateColName</td>
<td>List of available columns available with the validateLikeTableName table in the current database.</td>
</tr>
<tr>
<td>fieldType</td>
<td>List of available field type (only formonly if there is no database).</td>
</tr>
<tr>
<td>hidden</td>
<td>Adds USER to true, false if the selection contains only table columns.</td>
</tr>
<tr>
<td>posX, posY, gridwith, gridHeight</td>
<td>Returns the geometry limits depending on parent geometry.</td>
</tr>
<tr>
<td>rowCount, colCount, stepX, stepY</td>
<td>Corresponding min and max (for matrices).</td>
</tr>
<tr>
<td>totalRows</td>
<td>Min and max rows for table elements.</td>
</tr>
<tr>
<td>expandedColumns</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>masterTable</td>
<td>List of record implicit tables.</td>
</tr>
<tr>
<td>componentType</td>
<td>List of available web components.</td>
</tr>
<tr>
<td>lookup</td>
<td>List of lookups in the parent record.</td>
</tr>
<tr>
<td>srcFieldsContent</td>
<td>List of available record fields in relation source.</td>
</tr>
<tr>
<td>dstFieldsContent</td>
<td>List of available record fields in relation destination.</td>
</tr>
<tr>
<td>precision</td>
<td>Available precision for current qual1 qualifier.</td>
</tr>
<tr>
<td>scale</td>
<td>Available scale for current qual2 qualifier.</td>
</tr>
<tr>
<td>aggregatePrecision</td>
<td>Available aggregate precision for current aggregateQual1 qualifier.</td>
</tr>
<tr>
<td>children</td>
<td>List of available children names.</td>
</tr>
<tr>
<td>sibling</td>
<td>List of available sibling names.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dynamicContent</td>
<td>Name of the dynamic content source if the property is dynamic.</td>
</tr>
<tr>
<td>(BusinessApplication)</td>
<td>The available dynamic contents in BusinessApplication are:</td>
</tr>
<tr>
<td>srcFieldsContent</td>
<td>List of available fields in the relation source or the current item if not a relation.</td>
</tr>
<tr>
<td>dstFieldsContent</td>
<td>List of available fields in the destination item or the current item if not a relation.</td>
</tr>
<tr>
<td>actionContent</td>
<td>List of available actions in the source item or the current item if not a relation.</td>
</tr>
<tr>
<td>type</td>
<td>List of available types for the current element.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>editorInfo</td>
<td>Semicolon separated list of attribute:value pair containing the property editor information. For example:</td>
</tr>
<tr>
<td>editor</td>
<td>Use another editor than the default one associated to the property type.</td>
</tr>
<tr>
<td>alwaysUpdate</td>
<td>If true, the model is modified immediately on each user action.</td>
</tr>
<tr>
<td>contains</td>
<td>List of values for ENUM type if not dynamic.</td>
</tr>
<tr>
<td>editMode</td>
<td>If true, the property editor is editable (combobox, BRQuery).</td>
</tr>
<tr>
<td>filters</td>
<td>For FILE type, set the browse dialog extension filter.</td>
</tr>
<tr>
<td>isDynamic</td>
<td>If true, the values (contains property) is computed.</td>
</tr>
<tr>
<td>range</td>
<td>Range of values for the spinbox editor (similar to contains), two values separated with '</td>
</tr>
<tr>
<td>icon</td>
<td>Contains the banner icon for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
<tr>
<td>title</td>
<td>Contains the window title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
<tr>
<td>description</td>
<td>Contains the banner title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</td>
</tr>
</tbody>
</table>

**Example**

```xml
<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/>
  </DynamicProperties>
</File>```
The Item element adds new dynamic properties to some existing node(s) in the Form or Business Record or Metaschema, depending on nodeName.

The Item element has attributes.

**Table 309: Item element attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodeName</td>
<td>corresponds to the node type name, such as “Relation”.</td>
</tr>
<tr>
<td>srcProperty</td>
<td>name of the property used as the source fields list, if the item is a Relation on page 1150. Defines the source field while creating the query in application generation. The value of this property is the name of the dynamic property which holds source field. This dynamic property must be associated with relation through the dynamicProperty attribute.</td>
</tr>
<tr>
<td>dstProperty</td>
<td>name of the property used as the destination fields list, if the item is a Relation on page 1150. Similar to srcProperty it defines the destination field.</td>
</tr>
<tr>
<td>dynamicProperties</td>
<td>list of dynamic properties that apply to the item node.</td>
</tr>
</tbody>
</table>

**Syntax**

```
<BusinessRecord>
  <File extension="4rd">
    <DynamicProperties>
      ...
    </DynamicProperties>
  </File>
</BusinessRecord>
```
creatables.conf elements
A listing of all available elements in the creatables.conf file.

The file is located in the template directory GSTDIR/gst/bin/src/ag/tpl/dbapp<version_number>.
The XML schema for creatables.conf is GSTDIR/gst/conf/schema/creatables-2.xsd.

Note:
You can use GSTDIR/gst/conf/schema/creatables.xsd if you want to use the older format.

• Category on page 1164
  • New, File, Directory, Wizard on page 1165

Related concepts
Interpreting creatables.conf on page 1116
File > New menu options can be defined in the creatables.conf file. Added items are called creatables.

Creatables
The Creatables element is the root of the creatables.conf file.
The Creatables element contains two levels of Category elements.

Table 310: Creatables element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Optional version information.</td>
</tr>
<tr>
<td>xmlns:xsi</td>
<td>Standard XML attribute for schema namespace.</td>
</tr>
<tr>
<td>xsi:noNamespaceSchemaLocation</td>
<td>Standard XML attribute for schema location.</td>
</tr>
</tbody>
</table>

Syntax

```xml
<Creatables>
  <Category attributes />  
  <New attributes />       
  <File attributes />      
  <Directory attributes /> 
  <Wizard attributes />    
</Category>
</Creatables>
```
Child elements

The Creatables element may contain the following child elements:

1. One or more Category on page 1164 elements.

Example

```xml
<Creatables version="2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.4js.com/ns/gst/2.50/creatables.xsd">
    <Category index="5" label="Design (MDA)" name="MDA" icon="document_4ba">
        <New index="10" name="FDModuleForm" action="FDNew"
            label="Module Form (.4fdm)"
            icon="document_4fdm"
            description="Create an empty module in Form Designer"
            extension="4fdm"/>
    </Category>
    <Category index="40" label="Managed Code" name="ManagedCode">
        ...
    </Category>
</Creatables>
```

Category

Category elements group the actions and are used to organize actions in a way that are easy to find by a user.

The Creatables element contains two levels of Category elements. The first Category element corresponds to the Categories list in the File >> New dialog. The second Category element corresponds to the Types section in the File >> New dialog.

Table 311: Category element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td></td>
<td>Index indicates creatable order within a category.</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>Creatable name (internal id)</td>
</tr>
<tr>
<td>label</td>
<td></td>
<td>Creatable text.</td>
</tr>
<tr>
<td>icon</td>
<td></td>
<td>Icon in GSTDIR/images/</td>
</tr>
</tbody>
</table>

Syntax

```xml
<Creatables>
    <Category attributes >
        <Category attributes >
            <New attributes />
            <File attributes />
            <Directory attributes />
            <Wizard attributes />
        </Category>
    </Category>
</Creatables>
```
Child elements

The Category element may contain the following child elements:

1. One or more Category elements.
2. One or more creatable elements.

Example: Managed Form File Type

```xml
<?xml version="1.0"?>
<Creatables version="1.0">
  <Category index="5" label="Design" name="MDA" icon="document_4ba" >
    <Category index="40" label="Managed Code" name="ManagedCode">
      <Wizard index="10" 
        name="FDModuleForm" 
        action="FDNew" 
        label="Module Form from Database (4fdm)" 
        icon="document_4fdm" 
        description="Create an empty module in Form Designer" 
        extension="4fdm" />
    </Category>
  </Category>
</Creatables>
```

New, File, Directory, Wizard

A creatable is created with a New, File, Directory or Wizard element.

Table 312: Creatable element attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>Index of the creatable in a category.</td>
</tr>
<tr>
<td></td>
<td>New, Directory, Wizard</td>
</tr>
<tr>
<td>name</td>
<td>Creatable name (internal id). Required.</td>
</tr>
<tr>
<td></td>
<td>New, Directory, Wizard</td>
</tr>
<tr>
<td>action</td>
<td>See action attribute on page 1166. Required for New and Wizard creatables.</td>
</tr>
<tr>
<td>actionLabel</td>
<td>Label of the creatable when used as an action.</td>
</tr>
<tr>
<td></td>
<td>New, Directory, Wizard</td>
</tr>
<tr>
<td>icon</td>
<td>Creatable icon found. See Image directory structure on page 1121.</td>
</tr>
<tr>
<td></td>
<td>New, Directory, Wizard</td>
</tr>
<tr>
<td>extension</td>
<td>Extension of the file to be created by the creatable.</td>
</tr>
<tr>
<td></td>
<td>New, Wizard, File</td>
</tr>
<tr>
<td>copy</td>
<td>Copies file(s) before opening. If used, user is asked to choose file path and name when executing the creatable.</td>
</tr>
<tr>
<td></td>
<td>File, Directory</td>
</tr>
<tr>
<td>isTemplate</td>
<td>Specifies if file(s) should be opened using the template listed in the source attribute.</td>
</tr>
<tr>
<td></td>
<td>File, Directory</td>
</tr>
<tr>
<td>source</td>
<td>Path of source directory or source file used as the model for new file. Required.</td>
</tr>
<tr>
<td></td>
<td>File, Directory</td>
</tr>
<tr>
<td>Attribute</td>
<td>Options</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>masterFile</td>
<td>Specifies the master file (The name of a file in the directory).</td>
</tr>
<tr>
<td>open</td>
<td>List of files to open. This should specify files in the directory.</td>
</tr>
<tr>
<td>nameLabel</td>
<td>Label used in the File &gt;&gt; New dialog for the field where the user enters the name that will be given to the master file.</td>
</tr>
</tbody>
</table>

**Syntax**

```xml
<Creatables>
  <Category attributes >
      <Category attributes >
          <New attributes />
          <File attributes />
          <Directory attributes />
          <Wizard attributes />
      </Category>
  </Category>
</Creatables>
```

**Example: Managed Form File Type**

```xml
<Creatables version="1.0">
  <Category index="5" label="Design (MDA)" name="MDA" icon="document_4ba" >
      <Category index="40" label="Managed Code" name="ManagedCode">
          <Wizard index="10"
              name="FDModuleForm"
              action="FDNew"
              label="Module Form from Database (4fdm)"
              icon="document_4fdm"
              description="Create an empty module in Form Designer"
              extension="4fdm" />
      </Category>
  </Category>
</Creatables>
```

**action attribute**

The `action` attribute specifies the name of the action to activate when executing the creatable.

Action is required for `New` and `Wizard` creatables. If no action is provided for `File` and `Directory` creatables, `GSOpenFileAction` is used.

**Note:** Some file actions are unable to open some file types. If an action is used with an unsupported file type, the action will not be executed and the corresponding module error is displayed.

**Table 313: action values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Module</th>
<th>Content type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSOpenWithShell</td>
<td>Global</td>
<td>any</td>
<td>Open with external program.</td>
</tr>
<tr>
<td>GSOpenFile</td>
<td>Global</td>
<td>any</td>
<td>Open file.</td>
</tr>
<tr>
<td>BAFileOpenAction</td>
<td>Business</td>
<td>application/genero-ba</td>
<td>Open BA diagram.</td>
</tr>
<tr>
<td></td>
<td>Application</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modeling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BROpen</td>
<td>Business Record</td>
<td>application/genero-br</td>
<td>Open Business Record.</td>
</tr>
</tbody>
</table>
**$(generate) model**

The Business Application Modeler generates intermediate XML files. Examine the XML schema associated with an intermediate file to understand the content of the generated file.

The XML intermediate files generated by the Business Application Modeler - also referred to as the Application Generator (AG) model - can be opened and viewed in Code Editor. The grammar of the XML is defined by the XML Schema (.xsd).

To view the path to the grammar file, hover your mouse over the Code Editor tab containing the AG model. The resulting tooltip displays both:

- The full path of the AG model file.
- The full path to the grammar file.

The schema file used for an AG model can change frequently, depending on the version and the template. It is recommended that you use Code Editor to identify the grammar path correctly.

Locally, the schema files are located in `$GSTDIR/conf/schema/`. The files are named `agmodel*.xsd`, where * is replaced by a specific version number.

In addition to using the tooltip, you can examine the `<AG>` element to find the `xsi:noNamespaceSchemaLocation` attribute. It provides a URL to the .xrd. For example:

```xml
<xsi:noNamespaceSchemaLocation="http://www.4js.com/ns/gst/3.10/agmodel.xsd"```
Related concepts

\$\{generate\} on page 321

The \$\{generate\} command creates an intermediary XML file from modeled entities.

POINT and BLOCK reference

POINT and BLOCK sections that appear in the generated source code are defined within the template (Tcl) files. You can add additional POINT and BLOCK sections to the generated code.

Syntax

\<\texttt{POINT areaAttributes} \textgt;\textless\textgt;...	ext\textgt;\text\texttt{/\texttt{POINT}}\textgt;

\<\texttt{BLOCK areaAttributes} \textgt;\textless\textgt;...	ext\textgt;\text\texttt{/\texttt{BLOCK}}\textgt;

\texttt{areaAttributes} are standard XML attributes.

<table>
<thead>
<tr>
<th>Table 314: areaAttributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Aliases</td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td>Action</td>
</tr>
</tbody>
</table>

Name and Aliases Attribute

If the name of a POINT or BLOCK is to be changed and source code has been generated, the name can be updated using the Aliases attribute. The Name attribute will contain the new POINT / BLOCK name. The previous name of the POINT / BLOCK is preserved in the Aliases attribute.

A POINT / BLOCK may be renamed several times, thus the Aliases attribute is a space-separated list of names. It is recommended that this space-separated list is sorted from the most recent to the oldest POINT / BLOCK alias.

Packaging, deploying, and distributing apps

When you create a package, you create a bundled file that contains all of the files needed to run the app via the Genero Application Server (GAS) or on a mobile device. Once the package file is built, it can be deployed to the GAS or to a device for testing.

Packaging involves identifying the required files, organizing those files, and creating a package file that can be deployed to the application. Genero Studio eases the packaging and deployment of your app for testing by enabling you to build your package from within your project.
Packaging for the application server

Before you can deploy an application to the Genero Application Server (GAS), you need to package required files and resources in a Genero archive file.

Steps for packaging to the application server

Creating a package gathers and organizes all required files for an application.

To create a package:

1. Determine the files to include in the application.
2. Determine the directory structure for the packaged application.
3. Create a Package node in your Genero Studio project and set its properties.
4. Create directory nodes in your Package node and set their properties.
5. Build the Package.

Determine the files to include in the application

Before you can create a package, you must create an inventory of files needed to run your application, and the absolute path to these files.

Include the following files as they apply to your environment:

- Application configuration files.
- Binaries.
- fglprofile
- Database files
- Images
- Libraries
- Resource files such as topmenu, toolbar, or styles resource files
- Web component files

If you created your project using the Desktop Project (.4pw) or BAM Desktop Project options, you already have the default directory structures to organize and locate project files. In this case, your project directory contains a src directory for source modules and a bin directory for executable files. A default application configuration file (xcf) resides in the 'resources' directory.

Your project may use files that reside outside your source and target directory structures. You do not need to copy these external files to your project directory structure, because the Genero Studio packaging process copies them automatically.

Note: You can find the location of a file in your project by selecting the file in the Projects view and looking at its File path property. You can also find external files or libraries by looking at settings for key environment variables such as PATH, FGLLPATH, FGLRESOURCEPATH, and FGLIMAGESPATH.

Create an application configuration file

An application or Web service configuration file provides details required by the Genero Application Server to launch an application or Web service. If you created your project using the Desktop Project (.4pw) or BAM Desktop Project options, you already have this file. However, if this is not the case, you can create this file in Genero Studio.

About this task

When using Genero Studio, if you specify a Genero Browser Client front-end, Genero Studio generates a default application configuration file. You may need to change the default configuration. For example, you may need to pass parameters to the application or change the customization project used by the Genero Browser Client.

To specify a different configuration, create a custom application configuration file. Your custom file will eventually be merged with the default configuration file.
If you want to package multiple applications in your project, create configuration files for each one. The simplest method is to copy the existing configuration file, rename it, and update the MODULE element in the file. If you do not have an existing configuration file, see the *Genero Application Server User Guide* for information.

1. **Select File > New > Web / AS.**
2. Under the section **Configuration**, select either **Application Configuration (.xcf)** or **Web Service Configuration (.xcf)** and click **OK**.
   An untitled .xcf file displays in Code Editor, providing the basic shell needed for an application or Web service configuration file.
3. Modify the file as needed. Add those elements that you need to modify.
   See the *Genero Application Server User Guide* for specifics on updating application or Web service configuration files. You only need to include those elements you need to modify.

This example specifies the customization directory to use when displaying an application using the Genero Browser Client (GBC). The GBC element specifies the symbolic link to the customization project directory. For more information on GBC customization projects and configuration files, refer to the *Genero Browser Client User Guide*.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<APPLICATION Parent="defaultgwc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.4js.com/ns/gas/3.00/cfextwa.xsd">
    <UA_OUTPUT>
        <PUBLIC_IMAGEPATH>$(res.public.resources)</PUBLIC_IMAGEPATH>
        <GBC>mycust</GBC>
    </UA_OUTPUT>
</APPLICATION>
```

This example passes parameters at the application start.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<APPLICATION Parent="defaultgwc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.4js.com/ns/gas/2.10/cfextwa.xsd">
    <EXECUTION>
        <PARAMETERS>
            <PARAMETER>--test</PARAMETER>
            <PARAMETER>-o</PARAMETER>
            <PARAMETER>outputFile.txt</PARAMETER>
        </PARAMETERS>
    </EXECUTION>
</APPLICATION>
```

### Related concepts

- Application configuration files and Project Manager on page 411
- Genero Studio creates a default application configuration file to display applications to the Web client. You can update the default configuration with a custom application configuration file.

### Determine the directory structure for the packaged application

When you create a package, you must determine the directory structure you want for your archive.

The top-level directory is the Root directory or rootdir. You can organize different types of files in sub-directories under the rootdir. For example, you can have a bin directory for binaries and a resources directory for resource files. Alternatively, you can place all of the files into a single folder.

When you create a new Desktop Project (.4pw), all files are placed in the $(ProjectDir)\bin directory, which is the packaging Root directory. See Figure 453: Package nodes showing default settings on page 1171.
Create a Genero package node

A Genero Archive (GAR) packages your application for deployment to the Genero Application Server.

About this task

1. Select the Package node.

   **Note:** If you created your project using the Desktop Project (.4pw) or BAM Desktop Project options, it will already have a Package node and default directory nodes. If your project does not already have a Package node, right-click on a group node and select New Package.

   **Note:** The application configuration file (.xcf) should be included in the package. See Create an application configuration file on page 1169.

2. Set the Platform property to Genero Application Server.

3. Optional: If you want a different archive directory, set the Root Directory property.

4. In the Package section of the Properties pane, set the ID, Label, and Version properties.

Create directory nodes

Under the Package node, you create a directory node for each directory that contains required files.

There must be a virtual directory for any physical directory that includes required files. If you created your project using the Desktop Project (.4pw) or BAM Desktop Project options, it will already have the default directory nodes. If your project does not have directory nodes, right-click on the Package node and select New Directory. For further information about the types of files, see Determine the files to include in the application on page 1169.

Each directory has the following properties:

- **Source Directory** indicates where the files exist prior to packaging.
- **Destination Directory** indicates where the files will exist in the package structure. The Destination Directory must be within the package root directory.
- **Included files** filters the files to include in the package.
• **Excluded files** filters the files to exclude from the package.

## Build the package

If your package and packaging rules are ready, you can build the Genero archive file.

Right-click on the Package node and select **Build**.

The package is built according to the package rules. If the build is successful, a Genero archive (.gar) file is created and saved to the **Distribution directory** specified in the Package node properties. The default location is `projectdir/distbin`.

A default manifest is generated for the package. If you want to create a custom manifest, see the **Genero Application Server User Guide**; the menu option **File > New > Web / AS > Genero Archive MANIFEST** creates this template:

```xml
<?xml version="1.0" encoding="utf-8"?>
<MANIFEST xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.4js.com/ns/gas/3.00/mf.xsd">
  <DESCRIPTION>manifest description</DESCRIPTION>
  <TRIGGERS component="component_name">
    <DEPLOY>deploy_command</DEPLOY>
    <UNDEPLOY>undeploy_command</UNDEPLOY>
  </TRIGGERS>
  <RESOURCES>resources_directory</RESOURCES>
  <APPLICATION xcf="application.xcf" />
  <SERVICE xcf="service.xcf" />
</MANIFEST>
```

## Deploy the package

Use the contextual menu to deploy a package to a local or remote Genero Application Server (GAS) from the **Projects** view.

You have created a package in Genero Studio for an application server (with a .gar or .jar extension).

You can use a contextual menu to deploy packages created using Package nodes in the Genero Studio Project view.

1. Select the configuration for the Genero Application Server.
   - **Note**: If you don't have an existing configuration for your Application Server, you can create one using the Genero Configuration Management and Genero Application Server Management dialogs.

2. Right-click the Package node in the Project view and select **Deploy**.
   - The Deploy action executes the commands in the Deploy rule and you can follow the progress of the deployment in the Output view. The archive is deployed to the `$res.appdata.path)/deployment` directory defined in the GAS configuration file, `asxcf`, and a timestamp is added to the name of the deployment directory.
   - Configuration files are copied automatically to the Group directory defined in the `asxcf` file.

### Related tasks

- **Deploy a packaged file** on page 1189
  You have created a package for an application server (with a .gar or .jar extension) or a mobile device (with a .apk or .ipa extension). You need to deploy the file.

- **Deploy a mobile app for testing** on page 1178
  Follow this procedure to move your package to a connected device or emulator.

- **Define local Genero installations, GAS configurations, and environment sets** on page 164
  Local default configurations based on your installation are set up for you. Modify the configurations if the defaults are not set up for your needs.

- **Define remote Genero installations, GAS configurations, and environment sets** on page 188
Packaging for a mobile device

A mobile package includes all files needed to run the program. It specifies the directory structure containing those files when the package is installed on the device.

Android™ packages have an .apk extension, iOS packages have an .ipa extension.

What is packaging?

Packaging prepares and packages all files required to deploy an app onto a mobile device.

In other words, packaging involves identifying what files are needed for the app, organizing those files, and creating a package file that can be deployed to the app. The specifics are summarized in the following paragraphs.

Directories involved

When you create a mobile app, and when you prepare for packaging, there are three directory structures that you must understand and manage.

Source directory structure

There is a directory structure containing the app sources. These are your source files - 4gl files, form definition files, image files, and so on.

Target directory structure

There is a directory structure containing the compiled binary files. As part of the compilation process, the compiled binaries are placed in target directories. In Genero Studio, you specify this location with the Target directory property. You specify this property for group nodes, application nodes, and library nodes.

Root directory structure

There is a directory structure containing the files needed on the mobile device. The mobile device or platform will have rules on what structure is allowed. The top-level directory in this structure is known as the Root directory (rootdir). In Genero Studio, the Root directory is defined as a property of the Package node. All file that needs to be deployed to the mobile device must sit under the Root directory.

The goal of packaging

To package your app, start by having an inventory of all files needed to run the app on the mobile device. Some of these files will be in your source directory structure (such as image files), while others will sit in your target directory structure (the compiled binary files).

You must also define the directory structure you will need to create in the Root directory for your mobile device. This may sound complicated, but in reality it is quite simple. For both the Android™ and iOS devices, you can place all of the files required into a single folder. If you examine the default packaging nodes provided when you create a new BAM Mobile Project (.4pw), you see that all files end up in the directory $(ProjectDir)/bin, which is the packaging Root directory. You could build a more complex Root directory structure, however it is not necessary.

Once you have your inventory of required files and your plan for the directory structure on your device, you can build your packaging node.

In the Package node, you define the Root directory and specify which platform the package is for.

Genero installations, GAS configurations, and environment sets can be defined for the remote host.
Under the Package node, you create a Directory node for each directory you need to retrieve files from. You will have multiple Directory nodes, where each Directory node serves a purpose: locating the binaries, database files, fglprofile, images and so on.

**Important:** It is not sufficient to physically put a file in the Root directory on disk to have it included in the final package. Any file you want to include in the package must be listed as a source file in one of the directory nodes.

The Source directory indicates where the files exist prior to packaging, the Destination directory indicates where the files will be located in the package. The Destination directory must be located within the Root directory. In the Directory node, the Included files and Excluded files properties tell which files from the Source directory to include or exclude, based on filtering criteria (such as the file name or file extension).

During the packaging process, the files are copied from the source directory to the Destination directory, if these directories differ.

**Note:** If you examine the default packaging nodes provided when you create a new BAM Mobile Project (.4pw), you will see many Directory nodes where both the Source and Destination directories specify $(ProjectDir)/bin. Having the Source and Destination specify the same directory allows for packaging optimization, by not requiring files to be copied from one directory to another. Files are only copied when they are moving to a different directory. You can define separate Source and Destination directories, however the packaging process would not be optimized as a complete directory copy would be required each time the package is built.

**Have your program move files into a read-write (writable) directory**

If you create a flat file to hold all the files required by the app, a flat file is created on the mobile device. This is what is done by the default packaging nodes when you create a new BAM Mobile Project (.4pw).

For your Android™ applications, this works out-of-the-box. After the package is deployed to the device, it is unpacked into a single directory on the device. This is a read-write directory, which means that any file that needs to be writable (such as a database file) can be updated by the app.

**Warning:** If you redeploy an Android™ application, ALL files are overwritten, to include files such as a database file. You must take this into account as you plan your application upgrades, and handle any upgrade strategy in your app.

For your iOS applications, however, the directory created and holding the app files is a read-only directory. If there exists writable files that need to be updated by the app (such as a database file), those files must be placed into a read-write directory on the device. Moving files from the read-only directory to a read-write directory is not something handled by packaging. You must handle it within your app.

You are provided with two APIs that allow you to reference the underlying directories transparently:

- The `base.Application.getProgramDir()` method returns the base program directory, storing your compiled files, an initial database file, and so on. On an iOS device, this is a read-only directory.
- `os.Path.pwd()` defines a writable directory for holding writable files, such as an error log or the user database.

At deployment, when your application initially starts, we recommend that you copy the writable files from the application directory (using the function `base.Application.getProgramDir()`) to the current working directory (using the function `os.Path.pwd()`).

For an example, open the OfficeStoreMobile demo project, open the OrdersApp.4gl intermediate file, and look for the OrdersApp_install function. In this function, the `os.Path.copy` method is used to copy a database file from the read-only source directory to the read-write destination directory.

**Warning:** Take care that you do not use the same file name for a read-only resource and a read-write resource. Using the same name for both can lead to problems.
Create the user database in a writable directory

In addition to simply copying writable files, one of the first things your app needs to do is create the user database. This database may use the initial database file when creating the user database, or it may create one with a different schema for the user.

For an example, open the OfficeStoreMobile demo project, open the OrdersApp.4gl intermediate file, and look for the OrdersApp_install function. In this function, the os.Path.copy method is used to copy a database file from the read-only source directory to the read-write destination directory.

App initial deployment versus App upgrade

The example provided by the OrdersApp.4gl example needs to be evaluated in terms of initial installation and in terms of future upgrades to the application. This example solves how to initially place the file in the read-write directory. It does NOT cover an upgrade strategy. The developer needs to ensure that any new files need to be written to the read-write directory are written, that files that should not be overwritten are not overwritten, and that files that need to be updated or merged or replaced are handled. See Manage App updates on page 1187.

Related tasks
Package a mobile app on page 1175
You must package your app before you can deploy it to a mobile device.

Steps for packaging to a mobile device

Creating a package gathers and organizes all required files for a single mobile app.

If you created your project using the BAM Mobile Project (.4pw) or Mobile Project (.4pw) options, default package nodes were created for Android™ and iOS devices. If you need to create a package from scratch, the steps include:

1. Create a Package node in your project and set its properties.
2. Create Directory nodes in your Package node and set their properties.
3. Right-click on the Package node and select Build. A package is built.
4. Right-click on the Package node and select Deploy. A package is moved to the connected device or emulator.
5. Distribute your app to users. See Distribute your app on page 1184.

Related tasks
Package a mobile app on page 1175
You must package your app before you can deploy it to a mobile device.

Package a mobile app

You must package your app before you can deploy it to a mobile device.

Before you begin

Make a list of files needed to run your app and the absolute path to where these files reside.

About this task

If you created your project using the BAM Mobile Project (.4pw) or Mobile Project (.4pw) options, default package nodes were created for Android™ and iOS devices. You should still read through this procedure to add directories and validate properties.

1. Launch Genero Studio.
2. Open your project.
3. Make a list of files needed to run your app and the absolute path to where these files reside.
4. Right-click on the Packages node and select New Package.
5. Define the package properties.

Figure 454: Package properties
6. For each physical directory on disk that contains files to be included in the package, identify a corresponding logical directory under the package node. If no corresponding logical directory exists, create the directory.
   a) If a new directory is needed, right-click on the package node and select New Directory.
   b) Define the properties for each directory added.
      Files are copied from the source directory to the destination directory. In the Source directory property, specify the directory where the file(s) come from. In the Destination directory directory, specify the target directory. The destination directory must be in the root directory.
      Note: In the default project of a new BAM Mobile Project (.4pw), both properties are set to $(ProjectDir)/bin. When the source and destination directories are the same, the packaging process does not have to copy, optimizing the packaging process.
   c) To include or exclude files from the source directory, add the criteria to the Included files or Excluded files properties.

![Figure 455: Directory properties](image)

7. Confirm the Package rules. Right click on the package node and select Edit Package Rules.
   Note: You typically do not modify the package rules.
8. Right click on the package node and select Build.
   Progress of the package build appears in the Output view.
**Important:** You need an internet connection the first time you build an Android™ package. During this first build, an automated process will download and install Gradle with all necessary extensions into a directory in your user directory. Gradle is a project automation tool, find out more about Gradle at [http://www.gradle.org](http://www.gradle.org).

![Figure 456: Build package](image)

9. Confirm the package is on your local file system.
10. Validate this package by running the app on a device. Follow the instructions to [Deploy a mobile app for testing](#) on page 1178.

**What to do next**

The packaging of a GMA application is handled under the covers by the gmabuildtool, which uses a cache. This cache is created in a temporary directory and, if not managed, can grow to a significant size. You should take care to monitor the size of this cache and manually clean out the cache files, to ensure that it does not become too large in size. For more information about the gmabuildtool and the cache it creates, see the *Genero Business Development Language User Guide*.

**Set permissions for an Android™ Package**

Genero Studio selects some Android™ permissions when it creates a package node. You can assign more permissions or take permissions away.

An Android™ app must explicitly request access to resources and data outside of its process sandbox. They request this access by declaring the permissions they need for any additional capabilities. For example, if the application needs to access the contact database, you need to add the dangerous permission READ_CONTACTS. Within Genero Studio, these permissions are declared by setting the **Permissions** property for the Android™ packaging node. Genero Mobile for Android™ 1.20.15 or greater required.

1. Select the Android™ package node.
2. In the **Properties** view, within the **Android** group of properties, click the **Permissions** property.
   The ellipse icon appears.
3. Click the ellipse icon.
   The **Android permissions** dialog appears.
4. Scroll through the permissions list, selecting or de-selecting the permissions required by the application.
5. When you have finished, click **OK**.
Troubleshooting Android™ packaging issues

Here are troubleshooting tips for issues you may encounter when packaging your Android™ app.

Unexpected warning regarding a timestamp

If you are packaging with Java version 1.7.51 or later, you may receive this message:

Warning: No -tsa or -tsacert is provided and this jar is not timestamped. Without a timestamp, users may not be able to validate this jar after the signer certificate's expiration date (YYYY-MM-DD) or after any future revocation date.

This warning appears because the .apk is signed without a timestamp. A timestamp is not required for Android™. The debug certificate validity period is very long; its expiration should not be a problem.

Submission fails due to permissions assigned

Starting with Android™ API level 23, Google considers the following permissions dangerous:

- READ_CALL_LOG
- READ_CONTACTS
- WRITE_CONTACTS
- GET_ACCOUNTS
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION
- RECORD_AUDIO
- READ_PHONE_STATE
- CALL_PHONE
- READ_CALL_LOG
- WRITE_CALL_LOG
- ADD_VOICEMAIL
- USE_SIP
- PROCESS_OUTGOING_CALLS
- BODY_SENSORS
- SEND_SMS
- RECEIVE_SMS
- READ_SMS
- RECEIVE_WAP_PUSH
- RECEIVE_MMS
- READ_EXTERNAL_STORAGE
- WRITE_EXTERNAL_STORAGE

If you experience problems with your GMA submission, verify you do not have any of these permissions selected for your APK unless the application includes frontcalls or APIs requiring the permission. See Set permissions for an Android Package on page 1177.

Deploy a mobile app for testing

Follow this procedure to move your package to a connected device or emulator.

This procedure assumes you have packaged an app.

1. If deploying to a device, plug the device into a USB port.
   If necessary, update the drivers for your device.
For Android™, you can check that your computer recognizes your device by selecting **Tools > Android Tools > List Devices**. See **Configure for Genero Mobile** on page 142 for full configuration information.

2. In Genero Studio, select the correct Genero configuration.
   For example, select **Android** for Android™ devices.

3. Right-click on the package in your project and select **Deploy**.
   The Deploy action executes the commands in the Deploy rule for the platform specified for the package node. Progress of the deploy action appears in the output panel.

   You can test this with the **OfficeStoreMobile** project. Open the project and deploy the corresponding package in the Packages group.

   ![Figure 457: Deploying one of the example packages](image)

   The app appears on the device or emulator.

4. Tap the app icon to launch the app.

**Related tasks**
**Deploy the package** on page 1172
Use the contextual menu to deploy a package to a local or remote Genero Application Server (GAS) from the **Projects** view.

**Deploy a packaged file** on page 1189
You have created a package for an application server (with a .gar or .jar extension) or a mobile device (with a .apk or .ipa extension). You need to deploy the file.

Package and Directory nodes and properties
A package node defines the packaging instructions for the app. A directory nodes specifies one or more files to be included in the package.

Package and Directory Nodes

<table>
<thead>
<tr>
<th>Package nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each Package node defines the information needed to build and deploy the app. Similar to an Application node, a Package node can have dependencies set to other Libraries or Application in the project. Package nodes can be created within a Group node in a project. Build and Deploy actions are available by right-clicking on the Package node. Each Package node can have Directory nodes to further organize the package.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 315: Default Package nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node</strong></td>
</tr>
<tr>
<td>MyApp_Android</td>
</tr>
<tr>
<td>MyApp_iOS</td>
</tr>
<tr>
<td>MyApp_iOS_Simulator</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 316: Package Node properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group</strong></td>
</tr>
<tr>
<td>Project</td>
</tr>
<tr>
<td>Project</td>
</tr>
</tbody>
</table>
### Settings

<table>
<thead>
<tr>
<th>Group</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Settings</td>
<td>Platform</td>
<td>Platform used to package and deploy. See Platform on page 1192.</td>
</tr>
<tr>
<td>Settings</td>
<td>Distribution directory</td>
<td>Directory where the output files should be generated. The output files refer to the packaged deliverables -- the .apk file for Android™ devices or the .ipa file for iOS devices. The last step of the packaging process would be to put all of the Root directory files into an .apk or .ipa file and to put that final file into the Distribution directory.</td>
</tr>
<tr>
<td>Settings</td>
<td>Root directory</td>
<td>Directory from which files will be packaged. The package contents are organized using the path relative to this directory.</td>
</tr>
</tbody>
</table>

### Package

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Unique id.</td>
</tr>
<tr>
<td>Name</td>
<td>Package name. Also used in package file names.</td>
</tr>
<tr>
<td>label</td>
<td>Label displayed to user.</td>
</tr>
<tr>
<td>Version</td>
<td>Optional version number.</td>
</tr>
</tbody>
</table>

### Android™

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VersionName</td>
<td>Optional version name.</td>
</tr>
<tr>
<td>Primary color</td>
<td>Define the primary color, or the main color used in the app (Android™ 5.0 and later). For more information, see the gmabuildtool topic in the Genero Business Development Language User Guide</td>
</tr>
<tr>
<td>Primary dark color</td>
<td>Define the primary dark color, used for the status bar and the navigation bar (Android™ 5.0 and later). For more information, see the gmabuildtool topic in the Genero Business Development Language User Guide</td>
</tr>
<tr>
<td>Accent color</td>
<td>Define the accent color, used for widgets and table lines (Android™ 5.0 and later). For more information, see the gmabuildtool topic in the Genero Business Development Language User Guide</td>
</tr>
<tr>
<td>Action bar text color</td>
<td>Define the action bar text color, used as the foreground color for the texts in the action bar (Android™ 5.0 and later). For more information, see the gmabuildtool topic in the Genero Business Development Language User Guide</td>
</tr>
<tr>
<td>Icon-ldpi</td>
<td>Icon-mdpi (48x48 px) Icon-hdpi (72x72 px) Icon-xhdpi (96x96 px) Icon-xxhdpi (144x144 px) Application icon. Icons must be in PNG format. See Iconography and Devices and Displays on the Android™ Developer site for more information about icons and their use with Android™ apps.</td>
</tr>
<tr>
<td>Permissions</td>
<td>Android™ permissions requested during package install, such as request to access device camera. For further information, see Android Developer's Guide: Manifest.permission.</td>
</tr>
<tr>
<td>Cordova Plugins</td>
<td>Select which Cordova plugins to include in the package. For more information, see Programming with Cordova Plugins in the Genero Business Development Language User Guide.</td>
</tr>
<tr>
<td>Group</td>
<td>Property</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>iOS</td>
<td>Launch Screen (320x480 px)</td>
</tr>
<tr>
<td></td>
<td>Launch Screen (640x960 px)</td>
</tr>
<tr>
<td></td>
<td>Launch Screen (640x1136 px)</td>
</tr>
<tr>
<td></td>
<td>Launch Screen (750x1334 px)</td>
</tr>
<tr>
<td></td>
<td>Launch Screen (1334x750 px)</td>
</tr>
<tr>
<td></td>
<td>Launch Screen (1242x2208 px)</td>
</tr>
<tr>
<td></td>
<td>Launch Screen (2208x1242 px)</td>
</tr>
<tr>
<td></td>
<td>Launch Screen (768x1024 px)</td>
</tr>
<tr>
<td></td>
<td>Launch Screen (1024x768 px)</td>
</tr>
<tr>
<td></td>
<td>Launch Screen (1536x2048 px)</td>
</tr>
<tr>
<td></td>
<td>Launch Screen (2048x1536 px)</td>
</tr>
<tr>
<td></td>
<td>Launch Screen (2048x1536 px)</td>
</tr>
<tr>
<td>iOS</td>
<td>Launch Screen Storyboard</td>
</tr>
</tbody>
</table>

To create the file, open Xcode® and select File > New > File... > User Interface > Launch Screen. Design the launch screen using Xcode.

For more information, see Apple's documentation on Launch Screen.
Table 317: Default Directory nodes

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binaries</td>
<td>Location of binary files such as the compiled modules (42m files).</td>
</tr>
<tr>
<td>Config</td>
<td>Location of configuration files. By default, fglprofile is packaged to provide the database driver and source information.</td>
</tr>
<tr>
<td>Database</td>
<td>Location of database files. By default, db files are specified to be included as SQLite is the supported database.</td>
</tr>
<tr>
<td>Images</td>
<td>Location of image files. By default, png and jpg files are specified to be included as these are the supported image types.</td>
</tr>
<tr>
<td>libdbapp</td>
<td>Location of the Business Application Modeling template libraries.</td>
</tr>
<tr>
<td>libgst</td>
<td>Location of the libraries used with the Database Generation script.</td>
</tr>
<tr>
<td>Resources</td>
<td>Location of external files such as action defaults file (4ad) and style file (4st).</td>
</tr>
<tr>
<td>Resources_dbapp</td>
<td>Location of Business Application Modeling template resource files, dbapp.4ad and dbapp.4st.</td>
</tr>
</tbody>
</table>
Table 318: Directory Node properties

<table>
<thead>
<tr>
<th>Group</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>Description</td>
<td>Description of directory.</td>
</tr>
<tr>
<td>Settings</td>
<td>Source directory</td>
<td>The path to the directory containing the resources.</td>
</tr>
<tr>
<td>Settings</td>
<td>Destination directory</td>
<td>Directory where files are put during the packaging process. Must be within the package Root directory.</td>
</tr>
<tr>
<td>Settings</td>
<td>Included files</td>
<td>Filter of the files to include in the resources. For example, * to include all files, *42f *42m to include only files that match these types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Important:</strong> Separate filters with a space, not a comma.</td>
</tr>
<tr>
<td>Settings</td>
<td>Excluded files</td>
<td>Filter of the files to exclude from the resources.</td>
</tr>
<tr>
<td>Settings</td>
<td>Recursive</td>
<td>Check to search the files recursively in the sub directories.</td>
</tr>
</tbody>
</table>

Distribute your app

After creating your app package, you are responsible for getting the app into the hands of your users.

- Distribute an iOS app on page 1184
- Over-The-Air installation on page 1184
- Distribute an Android app on page 1186
- Distribute your app through Google Play on page 1186
- Distribute using other tools for testing on page 1187

Distribute an iOS app

Apple provides two options for distributing your app to iOS devices.

With Genero Mobile, you are able to create a signed iOS app package (with an .ipa extension).

At this point, you have two options offered by Apple® for distributing your apps.

- To distribute your app using the App Store, you must join the Apple iOS Developer Program.
- To distribute proprietary, in-house iOS apps to your employees, you must join the iOS Developer Enterprise Program.

These are two separate programs. The Apple ID that you get with your iOS Developer Enterprise Program will not let you submit your app to the App Store. You will need to join both programs (and get two Apple IDs) if you need to submit your app to the App Store and distribute proprietary, in-house iOS apps to your employees.

Related tasks

- Over-The-Air installation on page 1184
- Submit an iOS app to the AppStore on page 1185

Before you begin:

You have prepared to package your app for iOS. As you are packaging an iOS app, this can only be done from an Mac® OS development machine. The images defined in the package properties Icon (58x58 px) , Icon (60x60 px),
and **Icon (1024x1024 px)** are used during the over-the-air installation; be sure these properties were set. You must be a member of the **iOS Developer Enterprise Program**.

**Note:** If you are not a member of the iOS Developer Enterprise Program, yet want to distribute your app in-house for testing purposes only, see **Distribute using other tools for testing** on page 1187.

1. Set the **OTA_URL** environment variable to a URL. When the package is created, a **packagename.plist** file is generated. A URL is displayed in the output that can be used for OTA install.
   This step generates a HTML page with a CSS file to upload to a web server which automatically contains the URLs for downloading the IPA and installing the application.
   These files are generated in the directory specified by the **Distribution directory** property, which by default is `${ProjectDir}/distbin`.

2. Upload the **plist** file and the **ipa** package.
   **Important:** The Web server must be an HTTPS server.

3. Send the URL to the iOS device by mail or SMS. Alternatively, upload the generated HTML page and its companion CSS File to a web server.

4. On the iOS device, click on the URL.
   The app installs on the device.

**Submit an iOS app to the AppStore**

If you want your users to download their app from the Apple**®** AppStore, you must submit your app for approval.

It is assumed you have Apple Developer Program membership, which is required to request, download, and use signing certificates issued by Apple. See [http://developer.apple.com/programs/ios/](http://developer.apple.com/programs/ios/).

This procedure provides you with the steps for submitting an app to the AppStore. It includes steps for:

- Creating an App ID
- Obtaining a distribution certificate
- Creating an iOS Distribution Provisioning Profile
- Delivering an app to the AppStore

2. Go to the **iOS** platform page and register an iOS App ID by providing an app name and an App ID suffix. There are two options for setting the App ID Suffix:
   - By setting Explicit App ID
   - By setting a WildCard App ID
   **Note:** An Explicit App ID matches a single app and a WildCard App ID matches multiple apps; both methods are explained in the iOS App developer site.

3. Create an iOS Distribution Certificate:
   a) Go to the Certificates page and select the plus button +
   b) Select App Store and Ad Hoc under the Production section
   c) Follow the instructions to create your Certificate Signing Request (CSR) and upload it
   d) Download your iOS Distribution Certificate and open it in **Keychain Access** (Apple’s password-management application). In **Keychain Access** you should see your iOS Distribution Certificate with the private key.

4. Add a new iOS Distribution Provisioning Profile
   a) In the [http://developer.apple.com/](http://developer.apple.com/) web site go to the Provisioning Profiles page and select the plus button (+) to add a new profile
   b) Select the App Store
   c) Select your App ID (previously registered)
   d) Select your iOS Distribution Certificate
   e) Set a profile name
f) Generate and download your iOS Distribution Provisioning Profile

5. In Genero Mobile's configuration window
   a) Set the **IDENTITY** to your iOS Distribution Certificate's User ID
   b) Set **PROVISIONING_PROFILE** variables to the path to your Distribution Provisioning Profile (*.mobileprovision)
   c) On your project, edit the **ID** property of the iOS package you wish to submit, to match the suffix of the iOS app ID previously created
   d) Build your iOS Package

6. Create your application on iTunesConnect.
   Refer to Apple's documentation for information about using iTunes Connect
   
   **Note:** Only an agent of the Apple iOS Developer Program can connect to iTunesConnect
   a) From Genero Studio, select **Tools iOS > iOS Tools > App Store deployment > Open iTunesConnect**. (This opens the http://itunesconnect.apple.com web page.)
   b) Select My Apps and the option to add a New iOS App
   c) In the form, choose your iOS App ID from the Bundle ID field
   d) Fill in the other fields according to your app.
   e) Select Create

7. Upload the app for submission to Apple® using Xcode's Application Loader
   a) From Genero Studio, select **Tools > iOS Tools > iOS App Store deployment > Launch Application Loader**. This opens Xcode's Application Loader application. Refer to Apple's documentation for information about using Application Loader
   b) Select Deliver your App
   c) Select the Choose button and select the iOS App Store Package (.ipa) file generated on building your app with Genero Mobile. Your application will now be validated. If there are errors, they will be displayed at the end of the validation process.
   
   **Important:** Only an agent of the Apple iOS Developer Program can submit the app.

**Related concepts**

Distribute an **iOS** app on page 1184
Apple provides two options for distributing your app to iOS devices.

**Distribute an Android™ app**

Android™ has several methods for distributing your app to Android™ devices.

With Genero Mobile, you are able to create an Android™ app package (with an .apk extension).

As an open platform, Android™ offers several methods for distributing your app: via email, via a Web site, or via the Google Play Store. See Open Distribution for more details.

**Related tasks**

Distribute your app through Google Play on page 1186
To reach the broadest possible audience, distribute your Genero Mobile for Android™ (GMA) app through the Google Play marketplace.

Distribute your app through Google Play
To reach the broadest possible audience, distribute your Genero Mobile for Android™ (GMA) app through the Google Play marketplace.

Before you begin,

- You must have Android™ Software Development Kit (SDK) (Installed as part of Install and configure Java SDK and Android SDK on page 143).
- You must have a Google Developer account. To sign up, visit http://play.google.com/apps/publish/signup/. To see the agreement, visit http://play.google.com/about/developer-distribution-agreement.html.
Create a signed APK package that meets the requirements of Google Play, and load it to the Google Play Store.

1. Generate a private key using the Java keytool application. For example:

   ```
   keytool -genkey -v -keystore my-release-key.jks -keyalg RSA -keysize 2048
   -validity 10000 -alias my-alias
   ```

   You are prompted to provide the Distinguished Name (DN) fields for your key and a password for the keystore and the key. The keystore is identified by an alias, my-alias, in the example.

   **Note:** keytool is located in the bin/ directory of your JDK installation. To locate your JDK from Studio, select Tools > Genero Configurations to open the Genero Configuration Management window. From the environment set list, select Java SDK. The variable JAVA_HOME has your JDK location.

   A keystore file (.jks) is created, my-release-key.jks in the example. It is saved in the current directory (you can move it wherever you like). You now have a keystore file that contains a single key that is valid for 10,000 days.

2. Create the APK for the app.
   The APK must meet specific requirements in order to upload to the Google Play Store.
   Set the environment variables dedicated to application signing. These environment variables are defined in Genero Configurations settings, typically in the Android™ environment set:
   - KEYSTORE_PATH - Path to the key store
   - KEYSTORE_PW - Key store password
   - KEYSTORE_KEY_ALIAS - Entry name for the key in the key store
   - KEYSTORE_KEY_PW - Key password in the key store

   **Note:** We recommend you use a special keystore for development purposes in order to avoid distributing your company's private signing key to all developers. Set up a separate secured environment for signing with your company key.


Distribute using other tools for testing
After creating your app package, there are various third-party tools to distribute your app for testing purposes only.

Additional tools are available to assist you in beta testing your apps, such as TestFlight or HockeyApp (which offers Android support). Such tools provide an additional method for beta testing your app, and includes such features as Over-The-Air distribution and the gathering of feedback from testers. They are not, however, alternate solutions for the final distribution of your apps to your end users.

**Related tasks**
- Over-The-Air installation on page 1184
  Create and distribute an iOS package using an Over the Air (OTA) installation.

Manage App updates
When an new version of an app installs on a device, you might want to keep all or part of the data created with the previous version.

It is the role of the deployed program to manage the updates.

**App Update Strategy**
Detect if a previous app is installed. Each deployed app should store a file (named version.txt for this example) that contains the version of the deployed app. This file allows the app to detect if an older version of the app is installed by checking for the existence of the version.txt file. By reading the file contents, the app can retrieve the exact version of the previously deployed app.

When no previous app is detected, perform a fresh install. This might involve these tasks:
• Create version.txt.
• Create configuration files.
• Create and populate the database.
• Copy read-only files to the writable document area.

When an previous install of the app is detected and its version identified, perform an upgrade. This might evolve these tasks:

• Update version.txt to the new version number.
• Convert existing configuration files to a new file format.
• Upgrade the database schema while keeping the data.
• Delete obsolete files.
• Copy read-only files to the writable document area.

Tip: Having a clear separation between the initialization data (in the read-only directory) and the current user data (in the read-write directory) can help you manage your upgrades.

You are provided with two APIs that allow you to reference the underlying directories transparently:

• The base.Application.getProgramDir() method returns the base program directory, storing your compiled files, an initial database file, and so on. On an iOS device, this is a read-only directory.
• os.Path.pwd() defines a writable directory for holding writable files, such as an error log or the user database.

When the version of the previous install is an older version of the app, a strategy might be to run in sequence all upgrade scripts from the detected version to the latest. For example, when installing v1.5 on a device where v1.3 is the previous version, the upgrade script from v1.3 to v.4 will be executed first, then the upgrade script from v1.4 to v1.5.

This strategy involves storing modified data outside of the directory folders where the app installs, to prevent user data from being overwritten during install. This allows the programmer to read old data and do the appropriate steps to (eventually) convert them to a newer format.

Tools

The Diff Schema and Generate Database Update Script tools in Genero Studio provides database migration scripts from a previous database schema to a newer database schema.

• Diff Schema identifies the differences between two schemas
• Generate Database Update Script takes the file created using Diff Schema and generates an upgrade script. This database upgrade script can be incorporated in the app sources to manage the database upgrade.

These scripts can be called in sequence to migrate from an older version to the current.

Testing updates

App updates should be tested in a real life environment.

• Install the previous production app on a device.
• Use the app such that it stores user data.
• Deploy the new version of the app.
• Verify the upgrade works as expected.

For more information, see Technical Note TN2285: Testing iOS App Updates in the iOS Developer Library.
Deploy a packaged file

You have created a package for an application server (with a .gar or .jar extension) or a mobile device (with a .apk or .ipa extension). You need to deploy the file.

Use the contextual menu to deploy a package from the Files view in Genero Studio. The deploy action is associated with the specific file type. It does not take into consideration any custom deploy rules.

1. Right-click on the file name.
2. Select Deploy.

Related tasks
- Deploy the package on page 1172
- Use the contextual menu to deploy a package to a local or remote Genero Application Server (GAS) from the Projects view.
- Deploy a mobile app for testing on page 1178
Follow this procedure to move your package to a connected device or emulator.

Deploying a BAM application as a standalone application

To deploy a BAM application outside of a pre-configured Genero Studio environment, you must set some environment variables.

Tip: These environment variables can be set in the application configuration file.

Set the database environment.

Set FGLLDPATH to include the path to the template files. The template files themselves are part of a Genero Studio installation, and would need to be copied from a Genero Studio installation if not present.

Set FGLRESOURCEPATH to use the specific dbapp resources such as .4ad, .4st, .4tm, and .4tb. These resources can be retrieved from the styles subdirectory of the template file directory (such as %GSTDIR%\bin\src\ag\tpl\dbappX.X\styles). Setting FGLRESOURCEPATH is not needed if your generated application involves only standalone Web services.

Advanced packaging: Create your own packaging rules

Create your own packaging rules.

Package and deploy rules (desktop)

A platform includes a package rule and a deploy rule. These rules are the commands used to create and deploy a package file.

Package rules

The package rule contains a list of files that are generated by the package rule (output files) and a list of dependencies to determine if the output files are up to date. The default Package Rules can be viewed by right-clicking on the Package Node and selecting Edit Package Rules.
Figure 458: Example Package Rule

**Deploy rules**

A deploy rule runs a script to deploy the package to a configured application server.
Figure 459: Example Deploy Rule

Commands and variables

These commands and variables are specific to package and deploy rules, but other predefined node variables such as $(ProjectDir) and $(InstallDir) are also valid. See the Predefined node variables topic in the Genero Studio User Guide for a list of node variables.

$(delete)
This command, available in a package rule, deletes the specified file.

$(DistDir)
Variable for the path of the destination directory of the current package node.

$(generateXMLPackage)
This command, available in a package rule, generates an XML file that describes the contents of the package.

$(generateXMLPackage) -o outputFile.xml

$(PackageFiles)
Variable for the list of files in the package.

$(packageId)
Variable for the package id.
$\text{(PackageName)} \quad \text{Variable for the name of the current package node.}

**Package and deploy rules (mobile)**

A platform includes a package rule and a deploy rule. These rules are the commands used to create and deploy a package file.

**Platform**

A platform is similar to the concept of building and running an app based on a set of build, link and execution rules, but instead the rules are for packaging and deploying. The platform corresponds to the Platform property for the Package node in the project, defining the package and deploy rules to use.

There are currently four platforms for Genero Mobile:

- **Android™ (x86)** For Android™, use the appropriate processor depending on the setup of your Android™ Virtual Device (AVD) or you physical device hardware. To avoid performance issues, we recommend that you configure your AVD with an x86 processor and Intel® Hardware emulation enabled (HAX). When you create the default AVD using the Genero Studio menu option, the AVD has an x86 processor.

- **iOS** For an iOS physical device.

- **iOS Simulator** For an iOS simulator.

**Package rules**

The package rule contains a list of files that are generated by the package rule (output files) and a list of dependencies to determine if the output files are up to date. The Package Rules can be viewed by right-clicking on the **Package Node** and selecting **Edit Package Rules**.

![Figure 460: Example Package Rule](image)

**Deploy rules**

A deploy rule runs a script to deploy the package to a connected device or emulator.
These commands and variables are specific to package and deploy rules, but other predefined node variables such as \$\{ProjectDir\} and \$\{InstallDir\} are also valid. See the Predefined node variables topic in the Genero Studio User Guide for a list of node variables.

- \$\{delete\} This command, available in a package rule, deletes the specified file.
- \$\{DistDir\} Variable for the path of the destination directory of the current package node.
- \$\{generateXMLPackage\} This command, available in a package rule, generates an XML file that describes the contents of the package.

\$\{generateXMLPackage\} -o outputFile.xml

- \$\{GMATOOLSDIR\} Environment variable for the path to the tools that support Genero Mobile for Android™.
- \$\{GMITOOLSDIR\} Environment variable for the path to the tools that support Genero Mobile for iOS. Set
- \$\{PackageFiles\} Variable for the list of files in the package.
- \$\{packageId\} Variable for the package id.
- \$\{PackageName\} Variable for the name of the current package node.

**Upgrading GST**

These topics talk about what steps you need to take to upgrade to the next release of Genero Studio, and allows you to identify which features were added for a specific version.

- New Features of GST on page 1194
- Upgrade Guides for GST on page 1210
- Migrating to a new BAM template set on page 1215
New Features of GST

These topics provide a look back at the new features introduced with each release of Genero Studio.

GST 3.10 new features

This publication includes information about new features and changes in existing functionality.

These changes and enhancements are relevant to this publication.

Table 319: General

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual view allows you to view two documents simultaneously, side-by-side.</td>
<td>See Documents on page 121.</td>
</tr>
<tr>
<td>Create and run an automated unit test of your GUI application using Genero Ghost Client.</td>
<td>See Create and run a unit test on page 481.</td>
</tr>
<tr>
<td>Support for Hi-DPI (High Dots per Inch) screen displays.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>Use the -convert option of the gsmake command to migrate projects to newer versions.</td>
<td>See The gsmake command on page 421.</td>
</tr>
<tr>
<td>Use the gsform command to import .per files and migrate form files to newer versions.</td>
<td>See The gsform command on page 525.</td>
</tr>
<tr>
<td>User actions can be associated with Project Manager nodes; remote user actions can execute on the server side; $(open) command now available; and when executing a user action, the previous execution parameters are remembered.</td>
<td>See User actions on page 132.</td>
</tr>
<tr>
<td>The Genero Desktop Client and Genero Application Server can now be stopped, as they appear as tasks in the Tasks view.</td>
<td>See Tasks view on page 120.</td>
</tr>
<tr>
<td>Projects remember the last configuration used, and prompt you to use that configuration when you re-open a project.</td>
<td>See Genero project file (4pw) on page 400.</td>
</tr>
<tr>
<td>The Welcome Page is now based on Chromium.</td>
<td>See The Welcome Page on page 84.</td>
</tr>
<tr>
<td>Genero Studio now supports TUI (text mode) applications.</td>
<td>See Configure for a TUI client on page 179.</td>
</tr>
<tr>
<td>Genero Studio now manages opening files that have previously been saved using a newer version of Genero Studio.</td>
<td>See Open from a different version on page 101.</td>
</tr>
<tr>
<td>Restore the layout of the default workspaces.</td>
<td>See Workspaces Configuration on page 131.</td>
</tr>
<tr>
<td>When using a remote configuration, the Genero Workplace Window opens on the remote host.</td>
<td>See Tools menu on page 113.</td>
</tr>
<tr>
<td>Genero Archive files can be deployed from the Files view (File Browser).</td>
<td>See Deploy a packaged file on page 1189.</td>
</tr>
</tbody>
</table>

**Open as Text... and Open with Encoding...** menu options can be found under the File menu, and within the contextual menus in Project Manager and File Browser.

Genero Studio Help includes a button that opens a feedback page for documentation comments. The URL of the current topic is passed to the feedback page.

No additional reference.
### Overview

Genero Studio prompts when a software or documentation update is available. You have the option of completing the install or turning off update notifications.

<table>
<thead>
<tr>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>See <a href="#">Updating Genero Studio and its documentation</a> on page 124.</td>
</tr>
</tbody>
</table>

### Table 320: Project Manager

<table>
<thead>
<tr>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>A new project now contains a <strong>Test</strong> application node, for the generated unit test scenario.</td>
</tr>
<tr>
<td>Options added to the <code>$ (4fdcomp)</code> build command.</td>
</tr>
<tr>
<td>Filter results to better find your project items.</td>
</tr>
<tr>
<td>You can now select an application as a dependency.</td>
</tr>
<tr>
<td>Preview of text <code>.per</code> forms supported from both Project Manager and File Browser.</td>
</tr>
<tr>
<td>Structured projects are designed to manage the mapping and deployment of custom <code>.ttf</code> fonts.</td>
</tr>
<tr>
<td>Dependencies can be created between application nodes. This is required for generating unit tests for the Genero Ghost Client.</td>
</tr>
<tr>
<td>At the top of the opened document, the color of the line indicates the existence of errors or warnings. If the document contains an error, the line is red. If the document contains a warning, the line is orange. If there are no errors or warnings, the line is blue.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>See <a href="#">Command line options for build, link, execution rules</a> on page 416.</td>
</tr>
<tr>
<td>See <a href="#">Projects view</a> on page 119.</td>
</tr>
<tr>
<td>See <a href="#">Setting external dependencies</a> on page 411.</td>
</tr>
<tr>
<td>See <a href="#">Build menu</a> on page 108.</td>
</tr>
<tr>
<td>See <a href="#">Structured projects</a> on page 400.</td>
</tr>
<tr>
<td>See <a href="#">GUI Testing</a> on page 480.</td>
</tr>
</tbody>
</table>

### Table 321: Business Application Modeler

<table>
<thead>
<tr>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM now supports Genero Report Viewer for HTML5.</td>
</tr>
<tr>
<td>A Custom Form entity provides a form in which the code remains free.</td>
</tr>
<tr>
<td>For BAM template developers! New RecordField attributes have been added for use with templates: qualifier1, qualifier2, precision, scale, and length.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>See <a href="#">CRUD Form entity</a> on page 237.</td>
</tr>
<tr>
<td>See <a href="#">Custom Form entity</a> on page 243.</td>
</tr>
<tr>
<td>No additional reference.</td>
</tr>
</tbody>
</table>

### Table 322: Meta-Schema Manager

<table>
<thead>
<tr>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <strong>Category</strong> property provides a new method for grouping and filtering tables; the <strong>Description</strong> property allows developers to document schema tables and columns.</td>
</tr>
<tr>
<td>Copy a table or column from the meta-schema diagram and paste the variable definition into a Code Editor document.</td>
</tr>
<tr>
<td>Support added for index and foreign key extraction for PostgreSQL.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>See <a href="#">Meta-schema properties</a>.</td>
</tr>
<tr>
<td>See <a href="#">Copy a table or column definition as text</a> on page 361.</td>
</tr>
<tr>
<td>See <a href="#">Extract meta-schema information from database</a> on page 277.</td>
</tr>
</tbody>
</table>
## Overview

A database schema supports views.

---

### Table 323: Code Editor

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Code Editor now supports the Swift language, as well as the Web</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>customization languages CSS, SCSS, Sass, JavaScript and JSON.</td>
<td></td>
</tr>
<tr>
<td>The Code Editor supports right-to-left languages, such as Arabic. Set the</td>
<td></td>
</tr>
<tr>
<td>Text Encoding preference to ISO-8859-6.</td>
<td></td>
</tr>
<tr>
<td>You can now find a function or variable declaration using CRTL+CLICK.</td>
<td></td>
</tr>
<tr>
<td>The Structure view synchronizes with cursor movements to show the current</td>
<td></td>
</tr>
<tr>
<td>function.</td>
<td></td>
</tr>
<tr>
<td>In the Code Structure view, private functions and variables are displayed</td>
<td></td>
</tr>
<tr>
<td>with a lock icon.</td>
<td></td>
</tr>
<tr>
<td>You can find the callers of a function using the right-click menu.</td>
<td></td>
</tr>
<tr>
<td>You can turn off warning messages.</td>
<td></td>
</tr>
<tr>
<td>Full file names now appear in the document tabs.</td>
<td></td>
</tr>
<tr>
<td>The selection range now appears in the status bar.</td>
<td></td>
</tr>
<tr>
<td>You have the option to copy the entire line if nothing is already selected.</td>
<td></td>
</tr>
<tr>
<td>Code Editor can auto-detect UTF-16 and UTF-32 files, even when the byte</td>
<td></td>
</tr>
<tr>
<td>order mark (BOM) is missing from the file.</td>
<td></td>
</tr>
<tr>
<td>You can select the file encoding when opening a file.</td>
<td></td>
</tr>
<tr>
<td>The file encoding displays in the Status bar.</td>
<td></td>
</tr>
<tr>
<td>Bookmarks are now stored by project. Project-specific bookmarks are</td>
<td></td>
</tr>
<tr>
<td>restored when a project opens.</td>
<td></td>
</tr>
<tr>
<td>The Diff tool has been enhanced to quickly include or exclude case</td>
<td></td>
</tr>
<tr>
<td>sensitivity and white space differences; the number of differences</td>
<td></td>
</tr>
<tr>
<td>displays; and you can use the navigation toolbar to move between</td>
<td></td>
</tr>
<tr>
<td>differences.</td>
<td></td>
</tr>
<tr>
<td>Quick run allows you to run the current Genero .4gl file.</td>
<td></td>
</tr>
<tr>
<td>The Code Coverage view provides statistics about source code line usage.</td>
<td></td>
</tr>
<tr>
<td>Highlight identifiers and define the colors to use.</td>
<td></td>
</tr>
<tr>
<td>Set a preference where you can double-click on a word to highlight all</td>
<td></td>
</tr>
<tr>
<td>occurrences of the word in the document.</td>
<td></td>
</tr>
</tbody>
</table>
### Table 324: Form Designer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic generation of <code>.wcsettings</code> file for WebComponents.</td>
<td>See Form Designer preferences on page 528.</td>
</tr>
<tr>
<td>Form item property <code>initialPageSize</code> for <code>ScrollGrid</code> container.</td>
<td>See <code>initialPageSize</code> on page 558.</td>
</tr>
<tr>
<td>Form item property <code>keys</code> for formfields.</td>
<td>See <code>keys</code> on page 560.</td>
</tr>
<tr>
<td>Form item property <code>placeholder</code> defines a hint for the user when the field contains no value.</td>
<td>See <code>placeholder</code> on page 563.</td>
</tr>
</tbody>
</table>

### Table 325: Genero Report Writer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can set rendering hints for optimizing the printing of reports.</td>
<td>See Optimize the rendering process on page 715.</td>
</tr>
<tr>
<td>Driverless, server side, silent printing is available using PJL.</td>
<td>See Printing without a driver using PJL on page 704.</td>
</tr>
<tr>
<td>New functions <code>oddPhysicalPage()</code> and <code>evenPhysicalPage()</code> reveal whether an object starts on an odd or an even page.</td>
<td>See Start on an odd or even page on page 805.</td>
</tr>
<tr>
<td>An SAP® Connector for the Genero Report Writer can be provided on request. Please contact your support center.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>A Pentaho PDI plug-in can be provided on request. Please contact your support center.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>The scripts residing in <code>GREDIR/bin</code> now use the default Java version of the operating system.</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>

### Table 326: Genero Report Designer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original data produced by an application can now be transformed within the report designer without coding. Transformations include selecting, duplicating, moving, re-ordering, pivotizing and computing aggregations.</td>
<td>See Report schema transformations on page 775.</td>
</tr>
<tr>
<td>The Image Box can now embed the first page of a PDF file.</td>
<td>See Image Box on page 865.</td>
</tr>
<tr>
<td>The new Toolbox object, PDFBOX, can embed an entire PDF.</td>
<td>See PDF Box on page 864.</td>
</tr>
<tr>
<td>You can create a Spider Web chart.</td>
<td>See Category charts and <code>drawAs</code> property.</td>
</tr>
<tr>
<td>The business graphs now use a new &quot;flat&quot; design by default, although you can still specify the &quot;classic&quot; style.</td>
<td>See Specifying classic or default style on page 832.</td>
</tr>
<tr>
<td>If the parent object is a propagating container, the child object does not fit in the remaining space for the parent object, and you set the <code>Y-Size</code> property to <code>rest</code>, the child now expands to the maximum extent of the parent rather than just the remainder of the parent.</td>
<td>See Y-size property.</td>
</tr>
<tr>
<td>For <code>X-Size Adjustment</code> and <code>Y-Size Adjustment</code>, a value of <code>expandToParent</code> now causes the box to stretch as much as possible without intersecting the borders of a parent or sibling.</td>
<td>See X-Size Adjustment and Y-Size Adjustment properties.</td>
</tr>
</tbody>
</table>
For business graphs and pivot tables, the new `rangeUpperBound` and `rangeLowerBound` properties define the highest and lowest values on the Y-Axis.

For XY charts, the new `domainUpperBound` and `domainLowerBound` properties define the highest and lowest values on the X-Axis.

Report elements now include a `comment` property.

---

**Table 327: Graphical Debugger**

**Overview**

Look-and-feel improvements to the Data view include new icons for the scope folders, new icons for array, record and simple variables, and reorganized actions; the "Show modified variables" filter has been added; when an array contains more data than displayed, you can fetch and display more data; and the Data view displays variables from all modules of the debugged application.

The Breakpoints view is always available, allowing you to manage breakpoints before starting your debug session; and you can sort the breakpoints in the Breakpoints view by clicking on a column header.

Various debug views adapt to the state of the debugger, disabling additions of breakpoints and entry of commands while the debugger is active.

The Execute with Trace option can be selected to run applications, available from the context menu of an application node or from the Debug menu.

**Reference**

See `rangeUpperBound` and `rangeLowerBound` properties.

See `domainUpperBound` and `domainLowerBound` properties.

See `comment` property.

See Data view on page 600.

See Breakpoints view on page 602.

See Command view on page 599 and Breakpoints on page 592.

See Debug menu on page 109.

---

**Table 328: Genero Mobile**

**Overview**

Cordova plug-in support added for Android™ packaging.

Deploy applications using the Google Play Developer Console.

The Android™ SDK Manager and Android™ Virtual Device Manager are no longer provided as part of the Android™ SDK package, and the corresponding menu items have been removed.

Mobile packages can be deployed from the Files view (File Browser).

**Reference**

See Package and Directory nodes and properties on page 1180.

See Tools menu on page 113.

See Tools menu on page 113.

See Deploy a packaged file on page 1189.

---

**GST 3.00 new features**

This publication includes information about new features and changes in existing functionality.

**Important:** Please read GST 2.51 new features on page 1202, for a list of features that were introduced with the Genero Mobile 1.0 release.

These changes and enhancements are relevant to this publication.
### Table 329: General

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genero Studio now supports Genero, Genero Report Writer for BDL, and</td>
<td>N/A</td>
</tr>
<tr>
<td>Genero Mobile from a single installation.</td>
<td>See Configure for the Web client on page 171.</td>
</tr>
<tr>
<td>Genero Studio now supports the Genero Browser Client (GBC).</td>
<td></td>
</tr>
<tr>
<td>Genero Studio now supports connecting to the application server using</td>
<td></td>
</tr>
<tr>
<td>HTTPS.</td>
<td></td>
</tr>
<tr>
<td>Generate a Genero archive (GAR) from Studio, for deployment to the GAS.</td>
<td></td>
</tr>
<tr>
<td>Genero Studio gives the ability to locate a document in the File Browser,</td>
<td></td>
</tr>
<tr>
<td>in a BA diagram, or in the System File Browser (the file browser of the</td>
<td></td>
</tr>
<tr>
<td>operating system). The new System File Browser feature facilitates the</td>
<td></td>
</tr>
<tr>
<td>use of system file explorer integrated tools, such as SVN or Git</td>
<td></td>
</tr>
<tr>
<td>integration.</td>
<td></td>
</tr>
<tr>
<td>New configuration option for GAS, allowing the ability to run and debug</td>
<td></td>
</tr>
<tr>
<td>Web services from Genero Studio.</td>
<td></td>
</tr>
</tbody>
</table>

### Table 330: Project Manager

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>The properties Web Service and Web Service URL suffix have been added</td>
<td>See Project Manager node properties on page 434.</td>
</tr>
<tr>
<td>for the Application node, allowing the ability to run and debug Web</td>
<td></td>
</tr>
<tr>
<td>services from Genero Studio.</td>
<td></td>
</tr>
</tbody>
</table>

### Table 331: DB Explorer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Explorer module introduced to view and modify data in database tables</td>
<td>See DB Explorer on page 392.</td>
</tr>
<tr>
<td>and to test SQL query results. With this tool, you can right-click on</td>
<td></td>
</tr>
<tr>
<td>forms, reports and Web services to view the data.</td>
<td></td>
</tr>
<tr>
<td>DB Explorer expands support of SQL commands, in addition to SELECT,</td>
<td>See DB Explorer on page 392 and Write a SQL query by hand on page 398.</td>
</tr>
<tr>
<td>INSERT, UPDATE, and DELETE.</td>
<td></td>
</tr>
</tbody>
</table>

### Table 332: Meta-Schema Manager

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhanced schema view displays database schema modifications at a glance.</td>
<td>See Viewing a meta-schema on page 356.</td>
</tr>
<tr>
<td>Mouse over items in the schema for more detail, to include a summary of</td>
<td>See Viewing a meta-schema on page 356.</td>
</tr>
<tr>
<td>schema changes, primary key column details, and more.</td>
<td></td>
</tr>
<tr>
<td>Move columns to another table using drag-and-drop.</td>
<td>See Viewing a meta-schema on page 356.</td>
</tr>
<tr>
<td>HTML meta-schema documentation provides details of all database objects</td>
<td>See Generate meta-schema documentation on page 361.</td>
</tr>
<tr>
<td>and facilitates global schema review.</td>
<td></td>
</tr>
</tbody>
</table>
### Table 333: Genero Mobile

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toggle label display shows or hides foreign key names in the diagram.</td>
<td>See Viewing a meta-schema on page 356.</td>
</tr>
</tbody>
</table>

### Table 334: Business Application Modeler

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can debug an application deployed to a mobile device. With this new</td>
<td>See Attach the Graphical Debugger to a mobile process on page 617</td>
</tr>
<tr>
<td>feature, the application is running on the mobile device and the Graphical</td>
<td></td>
</tr>
<tr>
<td>Debugger is able to attach to the process.</td>
<td></td>
</tr>
<tr>
<td>The <code>DBAPP_MOBILE</code> environment variable provides warning messages</td>
<td>See <code>DBAPP_MOBILE</code> on page 320.</td>
</tr>
<tr>
<td>regarding features not supported by mobile devices during the compilation</td>
<td></td>
</tr>
<tr>
<td>of applications generated by the Business Application Modeler.</td>
<td></td>
</tr>
</tbody>
</table>

### Table 335: Code Editor

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Editor supports a horizontal view in the Diff tool, in addition to</td>
<td>See Using the Diff tool on page 451.</td>
</tr>
<tr>
<td>the vertical view of previous versions.</td>
<td></td>
</tr>
</tbody>
</table>

### Table 336: Form Designer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for new Form properties: `keyboardHint, completer, wantFixedPage</td>
<td></td>
</tr>
<tr>
<td>Size, action, Disclosure Indicator</td>
<td>See Properties and Form Designer on page 545.</td>
</tr>
<tr>
<td>Support for new <code>DateTimeEdit</code> widget.</td>
<td>See DateTimeEdit on page 508.</td>
</tr>
</tbody>
</table>

### Table 337: Search

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Results pane displays an improved search view, to include previous</td>
<td>See Search Results view on page 471.</td>
</tr>
<tr>
<td>and current search results organized as a collapsible tree.</td>
<td></td>
</tr>
</tbody>
</table>

### Table 338: Genero Report Writer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genero Report Viewer for HTML5 provides a browser equivalent of the</td>
<td>See <code>fgl_report_selectDevice</code> on page 743.</td>
</tr>
<tr>
<td>Genero Report Viewer.</td>
<td></td>
</tr>
<tr>
<td>A command-line utility checks and upgrades report design documents</td>
<td>See Update report design documents to the current version on</td>
</tr>
<tr>
<td>(.4rp) files in batch.</td>
<td>page 983.</td>
</tr>
</tbody>
</table>
Overview

Report templates provide a wizard-based method for creating report design documents (.4rp) from a generic report design. The wizard allows you to bind repeating sections, add fields, and bind placeholders and parameters from a data schema, in order to create a stand-alone report design document. A library of report templates have been provided, and you can create your own templates. A template expansion mechanism is available as a command line tool, usable from applications for generic reports.

The report engine now limits the number worker threads in distributed mode, to prevent memory exhaustion in times of critical load. Change the default value (25 threads) with the environment variable GRE_MAX_CONCURRENT_JOBS.

You can now configure the default output directory for the Genero Report Engine with the GREPORTOUTPUTDIR environment variable.

There is an improved architecture using HTTP for previewing documents in a distributed setup. Besides improvements in performance, the solution no longer requires the installation of a DVM on the remote machine.

Table 339: Genero BDL Reporting APIs

Overview

APIs support the Genero Report Viewer for HTML5:

A new API supports distributed mode.

APIs have been introduced to get error details.

A new API can programmatically set the value of environment or user-defined variables.

Table 340: Genero Report Designer

Overview

Genero Report Designer provides a LastPageFooter section property.

Support of Intelligent Mail bar code type.

New smartParse bar code property for bar code Code-128. When enabled, this allows you to enter the bar code value, and the internal code will be computed for you resulting in the shortest visual representation.

New gs1* bar code aliases.
**Table 341: Graphical Debugger**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can debug an already running process by attaching to the process. The process can be running locally or on a remote computer. Attaching to a remote process allows you to debug an application at a production site where Genero Studio is not installed.</td>
<td>See Start the Debugger on a running program on page 590.</td>
</tr>
<tr>
<td>You can debug an application deployed to a mobile device. With this new feature, the application is running on the mobile device and the Graphical Debugger is able to attach to the process.</td>
<td>See Attach the Graphical Debugger to a mobile process on page 617</td>
</tr>
<tr>
<td>You can debug Web services: server, client or both.</td>
<td>See Debug a Web services server application on page 614.</td>
</tr>
</tbody>
</table>

**GST 2.51 new features**

This publication includes information about new features and changes in existing functionality.

These changes and enhancements are relevant to this publication.

**Table 342: General**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genero Studio includes features to support mobile app development for Android™ and iOS.</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>

**GST 2.50 new features**

This publication includes information about new features and changes in existing functionality.

These changes and enhancements are relevant to this publication. See also Migration notes for Version 2.50.

**Table 343: General**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac® OS support. All build processes are done remotely for remote host configurations, optimizing speed and efficiency of remote builds on slower connections. Configurations are stored on the remote host allowing users to reconnect from another machine with configurations intact. Improved file compatibility. If you open a file that was saved with a different version of Genero Studio, you can decide how to handle the file. When working in environments with multiple versions of Genero Studio installed, files will open in the appropriate version. Improved Diff tool. Modifications made to a file are shown as annotations, indicating added, modified and removed lines. You can now switch from a single pane view to a dual pane view showing the original file (as it is in the SVN repository for example) and the locally edited file. --diff file1 file2 support. Genero Studio opens the given files in diff mode. This option is also integrated in File Browser. GAS Standalone (httpdispatch) can be automatically started by setting the startup script in the GAS configuration.</td>
<td>No further reference. See Setting up a remote environment on page 179. See Migration notes for GST 2.50 upgrade guide on page 1211. See Open from a different version on page 101. See Using the Diff tool on page 451. See Command line options on page 125. See Configure for the Web client on page 171.</td>
</tr>
</tbody>
</table>
Overview | Reference
---|---
The Environment Variable definition dialog now includes a value list type. | See Add or edit environment variables on page 168
User documentation has been migrated to XML-based DITA (Darwin Information Typing Architecture). | No further reference.
Distributed mode. Allows the report engine to be started as a daemon to which Genero applications can connect to process the reports, allowing for vastly faster processing for short documents and improved scalability. | See Distributed Mode on page 1015.

Table 344: Project Manager

Overview | Reference
---|---
Project Manager extended to support other programming languages. | See Building and linking programs on page 412.
A Deploy Project 4pw template (File > New > Genero Files > Deploy Project) creates a directory structure recommended for a project which is to be deployed on the Genero Application Server (GAS). | No further reference.
Build Rules dialog and File Associations dialogs are accessible from the Tools menu (Tools > Global setup or Tools > Specific setup) and no longer in Preferences. The global setup overrides the built-in factory setup. It applies to all configuration of a user for the current machine or remote machine. The specific setup overrides the global setup and is specified by setting the GSTSETUPDIR variable in the Genero configuration. | See Migration notes for GST 2.50 upgrade guide on page 1211
User build rules, template build rules, user and template file associations are also stored on the server and not on the local machine allowing users to reconnect from another machine with build rules intact. | 
$(tcl)$ has been deprecated and replaced with tclsh in Business Application Modeling build rules. The tclsh executable generates the final file by using both a Tcl template file and the intermediary XML file created by the $(generate) command. | See tclsh on page 322.
Standard/error output redirection supported in build rules and user actions. | See Add/Edit a build rule on page 414.

Table 345: Form Designer

Overview | Reference
---|---
Graphical Topmenu editor allows for Topmenu editing directly in the form design. | See Add a Topmenu to a form on page 519.
Improved ergonomics in the Form from Database wizard. Easier to create complex queries and select appropriate fields. Shows foreign key relations and NOT NULL constraints. | See Create a database-aware form on page 490.
Overview

New and improved form item properties:

**query**

A single *query* property is now used to enter queries instead of several properties.

**uniqueKey**

Replaces *uniqueQueryKey*. Instead of having one *uniqueQueryKey* boolean on each field, one *uniqueKey* contains the list of unique fields.

Form properties benefit from a new initializer functionality which can set its value from another property.

Web components locations are specified with the environment variable *GSTWCDIR* and are no longer specified in **Tools > Preferences**. Environment set configuration now includes a Web Components set.

Table 346: Business Application Modeling

Overview

New dbapp3.0 template with improved functionality:

- Properties in Business Application diagram allow for customization of the UI behavior.
- Cascade delete supported in generated code.
- Web Services can be generated and provide CRUD operations.
- Improved architecture based on different layers: ui, webservice, uidata, database.
- Improved insert, update, delete in the data layer.
- Improved libdbapp architecture.
- Dialog modularization used. Declarative **DIALOG** blocks are now defined as module elements and reused with the **SUBDIALOG** keyword of procedural **DIALOG** blocks. (See **DIALOG Block Structure** in the *Genero Business Development Language User Guide*.)
- Centralized actions in the generated code.
- The business record is checked to confirm that the unique key is defined as a primary key or a secondary key in the database schema.
- The foreign key database constraint existence is checked.
- Code generation uses unique constraint instead of unique index.
- Constraints on the relations of Business Application diagram entities can be defined in the settings.agconf file. Validation is done on the constraints during editing.
- Added native serial management for Informix® database.
- Manages concurrent access.
- New support for business record and form files without a database or a masterTable property set.
- New relation properties allow for specifying the row position in an opened form and applying a SQL filter while opening a form.

Reference

See *Links and Data order* on page 539.

See *Form item properties* on page 492.

See *GSTWCDIR* on page 168, Migration notes for GST 2.50 upgrade guide on page 1211.

See the default template features on page 230

Form behavior in CRUD states on page 293

Cascade delete on page 284

Add Web services on page 250

Add constraints or indexes on page 350

Managing SERIALs in a generated application on page 283

Managing concurrency on page 284

Add Relations on page 270
Overview

Business Application Modeling template files locations are specified with the environment variable GST_SETUPDIR and are no longer specified in Tools > Preferences. Environment set configuration now includes a set for each available template set.

Create BA entities in the current view only instead of all views. New BA entities are only visible in the current view and are automatically hidden in the other views.

A new Toolbar has been added in the Business Application Modeler. This Toolbar contains all the items defined for the Business Application Modeler in the settings.agconf file.

Orphan properties now managed. Templates used in Application Generator define dynamic properties that can be used in objects such as records, form fields, and database objects. Opening a file generated with a template version different than the one set in Application Generator preferences may produce a warning indicating that some properties are not found in the current template definition. New warnings signal that those properties can now be removed using the Tools > Specific setup > Clean orphan properties menu option.

Added built-in Find.

Improved performance for large 4DBX database meta-schema files.

Table 347: Meta-schema Manager

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhanced Revert action. A dialog is displayed listing changes made on the selected items in the schema with option to select which changes to revert.</td>
<td>See Revert schema changes dialog on page 373.</td>
</tr>
<tr>
<td>Database &gt; Diff Schema... option.</td>
<td>See Using the Diff tool on page 451.</td>
</tr>
<tr>
<td>Generate Database Update Script option. The Meta-schema Manager is able to generate a database update script. This script will modify the database according to the changes made in the Meta-schema. Previous version of the tables will be backed up and the data will be migrated to the new tables when applicable.</td>
<td>See Generate a database script from meta-schema on page 359.</td>
</tr>
<tr>
<td>Improved database creation script. The database creation script now supports Secondary Keys and case sensitivity and has improved support for Oracle® MySQL and SQLite databases.</td>
<td></td>
</tr>
<tr>
<td>Improved Locate in Diagram action.</td>
<td>See Meta-schema diagram context menu on page 376.</td>
</tr>
<tr>
<td>Create database objects in the current view only instead of all views. New Meta-schema objects are only visible in the current view and are automatically hidden in the other views.</td>
<td>No further reference.</td>
</tr>
<tr>
<td>Secondary Keys (also known as Unique Constraints) have been added in the Meta-schema Manager.</td>
<td>See Add constraints or indexes on page 350.</td>
</tr>
</tbody>
</table>
### Overview

Global meta-schema files are now specified with the environment variable `GSTSCHEMANAMES` and are no longer specified in **Tools > Preferences**. Environment set configuration now includes a set for **Global Database Schemas**.

### Table 348: Graphical Debugger

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack frames support, provided Genero 2.50 DVM is used. Navigate the call stack. Data view is automatically updated according to the current frame.</td>
<td>See <strong>Examining execution flow</strong> on page 596.</td>
</tr>
<tr>
<td>User variables of type ARRAY or RECORD are displayed in the Data view. Values can be re-assigned for complex types.</td>
<td>See <strong>Examining data</strong> on page 595.</td>
</tr>
<tr>
<td>Enhanced performance when retrieving and displaying data. Support of signal <code>SIGINT</code> and signal <code>SIGQUIT</code>.</td>
<td>No further reference.</td>
</tr>
</tbody>
</table>

### Table 349: Source Code Management (SCM)

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVN Blame command support and integration with Code Editor and SVN Lock command with a new view to manage all locks. Status view optimized for unversioned files. User and password are based on repository. Different users can now connect to different repositories at the same time. <strong>Ignore List</strong> action now adds to the ignore list of the current directory only, not recursive up to the checkout directory. SCM tasks are properly interrupted to avoid lock on the checkout. Authentication is global to the checkout and is no longer folder based.</td>
<td>See <strong>Blame</strong> on page 634 and <strong>Locking</strong> on page 627.</td>
</tr>
</tbody>
</table>

### Table 350: Genero Report Designer

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables support. Pivot tables support. PDF enhancements. Improved PDF output, to include better memory consumption, use of the PDF referencing mechanism to improve Page M of N processing, share recurring images and CID keyed fonts support. Null value support. Improved trigger updates. Algorithm improved to remove the need for frequent manual adjustments for each change within the data schema (rddl) file.</td>
<td>See <strong>Working with tables</strong> on page 819. See <strong>Working with pivot tables</strong> on page 836. No further reference.</td>
</tr>
<tr>
<td>Overview</td>
<td>Reference</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>QR code barcode support.</td>
<td>See qr-code on page 941.</td>
</tr>
<tr>
<td>Display position of footers. Layout nodes designated as footers display at the bottom of the Mini Page, providing a WYSIWYG experience for the report designer.</td>
<td>See Add headers and footers to a report on page 792.</td>
</tr>
<tr>
<td>Element creation by context. Create elements based on the document context in the report design. The object type created for a field is determined by the location in the document.</td>
<td>See Adding data values and captions on page 797.</td>
</tr>
<tr>
<td>Splitting of oversized elements across pages to prevent overfill.</td>
<td>See splitOversizedItem (Split Oversized Items) on page 892.</td>
</tr>
<tr>
<td>Rotation of items. When the Transform transparently property is set on a parent, its children map their orientation based on the parent's parent orientation rather than the parent.</td>
<td>See Transform transparently (transformTransparently) on page 894.</td>
</tr>
<tr>
<td>Support for reverse side printing.</td>
<td>See Specify a different reverse side page on page 803.</td>
</tr>
<tr>
<td>Chart sorting. For MapCharts and CategoryCharts, the sortBy property allows you to specify how the data is sorted: alphabetic, numeric, or by order of declaration of the chart items. The sortAscending property allows you to sort in ascending or descending order.</td>
<td>See sortBy (Sort By) on page 902 and sortAscending (Sort Ascending) on page 901.</td>
</tr>
<tr>
<td>Fallback image support when the requested image for an Image Box is not found.</td>
<td>See Image Box on page 865.</td>
</tr>
<tr>
<td>Edit triggers with a Repeat selected items on menu option in the context menu in the Report Structure view, allowing you to select a trigger to be the parent of a document node.</td>
<td>See Place a trigger within the report structure on page 792.</td>
</tr>
<tr>
<td>Class property added for report elements.</td>
<td>See class (Class) on page 879.</td>
</tr>
<tr>
<td>Display and modify the sizing policy of containers.</td>
<td>See Modify the sizing policy of containers on page 785.</td>
</tr>
<tr>
<td>The fidelity property has been added to business charts and the pivot table, applied only when the object in question is drawn as a table.</td>
<td>See Business Graphs on page 869.</td>
</tr>
<tr>
<td>The layout direction of a parent container is highlighted in the Genero Report Designer by the addition of a dashed, slowly moving, U-shaped yellow border.</td>
<td>See layoutDirection (Layout Direction) on page 887.</td>
</tr>
<tr>
<td>Preference added to control the appearance of RTL expressions in the document view.</td>
<td>See Report Writer preferences on page 982.</td>
</tr>
<tr>
<td>Added options to facilitate the mass generation of images that are sized by their content (e.g. for web sites).</td>
<td>See fgl_report_setImageUsePageNamesAsFileNames on page 748 and fgl_report_setImageShrinkImagesToPageContent on page 748.</td>
</tr>
</tbody>
</table>
Table 351: Genero Reporting APIs

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the server where a Genero Report Engine is running in server mode.</td>
<td>See fgl_report_configureDistributedProcessing on page 730.</td>
</tr>
<tr>
<td>Configure the environment when the daemon is running on a different machine with different resource directories.</td>
<td>See fgl_report_configureDistributedEnvironment on page 730.</td>
</tr>
<tr>
<td>Distinguish between log entries originating from different users.</td>
<td>See fgl_report_setDistributedRequestingUserName on page 748.</td>
</tr>
<tr>
<td>“Postscript” has been added as an output format to the function fgl_report_selectDevice. The function fgl_report_setPrinterWriteToFile is deprecated.</td>
<td>See fgl_report_setPrinterWriteToFile on page 761 and fgl_report_selectDevice on page 743.</td>
</tr>
<tr>
<td>The Reporting API source has been split into two files, libgre.4gl and libgreprivate.4gl. These two files replace helpers.4gl.</td>
<td>See Reporting API Functions on page 719.</td>
</tr>
<tr>
<td>Function to control the paper orientation.</td>
<td>See fgl_report_setSVGOrientationRequested on page 764.</td>
</tr>
<tr>
<td>Switch off Unicode embedding. This is useful when the entire report uses Latin characters only, improving performance and document size.</td>
<td>See fgl_report_configurePDFFontEmbedding on page 736.</td>
</tr>
<tr>
<td>The function fgl_report_configureAutoformatOutput has changed from &quot;sort by position of fields in PRINT statement&quot; to &quot;sort by matched pattern and then by position of field in PRINT statement.&quot;</td>
<td>See fgl_report_configureAutoformatOutput on page 728.</td>
</tr>
<tr>
<td>To follow the general pattern that calling a function with a NULL value exhibits the same behavior as not calling the function at all, the default values for the first three first parameters of the function fgl_report_configureImageDevice have changed to true.</td>
<td>See fgl_report_configureImageDevice on page 731.</td>
</tr>
<tr>
<td>Runtime localization. Report can now be localized independent of the language settings of the application.</td>
<td>See Change localization settings at runtime on page 717 and fgl_report_configureLocalization on page 733.</td>
</tr>
</tbody>
</table>

GST 2.41 new features
This topic lists features added for the 2.41 release of Genero Studio.

Table 352: General

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Application Modeling (BAM) includes a new default Database Applications template, dbapp2.0.</td>
<td>See The code generation template set on page 1110, Migrating to a new BAM template set on page 1215</td>
</tr>
</tbody>
</table>
Overview | Reference
---|---
The dbapp template in Genero Studio 2.40 was renamed in Genero Studio 2.41 to dbapp1.0. | See The code generation template set on page 1110
$dbapp2.0 includes new build.rules file for dedicating a set of build rules to a template. | See Build tab on page 429.
A new database schema file type, $4dbx$, is used for generated applications. | See Database meta-schema ($4dbx$) on page 276

**GST 2.40 new features**

This topic lists features added for the 2.40 release of Genero Studio.

**Table 353: General**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
</table>
| Update settings in Source Code Manager have been updated. For example, you can now omit externals. | See Update / Update All on page 630.
| You can now add multiple files. | See Add files on page 626.
| The Meta-Schema Manager has been updated for easier editing and viewing. Options have been added in Edit mode, and document errors and warnings are displayed while editing. | See Meta-schema Manager on page 343.
| Multiple views of the same database can be created. | See Viewing a meta-schema on page 356.
| With Business Application Modeling, new business records have been added for reports. | See Add a Report Design Document (4rp) on page 248.
| With Business Application Modeling, an improved editor assists with search and document errors. | No further reference.
| With Business Application Modeling, new templates for a new architecture and for the generation or report code are provided. | See The code generation template set on page 1110.
| With Business Application Modeling, advanced customization is now possible. | See BAM Template Developer Guide on page 1094.
| With Business Application Modeling, an integrated code analyzer is provided. | See Code Analyzer on page 482.
| With Report Writer, data matrix barcodes are supported. | See Bar Code type details on page 912.
| With Report Writer, new properties for Text and Image fields allow them to behave as hyperlinks. | See Use hyperlinks in a report on page 814.
| With Report Writer, additional options have been added for MapCharts. | See Map Chart
| With Report Writer, there is a new function to create a process-level data file and a document at the same time | See $fgl_report_setProcessLevelDataFile$ on page 762.
| With Report Writer, there are new functions to create document metadata for compatibility reports | See Report Design Document metadata on page 796.
Overview

With Report Writer, there are new functions to provide generic report auto-formatting when no report design document (4rp) is specified for a report

With Report Writer, there are additional new report API functions, indicated by a "New in 2.40" designation.

Additional support for application rendering with the Genero Web Client.

You can now import configuration setup and preferences of earlier versions of Genero Studio when you first start up a new installation of Genero Studio.

For ease of use, the Window menu allows you to select the default layout, and you can toggle Full Size Documents or Full Screen to maximize your working space.

States of previous sessions are remembered.

Support of high contrast mode and of Windows® screen readers for accessibility standards.

Multiple instances of Genero Studio can be opened simultaneously.

Various performance improvements have been implemented.

Reference

See Auto-formatting Reports that have no 4rp (Report Design Document) on page 684.

See Reporting API Functions on page 719.

See Genero Configuration Management dialog on page 197.

See General Preferences on page 128.

No further reference.

See Workspaces Configuration on page 131.

See Configuring Genero Studio on page 138.

See Configuring Genero Studio on page 138.

No further reference.

--

Upgrade Guides for GST

Review the list of migration recommendations each time you move to a new version.

GST 3.10 upgrade guide

Review when migrating to Genero Studio 3.10.

**ANDROID_SDK_ROOT replaces ANDROID_HOME**

The ANDROID_HOME environment variable is deprecated. To define the Android SDK installation directory, use ANDROID_SDK_ROOT instead.

**Note:** The gmbuildtool will first use ANDROID_SDK_ROOT, then ANDROID_HOME as fallback.

If you previously used ANDROID_HOME, you can continue to use it but consider changing your environment settings to follow Android SDK specifications.

**Business graphs now use "flat design" by default**

Business graphs in 3.10 look significantly different to graphs from older versions. By default, they use a more modern "flat design", with fewer 3D effects, gradients, and shadows. You can revert to "classic design"; see **Specifying classic or default style** on page 832.
GST 3.00 upgrade guide
Review when migrating to Genero Studio 3.00.

Genero Report Engine deployment
When in distributed mode, if the Genero Report Engine sits on a different machine than the runtime system (DVM) processing the report, you must do the following in order to preview reports:

- Start the Web server service (grehttpd).
- Set the GREOUTPUTDIR environment variable.

See Distributed Mode on page 1015 for more details.

GST 2.50 upgrade guide
Review when migrating to Genero Studio 2.50.

Add Remote Host Configurations
Configuration information is now kept on the host. This means that when a user connects to a remote host from a client, the host's configurations about the compiler, environment sets, GDC and Genero web client displays are available. To use your configurations from a prior version, you will need to follow these steps to re-associate the information to a named configuration. The named configurations are not transferred in 2.50, but the information to create them is available.

1. Start Genero Studio 2.50 on the client.

2. Select your remote host from the list in the lower right corner of Genero Studio. If you do not see your host in the list, follow these steps to add your remote host(s). See Add a remote host on page 180.

3. Select the wrench icon to display the Genero Configuration Management dialog on page 197. This dialog has changed to allow easy access to all configuration dialogs.

   Figure 462: Select remote host and then configurations

4. Select the Import Configuration button. You will be prompted to choose a Genero Studio installation from which to import.
5. Select Import. Genero installations are imported and added to the Genero Installation list. Environment sets are imported and added to the Environment Sets list.

6. If you wish, add new Configuration Names to the list using the Add button. All of the configuration information is available on the remote host to use in your named configurations. The named configurations, however, are not imported and you may want to recreate a new named configuration in 2.50 for each prior version configuration that you had listed. Each configuration contains information about:

   - Compiler / Runtime configuration (Genero Installations) on page 200
   - Environment sets on page 164
   - Configure for the Desktop client on page 170
   - Configure for the Web client on page 171

Figure 463: Import configurations

Figure 464: Add a named configuration
BAM Templates - GSTSETUPDIR

Business Application Modeling template files locations are specified with the environment variable GSTSETUPDIR and are no longer specified in Tools > Preferences.

An environment set is listed for each default template set. Select the correct template set to be used.

If you use a custom template set, you will need to create an Environment Set and set GSTSETUPDIR on page 168 to the location of your template files. Be sure to check mark your environment set in the configuration(s) for which it is being used. See Migrate customized template sets on page 1218 for specific instructions.

Web Components - GSTWCDIR

Web components locations are specified with the environment variable GSTWCDIR and are no longer specified in Tools > Preferences.

If you use web components, you will need to create an Environment Set and set GSTWCDIR on page 168 to the location of your web components. An environment set named Web Components is listed in the Environment Sets list and can be used as your web component environment set by setting its GSTWCDIR to the location of your web components. Be sure to check mark your environment set in the configuration(s) for which it is being used.

Meta-Schemas - GSTSCHEMANAMES

It is recommended that you add schemas to projects so that they are loaded when the project is opened (and not at Genero Studio launch) and so that the project can be available to all developers without any additional configuration needed. However, specifying global schemas is still supported. Global meta-schema files are now specified with the environment variable GSTSCHEMANAMES and are no longer specified in Tools > Preferences.

To make meta-schemas available to all projects and to appear in the DB Schemas tab, you can use the Global Database Schemas Environment Set and define the two environment variables within the environment set:

- GSTSCHEMANAMES on page 167: Defines the filenames of the schemas to make available. (Do not include file extension.) Use the Value List environment variable type to list multiple meta-schemas separated by semicolons.
- FGLDBPATH: Defines the directories in which to find the schema files entered in the GSTSCHEMANAMES variable.

Be sure to check mark your environment set in the configuration(s) for which it is being used.

Genero Report Writer

The fieldNamePatterns input parameter for the reporting API function fgl_report_configureAutoformatOutput() has changed from "sort by position of fields in PRINT statement" to "sort by matched pattern and then by position of field in PRINT statement." If you have a report application that uses this function to define the output of an auto-format report that is not of a COMPATIBILITY type, verify that the report formats as expected and make modifications if necessary.

See fgl_report_configureAutoformatOutput on page 728.

Prior to version 2.50, localization information (FGLPROFILE, FGLRESOURCEPATH, DBPATH, DBFORMAT and so on) was statically defined at the start of a program and could not be changed at runtime. A work-around at the time was to place reports in separate executables, then to run these executables from the main application with a modified environment. This achieved the effect of modifying the localization configuration on a per report basis. This work-around will not work with reports run in distributed mode; the fgl_report_configureLocalization function should be used instead.

See Change localization settings at runtime on page 717.

Related concepts

Environment sets on page 164
Define sets of environment variables to add to or overwrite the default environment. These environment settings are also usable by each Genero Studio client that adds a remote Genero Studio Server to its list.

**GST 2.41 upgrade guide**
Review when migrating to Genero Studio 2.41.

**Application Generator**
- New default Database Applications template, dbapp2.0
- If you already have a BAM project, you should change the current template used to the environment set for the new template. See Environment sets on page 164.
- dbapp2.0 includes new build.rules file for dedicating a set of build rules to a template.
- The dbapp template in Genero Studio 2.40 was renamed in Genero Studio 2.41 to dbapp1.0. The dbapp template set is the same template set as dbapp1.0.

**Related concepts**
- Migrating to a new BAM template set on page 1215
Topics to assist you with upgrading your generated application from one template set to another.
- Environment sets on page 164
Define sets of environment variables to add to or overwrite the default environment. These environment settings are also usable by each Genero Studio client that adds a remote Genero Studio Server to its list.

**GST 2.40 upgrade guide**
Version 2.40 migration notes.

The format of all Genero Studio XML files (4fd, 4rp, 4pw etc.) has been modified, and you will receive a warning if you execute an application compiled using Genero Studio versions 2.2x or 2.3x. For Genero Report Writer reports only, an error will occur if you have not converted the 4rp file. Opening a file in Genero Studio and saving it will convert it.

**Form Designer**
For Genero Studio form definition files (4fd), the gsform command-line tool can be used for a batch conversion of the files to version 2.40:

```
gsform -c <files>
```

**Application Generation**
The format of the settings.agconf file in the templates directory (<Studio-install-dir>/gst/bin/src/ag/tpl/dbapp/settings.agconf) has changed. If you have previously modified this file, you must make comparable changes in the 2.40 version of this file.

**GST 2.30 upgrade guide**
Version 2.30 migration notes.

**Report Writer**
- TOP Margin is ignored in compatibility reports.

  TOP MARGIN is now ignored when ASCII reports are run in compatibility mode. Margins can be set by using the API function fgl_report_setPageMargins().
- Parameter changed in fgl_report_configureCompatibilityOutput().
The `reportName` value for the `fgl_report_configureCompatibilityOutput` function is now automatically set to the name of the currently running report. When using the function, however, you must pass NULL as the value, for compatibility reasons.

- **4GL Boolean type is now interpreted as Numeric instead of String.**
  
  The 4GL Boolean type is now regarded as a Numeric type, instead of String as in previous versions. Existing expressions that contain references to boolean variables may require modification accordingly.

- **START REPORT syntax must be TO XML HANDLER.**

  The BDL statement `START REPORT <repname> TO XML HANDLER handler` is now the only valid syntax for starting graphical reports using a 4rp design document.

### GST 2.20 upgrade guide

Version 2.20 migration notes.

**Important:** Genero Studio Form files (4fd), Project files (4pw), and Report files (4rp) are converted to the Genero Studio 2.20 format when opened. We strongly advise making a backup of your files before migrating them.

#### Meta-Schema Manager (formerly Database Browser)

1. Database Browser is replaced by Meta-schema Manager.
2. Schema availability has changed.

#### Form Designer

1. Form files (4fd) will be converted to the Genero Studio 2.20 form format when opened.
2. The Matrix container replaces the MFArray container.

#### Project Manager

1. Project files (4pw) will be converted to the Genero Studio 2.20 project format when opened.
2. The Genero variable `FGLRESOURCEPATH` defines the path to resource files.

#### Report Writer

1. Report files (4rp) will be converted to the Genero Studio 2.20 project format when opened.
2. The Reporting API is now linked as one library, `libgre.42x`.
3. The callback function `fgl_report_getFieldCaption()` is no longer called by default.
4. The Configuration menu has moved to File>>Report Properties.

### Migrating to a new BAM template set

Topics to assist you with upgrading your generated application from one template set to another.

- **Migrate from dbapp3.2 to dbapp4.0 on page 1216**
- **Migrate from dbapp3.1 to dbapp3.2 on page 1216**
- **Migrate from dbapp3.0 to dbapp3.1 on page 1216**
- **Migrate from dbapp2.0 to dbapp3.0 on page 1216**
- **Migrate from dbapp1.0 to dbapp2.0 on page 1217**
- **Migrate from 2.3x to dbapp1.0 on page 1217**
- **Migrate customized template sets on page 1218**
Migrate from dbapp3.2 to dbapp4.0

To migrate your generated application from the dbapp3.2 template set to the dbapp4.0 template set, you simply need to confirm that you are using the dbapp4.0 template set.

The template dbapp4.0 provides the following benefits:
• The DBAPP_MOBILE environment variable can be set to 1 (TRUE) for generating applications for mobile devices. See DBAPP_MOBILE on page 320.
• In settings.agconf, the editorInfo attribute for MULTIPLELINES has changed for dialog titles to match other editorInfo syntax:
  • bannerTitle is now description.
  • dialogTitle is now title.

See settings.agconf elements on page 1122 for details on the settings.agconf

Migrate from dbapp3.1 to dbapp3.2

To migrate your generated application from the dbapp3.1 template set to the dbapp3.2 template set, you simply need to confirm that you are using the dbapp3.2 template set.

The template dbapp3.2 provides the following benefits:
• The canDisplay and canEmpty functionalities can be specified for a form.
• Concurrent access management is disabled for generated mobile apps, as there is no need for a mobile app to manage concurrent access.

1. Select Tools > Genero Configurations.
2. In the Environment sets on page 164 list, select the environment set named Template dbapp3.2. This should be the only template environment set selected.

Migrate from dbapp3.0 to dbapp3.1

You can migrate your generated application from the dbapp3.0 template set to the dbapp3.1 template set.

1. Confirm that you are using the dbapp3.1 template set. Select Tools > Genero Configurations.
2. In the Environment sets on page 164 list check mark the environment set named Template dbapp3.1.
3. Data refresh on page 295 is a new feature in the dbapp3.1 template and affects relations between forms. After migrating templates, the default value of the property is Current Row, therefore if you want to have the same functional behavior as in dbapp3.0, you must set the Data Refresh property to None on all relations between forms.

Migrate from dbapp2.0 to dbapp3.0

You can migrate your generated application from the dbapp2.0 template set to the dbapp3.0 template set.

1. Confirm that you are using the dbapp3.0 template set. Select Tools > Genero Configurations.
2. In the Environment sets on page 164 list check mark the environment set named Template dbapp3.0.
3. Confirm that your business records each have a unique key. Set the unique key if it has not already been defined as a primary key or secondary key in the database meta-schema.
4. The Source Field and Destination Field properties have been replaced by the Source Field in the group Position and by Source Field/Destination Field in the group Filter. Consequently, you may need to update some relations in your Business Application Diagram. When you open your existing BA diagram, the values of the orphan property Source Field are in the Source Field of the group Position and the Destination Field property becomes an orphan property.

a) Update your relations between two Forms.

• In the Filter group, update the Source Field property with the same values as those in the Source Field property in the group Position.
• Update the Destination Field with the same values as those in the Destination Field.
• In the **Position** group, reset the values specified in the property **Source Field**.

b) Update your relations between a Form and a Zoom

• No update needed.

5. The dbapp3.0 template set is a new application architecture thus some BLOCK/POINT sections are in different source files. To migrate your modified BLOCK/POINT sections, please contact your support center.

**Related concepts**

*Using templates with another version of compiler* on page 1218

All templates are pre-compiled with the current version of Genero. If you wish to use the templates with a different version of the compiler, you must recompile them with that version.

**Migrate from dbapp1.0 to dbapp2.0**

You can migrate your generated application from the dbapp1.0 template set to the dbapp2.0 template set.

1. Modify your 4pw project file by removing any Application Generator file build rules. Delete the duplicate rules from the **Edit Build Rules** dialog so that your program will use the new template build rules.

2. The dbapp2.0 build rules are in a file called build.rules. If you have your own template directory, you must copy the build.rules file from the dbapp2.0 template directory. If you have modified the build rules, re-integrate your changes in the build.rules file.

3. The dbapp2.0 template uses the 4dbx database schema file type. Before using the dbapp2.0 template, replace the 4db files with 4dbx ones.

4. Open the 4db file and use **Save As** to save the file with a 4dbx extension. If you have defined your own template directory:
   a) Confirm that you have the 4dbx file type definition.
   b) Confirm that the actions are present in the creatable.conf.
   c) Remove the build rules from your projects if you have added them (or update them to execute the same operations as the provided ones).
   d) Confirm that the build.rules file contains the same build rules as the provided one for 4dbx.
   e) Confirm that the settings.agconf contains the 4dbx section.

5. Report instruction names have changed. For each 4rp file, you need to select its data source again. The data source can be created by building your 4rd file.

6. Confirm that you are using the dbapp2.0 template set. Select **Tools >> Preferences, Application Generator** and select **Database Applications 2.0**.

**Migrate from 2.3x to dbapp1.0**

You can migrate your generated application from a 2.3x template set to the dbapp1.0 template set.

This procedure assumes you have not modified your source files with code in BLOCK and POINT sections or modified the Tcl files. If so, you will have to re-generate the program and then manually add your code changes to the newly generated code.

1. Open the project to be migrated from 2.3x to dbapp1.0.

2. Right-click on the 4pw and select **Edit Build Rules**. Remove any user-defined build rules.

3. Save and close your project.

4. Select **Tools > Genero Configurations** and select the dbapp 1.0 environment set. This will now be the code generation template used for the project.

5. Open your project.

6. Remove libag.42m from the **External Dependencies** of the **Library** node and save the project.

7. The CRUD Form and Zoom Form types replace the Module type. Open the Business Application Diagram. For every Module entity corresponding to a form, right-click on the entity and select **Convert to > CRUD Form**. For every Module entity corresponding to a zoom, right-click on the entity and select **Convert to > Zoom Form**.

8. Save the project.
9. Rebuild the project.

**Migrate customized template sets**

You can migrate your customized template set.

1. Confirm that you are using your template set. Select **Tools > Genero Configurations**.
2. In the **Environment sets** on page 164 list check mark the environment set for your template, or create one pointing **GSTSETUPDIR** on page 168 to your template set.
3. Open the **settings.agconf** file and modify the version attribute of the root element to 5. `<AGSettings version="5">`
4. Open the other settings files in the Code Editor and check if there are errors (file-types.xml, creatable.conf, and build.rules).
5. In case of errors, review the template setting files to which you want to migrate and modify your settings to match the new file format.
6. A new template may generate different code. If you have modified the template files, you may lose those changes when using a new template set. If you want to use a new template, you may have to manually integrate your changes into the new template set.

If the Application Generator model is identical or there is no impact on your template files or you are not interested in the new template features, you can keep your templates unchanged. If you have modified the generated source (4gl) code in a BLOCK or a POINT, changing the current template does not delete your changes. If the corresponding BLOCK/POINT still exists in the template, your changes will be integrated during the next compilation. If the BLOCK/POINT has been removed from the template, it will appear at the end of the file in a lost BLOCK/POINT, that you can reintegrate into your code (with copy/paste, for example).

**Using templates with another version of compiler**

All templates are pre-compiled with the current version of Genero. If you wish to use the templates with a different version of the compiler, you must recompile them with that version.

To recompile templates:

1. Configure Genero Studio to specify the location of your **Genero runtime / compiler**.
2. Confirm that your environment has **GSTSETUPDIR** pointing to the location of the templates that you want to recompile.
3. Open the project file (`4pw`) in **GSTSETUPDIR** and recompile.

---

**Framework error messages**

Error messages at the framework level can apply to various components of Genero Studio. Each component also maintains their own set of error messages.

**Table 354: Framework error messages**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-1001</td>
<td>Unable to start process: command line empty</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1002</td>
<td>Process not running, nothing to abort. Command line: <code>command</code></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1003</td>
<td>Process not started, nothing to abort. Command line: <code>command</code></td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-1004</td>
<td>Process failed to start. Check executable files existence and permissions. Command line: command Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1005</td>
<td>The process crashed some time after starting successfully. Command line: command Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1006</td>
<td>Unable to write to the standard input of a non-running process. Command line: command Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1007</td>
<td>Unable to read from the standard output or error of a non-running process. Command line: command Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1008</td>
<td>An unknown error occurred on process. Command line: command Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1020</td>
<td>There is no display client running on host:port A connection to the display client failed. Ensure that the display client has started on the host and port.</td>
</tr>
<tr>
<td>GS-1021</td>
<td>File filename not found, unable to run the preview. Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1023</td>
<td>Invalid display client directory The startup script of the display client was not found. Check the display client settings in the configuration.</td>
</tr>
<tr>
<td>GS-1024</td>
<td>Failed to start display client in pathname Check the display client settings in the configuration.</td>
</tr>
<tr>
<td>GS-1025</td>
<td>Display client already running on host:port Stop the running display client or check the display client settings in the configuration.</td>
</tr>
<tr>
<td>GS-1026</td>
<td>Current Genero configuration's FGL is invalid. Unable to verify if client has correctly started. Check the Genero install settings in the configuration.</td>
</tr>
<tr>
<td>GS-1027</td>
<td>Display client has been successfully started on host:port Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1028</td>
<td>A variable needed to start display client is not set: varname Check the environment settings in the configuration.</td>
</tr>
<tr>
<td>GS-1029</td>
<td>Client on port portname is not GDC/GMA/GMI Another application is using the port and is not a display client. Stop the application or check the display client settings in the configuration.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-1031</td>
<td>Using remote display client</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1032</td>
<td>Cannot connect to display client on host:port. Verify that your display client configuration is correct.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1034</td>
<td>Unable to create path pathname</td>
</tr>
<tr>
<td></td>
<td>Check the access permissions on the path.</td>
</tr>
<tr>
<td>GS-1035</td>
<td>Unable to copy filename, the file doesn't exist</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1036</td>
<td>Unable to copy filename to destination, check the permissions of the original file, of the destination directory and the destination file if it already exists.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1037</td>
<td>Failed to create tools directory.</td>
</tr>
<tr>
<td></td>
<td>Check the access permissions on the tools directory.</td>
</tr>
<tr>
<td>GS-1038</td>
<td>Failed to generate application configuration file (filename)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1039</td>
<td>Application configuration file generated (filename)</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1040</td>
<td>Unable to move filename, the file doesn't exist</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1041</td>
<td>Unable to move filename to destination, check the permissions of the original file, of the destination directory and the destination file if it already exists.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1042</td>
<td>Unable to delete filename, check the permissions of the original file and its directory.</td>
</tr>
<tr>
<td></td>
<td>Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1043</td>
<td>Unable to start the proxy process. Check your product installation.</td>
</tr>
<tr>
<td></td>
<td>It might be necessary to reinstall the product.</td>
</tr>
<tr>
<td>GS-1044</td>
<td>The proxy process crashed. Please contact product support.</td>
</tr>
<tr>
<td></td>
<td>Contact your support center to report the issue.</td>
</tr>
<tr>
<td>GS-1045</td>
<td>Unable to write to the proxy process. Check your product installation.</td>
</tr>
<tr>
<td></td>
<td>It might be necessary to reinstall the product.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-1046</td>
<td>Unable to read from the proxy process. Check your product installation. It might be necessary to reinstall the product.</td>
</tr>
<tr>
<td>GS-1047</td>
<td>Unable to start the DVM process. Check your product installation. Check the Genero install settings in the configuration.</td>
</tr>
<tr>
<td>GS-1048</td>
<td>The DVM process crashed. Please contact product support. Check the Genero install settings in the configuration.</td>
</tr>
<tr>
<td>GS-1049</td>
<td>Unable to write to the DVM process. Check your product installation. Check the Genero install settings in the configuration.</td>
</tr>
<tr>
<td>GS-1050</td>
<td>Unable to read from the DVM process. Check your product installation. Check the Genero install settings in the configuration.</td>
</tr>
<tr>
<td>GS-1051</td>
<td>Unable to start the proxy server. Contact your support center to report the issue.</td>
</tr>
<tr>
<td>GS-1052</td>
<td>An error occurred on the socket connection to the proxy. Contact your support center to report the issue.</td>
</tr>
<tr>
<td>GS-1053</td>
<td>The socket connection to the proxy has been closed. Contact your support center to report the issue.</td>
</tr>
<tr>
<td>GS-1054</td>
<td>Path is not accessible from the server. Check your mount points. The paths configured in the Genero Host are not accessible. Check the Genero Host settings of the configuration.</td>
</tr>
<tr>
<td>GS-1055</td>
<td>Launching application at url Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1056</td>
<td>Cannot start web browser. Check path in Genero configuration and preferences Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1057</td>
<td>Launching GAS monitor at url Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-1058</td>
<td>The following warnings generated during xcf file generation A Run task generated an application configuration file warnings. Those warnings are listed after this message.</td>
</tr>
<tr>
<td>GS-1059</td>
<td>Environment variable varname contains invalid characters Check your environment settings in the Genero Configuration or at the system level.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>GS-1060</td>
<td>Unable to write the file <code>filename</code>. Check path existence and permissions. <strong>Error message and resolution should be self-explanatory.</strong></td>
</tr>
</tbody>
</table>
| GS-1061 | Invalid URL `url`  
Check the Genero Application Server settings in your configuration. |
| GS-1062 | Invalid configuration  
Check your Genero configuration status and settings. |
| GS-1063 | `Java_error`  
The Java compiler output an error. |
| GS-1064 | Cannot find text codec for encoding `encoding`  
Check the encoding and the encoding map are valid. |
| GS-1065 | `Java compilation server error : error`  
Check the error details for more information. |
| GS-1066 | Incorrect environment variable `varname`  
Check your environment settings in the Genero Configuration or at the system level. |
| GS-1067 | Command line too long  
A task tried to execute a command which exceed the system length limit. |
| GS-1068 | Cannot load file-types.  
Check your template configuration settings and the `file-types.xml` file format. |
| GS-1069 | Unable to delete `filename`, file not found.  
Check that the file exists and that you have access permission. |
| GS-1070 | Cannot save file. Unable to create backup file, aborting. **Error message and resolution should be self-explanatory.** |
| GS-1071 | File `filename` was created with a newer version and cannot be opened.  
Try opening the file with a different version of the product. |
| GS-1072 | The command `command` failed with exit code `code`.  
Check the additional output to get details about the error. |
| GS-1073 | Executing `"action\"`  
Error message and resolution should be self-explanatory. |
| GS-1074 | Opening `"filename\"`  
Error message and resolution should be self-explanatory. |
| GS-1075 | Opening with shell `"filename\"`  
Error message and resolution should be self-explanatory. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-3001</td>
<td>Invalid model node info version. A file has been saved with an unsupported model node info version. Try opening it with the appropriate product version.</td>
</tr>
<tr>
<td>GS-3002</td>
<td>Invalid dynamic property version. A file has been saved with an unsupported dynamic property version. Try opening it with the appropriate product version.</td>
</tr>
<tr>
<td>GS-3003</td>
<td>Dynamic property property not found for node node Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-3004</td>
<td>Failed to add property name to model node info: a property with this name was not found Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-3005</td>
<td>Failed to add property name to model node info: a dynamic property with this name already exists A dynamic property has been defined twice in a model node info. The file cannot be loaded.</td>
</tr>
<tr>
<td>GS-3006</td>
<td>Failed to add property name to model node info: a static property with this name already exists A property has been defined twice in a model node info. The file cannot be loaded.</td>
</tr>
<tr>
<td>GS-3007</td>
<td>Model node info for type type not found. Properties have been defined for a non-existing model node information.</td>
</tr>
<tr>
<td>GS-3008</td>
<td>Failed to add property info name: a dynamic property with this name already exists Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-3009</td>
<td>Failed to add property info name: a static property with this name already exists Error message and resolution should be self-explanatory.</td>
</tr>
<tr>
<td>GS-3010</td>
<td>Failed to add property group name: a dynamic property group with this name already exists A dynamic property group has been defined twice. The file cannot be loaded.</td>
</tr>
<tr>
<td>GS-3011</td>
<td>Failed to add property group name: a static property group with this name already exists A dynamic property has been defined twice. The file cannot be loaded.</td>
</tr>
<tr>
<td>GS-3012</td>
<td>Invalid property info version. A file has been saved with an unsupported property information version. Try opening it with the appropriate product version.</td>
</tr>
<tr>
<td>GS-3013</td>
<td>Failed to read dynamic property. An error occurred while reading dynamic properties.</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GS-3014</td>
<td>Found orphan property '%1'. Update your setup or clean document settings to remove it</td>
</tr>
</tbody>
</table>

Some examples of why this may occur:

**Projects view**
- User settings in the project file are not part of the current template.

**Meta-Schema Manager**
- The meta-schema file contains dynamic properties which are not defined in the current set of templates. It was likely created using a different set of template files.

**Form Designer / Business Record**
- The form contains a dynamic property which is not present in the specific settings. This error is generated when the specific settings do not match the document dynamic property settings. These orphan properties won’t be taken into account during compilation.

Possible solutions:

- The file was opened in the wrong setup. Change your Genero Configuration to load the correct setup (i.e., choose the correct template as defined by GSTSETUPDIR) and re-open the file.
- If the orphan property is not required, clear all orphan properties by selecting **Tools > Specific setup > Clean orphan properties.**
  
  **Warning:** The definition of all orphan properties and their entered data will be permanently discarded. Review the list of orphan properties in the **Document Errors** view before you clean orphan properties.

- If the orphan property is required, add the orphan property to the **settings.agconf** file.
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-3015</td>
<td>Found orphan property group '%1'. Update your setup or clean document settings to remove it</td>
</tr>
</tbody>
</table>

Some examples of why this may occur:

**Projects view**

User settings in the project file are not part of the current template.

**Meta-Schema Manager**

The meta-schema file contains dynamic properties which are not defined in the current set of templates. It was likely created using a different set of template files.

**Form Designer / Business Record**

The form contains a dynamic property which is not present in the specific settings. This error is generated when the specific settings do not match the document dynamic property settings. These orphan properties won’t be taken into account during compilation.

Possible solutions:

- The file was opened in the wrong setup. Change your Genero Configuration to load the correct setup (i.e., choose the correct template as defined by GSTSETUPDIR) and re-open the file.
- If the orphan property group is not required, clear all orphan properties by selecting **Tools > Specific setup > Clean orphan properties.**
  
  **Warning:** The definition of all orphan properties and their entered data will be permanently discarded. Review the list of orphan properties in the **Document Errors** view before you clean orphan properties.
- If the orphan property group is required, add the orphan property group to the **settings.agconf** file.
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-3016</td>
<td>Node '%1' contains orphan properties. Update your setup or clean document settings to remove them. The file settings for the node differ from the Genero Studio ones, either the Genero Studio settings are not up to date, or the file contains old settings.</td>
</tr>
<tr>
<td></td>
<td><strong>Project Manager</strong> User settings in the project file are not part of the current template.</td>
</tr>
<tr>
<td></td>
<td><strong>Meta-Schema Manager</strong> The meta-schema file contains dynamic properties which are not defined in the current set of templates. It was likely created using a different set of template files. Some database objects are using those obsolete properties.</td>
</tr>
<tr>
<td></td>
<td><strong>Form Designer / Business Record</strong> The form contains a dynamic property which is not present in the specific settings. This error is generated when the specific settings do not match the document dynamic property settings. These orphan properties won’t be taken into account during compilation.</td>
</tr>
</tbody>
</table>

Possible solutions:

- The file was opened in the wrong setup. Change your Genero Configuration to load the correct setup (i.e., choose the correct template as defined by GSTSETUPDIR) and re-open the file.
- If the orphan properties are not required, clear them by selecting **Tools > Specific setup > Clean orphan properties.**

  **Warning:** The definition of all orphan properties and their entered data will be permanently discarded. Review the list of orphan properties in the Document Errors view before you clean orphan properties.

- If the orphan properties are required, edit the settings.agconf file.

<p>| GS-6001  | Path is not accessible from the server. Check your mount points. A task tried to convert a path to a remote path but the server cannot access the file. Check the mount points in the Genero Host configuration. |
| GS-6002  | Invalid product version. Error message and resolution should be self-explanatory.                                                        |
| GS-6004  | Product bundle is up to date. Error message and resolution should be self-explanatory.                                                   |
| GS-6005  | Download started. Error message and resolution should be self-explanatory.                                                              |
| GS-6006  | Install cancelled: The md5 hash doesn't match. Error message and resolution should be self-explanatory.                                  |
| GS-6007  | Install cancelled: The file name doesn't match. Error message and resolution should be self-explanatory.                                 |
| GS-6008  | Update failed, extraction error. Error message and resolution should be self-explanatory.                                              |</p>
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
</table>
| GS-6009 | Update server returned an error: error  
The updater cancelled the installation because the update server returned an error. |
| GS-6010 | Invalid software update login.  
Error message and resolution should be self-explanatory. |
| GS-6011 | Update error: invalid product.  
Error message and resolution should be self-explanatory. |
| GS-6012 | Update error: invalid product version.  
Error message and resolution should be self-explanatory. |
| GS-6013 | Update error: invalid software update request.  
Error message and resolution should be self-explanatory. |
| GS-6014 | Download finished.  
Error message and resolution should be self-explanatory. |
| GS-6015 | Update successful.  
Error message and resolution should be self-explanatory. |
| GS-6016 | Update aborted.  
Error message and resolution should be self-explanatory. |
| GS-6100 | Cannot load creatable file.  
The creatable file is invalid and cannot be loaded. |

**Genero Glossary**

A glossary of terms used in Genero documentation.

**A**

**absolute positioning**

A layout where the objects are positioned at specific coordinates that do not change at runtime.

- **static layout**
- **positioned layout**

**active configuration**

The configuration currently set to active within Genero Studio. Only one configuration may be active at a time.

**application data**

The data manipulated by the product, typically managed by one or more database systems. Application data has a volatile state when loaded in the runtime system, and a static state when stored in the database system.
**Business Application Modeler**
A tool that allows you to visually model your report program and generate the code from the design models. You focus on the models while BAM handles the coding.

Business Application Modeler (BAM)

**Business Development Language**
A program language designed to write an interactive database application. BDL programs communicate with the database server using Structured Query Language (SQL).

BDL
FGL
FGLGWS
4GL

**Code Analyzer**
A tool in Genero Studio that reverse-engineers existing applications and can generate diagrams to provide an overview of the application.

**configuration**
A collection of environment sets. Each defined configuration is known by its configuration name. Configurations are defined in the Genero Configuration Management dialog.

**contextual menu**
A menu that displays when you right-click on an object in Genero Studio. It provides a list of actions available for the object.

**Data Model Diagram (.4rdj)**
A diagram that defines the records (data sets) that comprise the data model for your report application.

**Note:** Genero Report Writer

**data schema file**
The file used by the Genero Report Designer to provide a list of data objects for the report design. Supported schema formats are .xsd (XML schema), .rdd (4GL report schema), and .rsd (Genero template schema).

**Note:** Genero Report Writer
report schema file
Tip: The term report schema file is sometimes used in the Genero Studio user interface to refer to a data schema file.

data source
The source code that defines the data extracted and streamed to the report. When an application is generated, the data source is generated from the Data Model Diagram (.4rdj).
Note: Genero Report Writer

database meta-schema file
See meta-schema files.

DB Explorer
A Genero Studio tool that allows you to view, create and modify data stored in a relational database.

dependency diagram
A diagram that displays a graphical view of the complex relationships between the various pieces of a project. It shows the components that depend on other components, and have components that depend on them. Dependency diagrams are displayed in the Code Analyzer.

deprecated feature
A feature, design, or practice whose use is discouraged although not prohibited. Typically, a deprecated feature has been superseded or is no longer considered safe, but it is not yet removed from the system. Four Js provides support for deprecated features. Bugs will be fixed but enhancements will not be made.

design file

Designer
See Report Designer on page 1234.

desupported feature
A feature, design, or practice that is no longer supported. A desupported feature may still exist, but bugs will not be fixed. The code supporting the feature may be removed without notice.

dialog box
A modal window with a simple message and OK, OK|Cancel, or Yes|No|Cancel buttons.
In Genero, you can create dialog boxes using MENU ...
ATTRIBUTES(STYLE="dialog",COMMENT=dialog_text). For example:

```plaintext
ATTRIBUTES(STYLE="dialog",COMMENT="Do you want to delete the record?")
```

dialog

dynamic layout
See relative positioning on page 1234.

dynamic property
A dynamic property is a property added to a Genero Studio design file (such as a form file (.4fd), a project file (.4pw), a BA diagram file (.4ba), and so on) as defined in a settings files for a specific setup of Genero Studio. The GSTSETUPDIR environment variable specifies the directory where the settings files for the current specific
setup can be found. For example, when using the Business Application Modeler, dynamic properties can be defined by the settings.agconf file defined for the selected template.

**Dynamic Virtual Machine**

The system installed on the application server and executing the application program.

- **DVM**
- **runtime system**
- **fglrun**

**E**

**environment set**

A named collection of environment variables. Each environment set typically includes a group of related environment variables, where all would be needed to successfully complete a configuration task. Environment sets are defined in the Genero Configuration Management dialog.

**F**

**fallback image**

The image to be displayed if the requested image is not found.

**G**

**Genero Application Server**

An engine that delivers Genero applications for various Genero front-ends in both development and production environments.

- **GAS**

**Genero Archive**

A zip archive that provides installation instructions and the list of application and services to make available.

- **GAR**

**Genero Browser Client**

The client technology that renders the application in a browser.

Details about the client are covered in the *Genero Browser Client User Guide*.

- **GBC**

**Genero Web Client for JavaScript (GWC-JS)**

Tip: GWC-JS was the term for GBC, but is deprecated term since 3.10
Genero Desktop Client
The graphical front-end for a Genero application. The Genero Desktop Client is multi-platform and can run under Windows®, macOS™, and Linux®.

GDC

Genero Ghost Client
A Java framework that allows you to test your business logic and size the infrastructural needs of your applications. The GGC acts as a ghost client because it does not render a graphical user interface. As a result, you can use it to test applications for different front-ends.

GGC
ghost client

Genero Mobile
The solution that enables developers to create Genero applications that run natively on both iOS and Android™ devices, using a single code base from one integrated development environment.

Genero Mobile for Android™
The Android™ client used to display the user interface for your Genero apps during development.

GMA

Genero Mobile for iOS
The iOS client used to display the user interface for your Genero apps during development.

GMI

Genero Report Engine
The runtime component that transforms report documents into report formats.

GRE

Genero Report Viewer
The report viewer included with Genero Desktop Client (GDC). Used when the output is SVG.

GRV

Genero Report Viewer for HTML 5
The report viewer used when the application runs using Genero Browser Client (GBC) or when the output is Browser.

GRV for HTML 5

Genero Web Client
Any of the historical web clients that came before Genero Browser Client (GBC).

Tip: Deprecated since 3.10.
GWC

**Genero Web Services**
A piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service.

GWS

**ghost client**
See Genero Ghost Client on page 1231.

**guilog**
A log file that captures user interaction with an application.

The guilog file is created by starting the application with the `--start-guilog` option and interacting with the front-end. The guilog file can be replayed to mimic the user interaction and reproduce potential issues. A guilog file can also be used as input to create a test scenario for the Genero Ghost Client.

L

**load test**
A test that simulates a specified number of users using your application at the same time and at normal human speed.

Load testing your application during development allows you to see how the application behaves under similar conditions in a production environment. Load testing can help you identify server and network requirements based on the number of anticipated users.

M

**managed project**
A project created with the intention to use BAM.

**meta-schema file**
The central repository of a database's meta-data, containing information about the tables, columns, and relations, and default values of a relational database.

*Note:* Used in Genero Report Writer

**database meta-schema file**

**Meta-schema Manager**
A visual tool used to design, create and maintain database meta-schema files.

*Note:* Used in Genero Report Writer

**modal window**
A window that forces you to interact before you can return to the parent application. An example would be an Open File dialog box.
In Genero, you define a modal window using the windowType:modal style attribute. Programmers typically use predefined window styles with `OPEN WINDOW ... ATTRIBUTES(STYLE="dialog")`.

**modal dialog**

**N**

**named port**
A layout node with a defined section attribute.

**Note:** Used in Genero Report Writer

**Network Address Translation**
A method of allowing computers to access the Internet without assigning real Internet addresses. The connections must originate from the internal machines to reach Internet addresses. The NAT router puts these on the Internet using the router's IP address. When data is returned, it forwards the data to the requesting internal machine. Part of this process includes mapping the internal IP/Port combinations that correspond to external port usage. This allows the router to know where data needs to be sent when it returns. Special port mappings can be made to specific internal IP addresses to support connections originating from the Internet.

Network Address Translation (NAT)

**NAT**

**O**

**orphan property**
An orphan property is a dynamic property present in a Genero Studio design file (such as a form file (.4fd), a project file (.4pw), a BA diagram file (.4ba), and so on) but that is not defined in the current specific setup for Genero Studio. An orphan property is typically the result of a dynamic property that existed in an earlier setup but that has since been removed.

In Genero Studio, you can clear orphan properties by selecting **Tools > Specific Setup > Clean orphan properties.**

**P**

**performance test**
A test that simulates a number of users using your application, increasing the number of users to observe when the system performance starts to degrade.

**pivot table**
A table with fixed roles and types for its columns, suitable for processing and aggregating multi-dimensional data. A pivot table has two types of columns (dimensions and measures) and one type of row (fact row). Data is sorted by the dimensions, and the measures are aggregated.

**Note:** Used in Genero Report Writer

**positioned layout**
See **absolute positioning** on page 1227.
primary port
A layout node without a defined section attribute.

Note: Used in Genero Report Writer

propagate
To generate an extra copy of a container when that container is full. Extra content overflows to the copy.

Note: Used in Genero Report Writer

R

radar chart
See spider web chart on page 1236.

relative positioning
A layout where the elements are positioned relative to the other elements and the size of the report or form. The final size and position of the elements are determined at runtime.

dynamic layout

Report Design Document (.4rp)
A document that defines the design of a report page, including the report data, the report elements for organizing and displaying this data, and the rules how data is merged to produce the result document. Report Design documents have the suffix .4rp and are edited with the graphical report designer or created from a template.

Note: Genero Report Writer

design file

report design file

report design file

Report Designer
A module that provides a graphical editor for editing report design documents.

Note: Genero Report Writer

Designer

report schema file
See data schema file on page 1228.

Tip: This term is sometimes used in the Genero Studio user interface.

Report Template Language
A streaming-capable transformation language that is used to transform XML documents (typically into PXML documents).

Note: Used in Genero Report Writer
**RTL**

**responsive tile list**

A list with data rows (tiles) that reorganize themselves based on the page or browser they are displayed in.

Genero uses presentation styles to implement a responsive tile list. The tiles arrange to maximize use of the page or browser.

![Responsive Paged ScrollGrid](image)

**Figure 465: Scrollgrid as a responsive tile list**

By default, the tiles are displayed linearly.

![Responsive Paged ScrollGrid](image)

**Figure 466: Scrollgrid using the default tile list display**

**Runtime system**

See Dynamic Virtual Machine on page 1230.
sequence diagram
A diagram that visually displays the flow of your application logic. It shows how the application functions call and are called by other functions. Sequence diagrams are displayed in the Code Analyzer.

spider web chart
A type of chart that plots multiple quantitative variables in a radial grid pattern. Each variable is given an axis that starts from a central point. Each axis is arranged radially around that point, with equal distance between them.

radar chart

static layout
See absolute positioning on page 1227.
trigger

A node in the report structure that specifies what happens when data repeats.

Note: Used in Genero Report Writer

unit test

A test that checks each feature of your application in isolation to make sure the feature works as expected. A unit test should provide you with the anticipated responses to a given set of user input, showing that the feature is able to handle both correct and incorrect input.

waterfall chart

A type of chart that plots how an initial value is increased or decreased by a series of intermediate values, leading to a final value. The initial and final values are displayed as entire columns, while the intermediate values are displayed as floating columns.

Web Services Description Language

An XML-based language for describing Web services and how to access them.
WSDL

window
A Genero BDL object created with OPEN WINDOW and destroyed with CLOSE WINDOW. You can create a normal or modal window, depending on the STYLE or the TYPE.

Legal Notices

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software developed by CollabNet (http://www.Collab.Net/).

This product includes software developed by the University of California, Berkeley and its contributors.

This product includes software developed or owned by Caldera International, Inc

Copyrights and Trademarks

The Copyright and trademark page is shared amongst all Four Js documentation. Appearance in this page does not guarantee a mention within the current document.

Copyright notice
© Copyright Four Js Development Tools, Inc. Use, duplication or disclosure restricted.

Trademarks
Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated.

IBM and WebSphere are trademarks or registered trademarks of International Business Machines Corp., in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information.

Linux is a registered trademark of Linus Torvalds.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Other company, product, or service names may be trademarks or service marks of others.
Index

Special Characters

$(delete) 1192
$(DistDir) 1192
$(generateXMLPackage) 1192
$(GMATOOLSDIR) 1192
$(GMITOOLSDIR) 1192
$(PackageFiles) 1192
$(packageId) 1192
$(PackageName) 1192
$generate
finding the XML schema 1167

Numerics

3D charts
   category charts 822
   map charts 822

A

absolute positioning 782
address labels 806
administration
   command line options 125
   generostudio 125
aggregating data
   with transformations 775
Android
   packaging issues 1178
   set Android permissions 1177
Android Logcat 622, 1089
application configuration file
   create 1169
application modeling
   Web services
      create a JSON Webservice server 263
      create a service from a form 256, 262
      create a SOAP client 257
      create a SOAP WebService server 256
      create a standalone JSON Web Service 262
      create a standalone SOAP Web Service 256
      CRUD form and JSON Web Service entity 259
      CRUD form and SOAP Web Service entity 253
      JSON Web Service entity 257
      JSON Web services 257
      JSON Webservice server entity 261
      public fields 257, 263
      SOAP Web Service entity 251
      SOAP Web services 251
      SOAP Webservice server entity 254
      Zoom form and JSON Web Service entity 260
      Zoom form and SOAP Web Service entity 254
application server
   packaging 1169
applications
   debugging 589
   deploying 1168
distributing 1168
   packaging 1168
architecture
   GRW for BDL 672
   Report Writer 649
area charts
   category charts 822
   XY charts 823
Asian fonts 657
AUI tree
   Android device 619, 1086
   Genero Development Client 621, 1088
   iOS device 620, 1087
   view in developer mode 619, 1086

B

backside printing
   in reports 803
BAM
   adding custom code 303
   configuration 142, 231
   environment variables 320
   error messages 325
   file extensions 324
   generating code 225, 1107, 1110
   projects 232
   quick start 1095
   reference 320
   template reference 1121
   templates 1094
BAM desktop
   rename database 282
BAM mobile
   add phone functionality 219
   create form from database 218
   create project 214
   customize the app 221
   generate app 218
   model the database 216
   package and deploy 222
   rename database 281
   run app 218
bar charts
   category charts 822
   map charts 822
bar code
   4-CB 939
   4CB 939
codabar-18 913
codabar-2 912
code-11-matrix 914
code-128 915
code-2-5-datalogic 919
code-2-5-IATA 919
code-2-5-industrial 920
code-2-5-interleaved 920
code-2-5-inverted 920
code-2-5-matrix 920
code-32 921
code-39 922
code-39-extended 927
code-93 931
code-93-extended 933
code-BCD-matrix 920
data-matrix 937
ean-13 938
ean-8 937
ean-code-128 938
ean-data-matrix 938
ean-supplemental-2 938
ean-supplemental-5 938
gs1-13 938, 939
gs1-8 937, 939
gs1-code-128 939
gs1-data-matrix 939
gs1-supplemental-2 939
gs1-supplemental-5 939
Intelligent Mail 939
intelligent-mail 939
list of 911
pdf-417 939
qr-code 941
report objects, properties 902
upc-a 943
upc-e 944
upc-supplemental-2 944
upc-supplemental-5 944
USPS 4-State Customer Barcode 939
USPS OneCode Solution 939
USPS4CB 939
bar codes
in Tool Box 875
basics
architecture 649
first look 16
OrderReport demo 651
Report Writer 647
BDL schema file 349
BLOCK
BAM templates 1117, 1168
bookmarks 447
borders
report objects
color properties 909
examples 786
size expressions 788
style classes 959
style properties 909
width properties 908
table objects
border property 879
bottom border 879
left border 888
right border 890
top border 893
breakpoints
conditional 593
setting 592

Browser
Report Viewer for HTML5 1014
view reports 713
Business Application Diagram
error messages 340
Business Application Modeler
master-detail 302
Business Application Modeling
create a managed project 206
overview 86
quick start 206
business graphs
creating 824
custom keys 829
drawing style 832
in Tool Box 869
outputting as tables 835
types 822
business records
diagram 494, 532
editing 496
error messages 336, 582
properties 495

Captions
adding to report 797
category charts
adding to report structure 833
creating 826
custom keys 829
in Tool Box 870
character mapping 193
child node
in report structure 790
switch with parent 793
client configuration
debugging 615
desktop 170
GBC 171, 176
GDC 170
GMA 177
GMI 177
mobile 177
text mode 179
web 171, 176
Clients
switch clients in Genero Studio 46
Code Analyzer
dependency diagrams 484
overview 86
sequence diagrams 483
code coverage 454
code editor
error messages 479
Code Editor
auto completion 447
basics 445
bookmarks 447
Code Coverage view 454
identifiers 455
Index

overview 89
preferences 459
printing 457
search and replace 448
smart editing 446
splitting document 448
templates 447
vi commands 476
views 466
Code Editor preferences
e extra identifiers 455, 456
command options
   GenerateReport 1002
compare files
diff tool 451
conditional expressions
   in RTL 849
configuration
   BAM 142, 231
   changing the active configuration 141
   distributed mode 1015
   httpdispatch 202
   reference 197
configure Genero Studio
   local environment 163
   remote environment 179
Configure GST
   Update GMA version 150
   Update GMI version 161
Configuring Genero Studio
   mobile client configuration 142
constraints
   meta-schemas 350
creatables.conf
      action attribute 1166
      Category element 1164
      Creatables element 1163
      Directory element 1165
      element listing 1163
      File element 1165
      New element 1165
      Wizard element 1165
custom keys
   business graphs 829

D

data
   choosing methods 670
data model
   add custom code 303
data schema
   adding to report design 796
   creating 774
   generating from BDL 680, 774
data sets 493
data source
   arbitrary XML input source 798
data transformations 775
data values
   adding to report 797
Data view
   in report designer 796
databases
   DB Explorer 392
   servers and users 365
DB Explorer
   changing data 396
   connections 394
   executing queries 399
   limiting rows 397
   opening 393
   showing data 395
   writing queries 398
debug a program
   debug a running local program 590
   debug a running remote program 591
   debug and start a program simultaneously 590
debug version
   create 615, 1083
   run on Android 616, 1084
   run on iOS 616, 1084
debugger
   context menu 598
   executing programs 590
Debugger
   overview 90
debugger command
   backtrace 604
   break 603
   clear 604
   continue 604
   define 605
   delete 605
   disable 606
   enable 607
   file 607
   finish 608
   display 606
   down 607
   enable 607
   file 607
   finish 608
   help 608
   info 608
   list 609
   list of supported commands 603
   next 609
   output 610
   print 610
   quit 610
   run 611
   set 611
   signal 612
   source 611
   step 612
   unsupported commands 614
   until 613
   up 613
   watch 613
   where 614
debugger views
   Backtrace view 603
   Breakpoints view 602
   Command view 599
   Data view 600
Index

1242

listing of views 599
Watchpoints view 601
debugging
applications 589
breakpoints 592
developer mode 617, 1085
graphical debugger 589
menu 109
mobile applications 615, 617, 1083
report design document 858
running programs 590
standalone mobile applications 615, 1083
watchpoints 593
Web component 623, 1090
Web services client application 614
Web services server application 614
decimals
formatting 862
demos
running demos 42
dependency diagrams 484
deploy
file 1189
mobile app 1178
package to GAS 1172
design file, See report designer
designer, See report designer
designing a report, See report designer
developer mode 1080
development environment 1080
device logs 622, 1089
dialogs
Advanced Properties dialog 367
Connection information dialog 370
Database Generation Script dialog 372
Filter View 121
Genero Configuration Management 197
Import Configuration 200
Import Preferences 200
import project 426
list 121
Meta-Schema Manager 367
Print preview 122
Revert schema changes dialog 373
Style Selection 541
Diff
comparing files 451
menu 106
overview 91
preferences 465
SVN 635
Directory node 1180
distribute app
Android app 1186
AppStore 1185
Google Play Store 1186
iOS app 1184
other tools 1187
over-the-air installation 1184
overview 1184
distributed mode
fonts 658
duplex printing
different reverse side 803
start on odd page 805
duplicating data
with sub reports 684
with transformations 776
dynamic layout 782
E
evironment variables
add or edit 168, 418
DBAPP_MOBILE 320
environment sets 164
Genero Studio list 166
GMADIR 167
GMIADIR 167
GMEXTDIR 167
GRE_DEFAULT_IMAGE_URL 1014
GREDATAFILE 1075
GREDEBUG 1013
GREDIR 1013
GRESETUPDIR 168, 321
GSTDIR 167
GSTLIBRARYDIR 167
GSTSCHEMANAMES 167
GSTTERM 168
GSTUSERSAMPLESDIR 168
GSTWCDIR 168
precedence and priority 419
project manager 418
error messages
BAM 325
Business Application Diagram 340
Business Records 336, 582
code editor 479
configuration 202
Form Designer 572, 1218
Genero Mobile 1092
Meta-schema Manager 376
Project Manager 440
report engine 1017
Report Writer 1017
SVN 645
XML validation 586
even page
in reports 805
evenPhysicalPage() 805
Excel
output reports 708
expressions
in properties 847
F
fgldb
debugger 595
file browser
locate file 589
File Browser
overview 92
file extensions
  .4ba 324
  .4dbx 324
  .4fdc 324
  .4fdm 324
  .4fdms 324
  .4fdmsj 324
  .4fdz 324
  .4dzs 324
  .4dzws 324
  .4dzwsj 324
  .4prg 324
  .4pw 324
  .4rd 324
  .4ws 324
  .4wsj 324
  .4wsprg 324
  .4wsp 324
first page
  in reports 803
fonts
  bold 883
  Code Editor 461
  compatibility reports 659
  installing 658
  italic 884
  name 884
  PDF 658
  size 884
  troubleshooting 657
footers
  in reports 792
  in tables 820
foreign keys
  meta-schemas 351
form
  ComboBox 507
  Field 509
Form Designer
  error messages 572, 1218
  overview 93
  usage 523
Form Designer properties
  properties listing 545
Form Designer property
  properties 545
accelerator 545
accelerator2 546
accelerator3 546
accelerator4 546
action 546
aggregate 546
aggregateText 546
aggregateType 547
autoNext 547
autoScale 547
blink 547
buttonTextHidden 547
case 548
century 548
colName 548
color 548
comment 549
completer 549, 549, 560, 563, 564, 566, 566
componentType 550
contextMenu 550
databaseName 550
dataType 550
defaultValue 551
defaultView 551
disclosureIndicator 551
Display Like 551
doubleClick 552
expandedColumn 552
fieldId 552
fieldIdRef 552
fieldType 552
fontPitch 552
format 553
gridChildrenInParent 555
gridHeight 556
gridWidth 556
hidden 556
idColumn 557
image 557
imageCollapsed 557
imageColumn 557
imageExpanded 558
imageLeaf 558
include 558
initializer 558
initialPageSize 558
invisible 559
isNodeColumn 559
items 559
justify 559
keyboardHint 560
left 561
length 561
lookup 561
minHeight 561
minWidth 561
name 562
noEntry 562
notEditable 562
notNull 562
orientation 562
parentIdColumn 562
PER comments 562
picture 562
placeholder 563
posX 563
posY 563
program 563
queryEditable 563
required 564
reverse 564
rowHeight 564
sample 564
scroll 565
scrollbars 565
sizePolicy 565
sliderOrientation 566
spacing 566
splitter 566
sqlTabName 548
Index 1244

step 566
stretch 567
styleFile 567
tabIndex 567
tag 567
text 568
title 568
totalRows 568
underline 568
unHidable 568
unHidableColumns 568
unMovable 568
unMovableColumns 569
unSizable 569
unSizableColumns 569
unSortable 569
unSortableColumns 569
validate 569
Validate Like 569
valueChecked 570
valueMax 570
valueMin 570
valueUnchecked 570
verify 570
wantNoReturns 571
wantTabs 571
width 556, 571
widthUnit 556, 571
windowStyle 572

formField
ComboBox 507
Field 509

forms
business records 493
compiling 493
containers 496
converting per file 493
creating
drag and drop 491
from database 490
item properties 492
overview 486
preview 493
previewing on mobile 617, 1085
quick start 487
web components 515
widgets 504

G

generated applications
action defaults
default 316
modify 318
styles
default 316
modify 318
ToolBar
default 319
modify 320
Topmenu
default 319

modify 319
Genero Archive (GAR) 1171
Genero Mobile for Android (GMA)
  extending, configuration for 145
Genero Mobile for iOS
  configuration 152
Genero Report Engine
  configuration 1013
Genero Report Viewer
  Browser output 1014
  Genero Report Viewer for HTML5 1014
Genero Reporting API
  fgl_report_commitCurrentSettings 721
  fgl_report_configureAutofORMATOutput 728
  fgl_report_configureCompatibilityOutput 729
  fgl_report_configureDistributedEnvironment 730
  fgl_report_configureDistributedProcessing 730
  fgl_report_configureDistributedURLPrefix 731
  fgl_report_configureHTMLDevice 733
  fgl_report_configureImageDevice 731
  fgl_report_configureLabelOutput 732
  fgl_report_configureLocalization 733
  fgl_report_configure MultipageOutput 734
  fgl_report_configureOORTFDevice 735
  fgl_report_configurePageSize 735
  fgl_report_configurePDFDevice 736
  fgl_report_configurePDFFontEmbedding 736
  fgl_report_configureRTFDevice 737
  fgl_report_configureSVGDevice 737
  fgl_report_configureSVGPreview 738
  fgl_report_configureXLSDevice 739
  fgl_report_configureXMLDevice 740
  fgl_report_createProcessLevelDataFile 741
  fgl_report_findResourcePath 741
  fgl_report_getErrorStatus 771
  fgl_report_getErrorString 771
  fgl_report_loadAndCommit 721
  fgl_report_loadCurrentSettings 720
  fgl_report_runFromXML 742
  fgl_report_runReportFromProcessLevelDataFile 742
  fgl_report_selectDrive 743
  fgl_report_selectLogicalPageMapping 744
  fgl_report_selectPreview 745
  fgl_report_setAutoformatType 746
  fgl_report_setBrowserDocumentDirectory 746
  fgl_report_setBrowserDocumentDirectoryURL 746
  fgl_report_setBrowserFontDirectory 747
  fgl_report_setBrowserFontDirectoryURL 747
  fgl_report_setCallbackLocalization 747
  fgl_report_setDistributedRequestingUserName 748
  fgl_report_setEnvironment 744
  fgl_report_setImageShrinkImagesToPageContent 748
  fgl_report_setImageUsePageNamesAsFileNames 748
  fgl_report_setOutputFileName 749
  fgl_report_setPageMargins 749
  fgl_report_setPageSwappingThreshold 750
  fgl_report_setPaperMargins 750
  fgl_report_setPDFImageResolution 751
  fgl_report_setPDFJPEGImageEncoding 751
  fgl_report_setPrinterChromaticity 752
  fgl_report_setPrinterCopies 752
  fgl_report_setPrinterDestinationUrl 752

Index 1244
fgl_report_setPrinterFidelity 753
fgl_report_setPrinterJobImpressions 753
fgl_report_setPrinterJobMediaSheets 754
fgl_report_setPrinterJobName 754
fgl_report_setPrinterJobPriority 754
fgl_report_setPrinterJobSheets 755
fgl_report_setPrinterMediaName 755
fgl_report_setPrinterMediaTray 756
fgl_report_setPrinterName 757
fgl_report_setPrinterNumberUp 757
fgl_report_setPrinterOrientationRequested 757
fgl_report_setPrinterPageRanges 758
fgl_report_setPrinterPJLVariables 758
fgl_report_setPrinterQuality 759
fgl_report_setPrinterRequestingUserName 759
fgl_report_setPrinterSheetCollate 760
fgl_report_setPrinterSides 761
fgl_report_setPrinterSizeName 756
fgl_report_setPrinterWriteToFile 761
fgl_report_setProcessLevelDataFile 762
fgl_report_setRenderingHints 762
fgl_report_setRTFMemoryThreshold 763
fgl_report_setSharePortWithGDC 764
fgl_report_setSVGCopies 764
fgl_report_setSVGOrientationRequested 764
fgl_report_setSVGPageRange 765
fgl_report_setSVGPaperSource 765
fgl_report_setSVGPrinterName 765
fgl_report_setSVGSheetCollate 766
fgl_report_setXLSMergeCells 766
fgl_report_setXLSMmergeCells 767
fgl_report_stopGraphicalCompatibilityMode 767
method list 722

Genero Studio
Tour 22

Genero Workplace Window 124

Geometry
  Alignment 878
  Anchor X 878
  Anchor Y 878
  Scale X 892
  Scale Y 892
  X 895
  X-Size 895
  X-Size Adjustment 896
  Y 896
  Y-Size 896
  Y-Size Adjustment 896

graphical debugger
  running an app 617, 1085

Graphical Debugger 589

GUI testing
  framework 481
  GGC 480
  how to ... 481
  quick start 480
  view macro definitions 482

H

headers
  in reports 792
  in tables 820

HTML
  displaying in report 862
  output reports 708

HTML5
  Report Viewer
    customization 1014

hyperlinks
  in reports 814

I

images
  based on variables 857
  directory 1121
  displaying in report 865
  location 888
  report templates 998

indexes
  meta-schemas 350

Info Node
  layout-dependent reference 811
  Tool Box object 875

initialPageSize 558

internationalization
  encoding 459
  language support 189
  localized strings 716
  mobile applications 1091
  using translate() 858

invoices
  as Genero reports 779
  displaying page numbers 801
  report templates 987

J

JSON
  client 263

K

keyboard shortcut
  buffers and files 475
  clipboard 475
  code completion 476
  Code Editor 472
  code templates 476
  cursor movement 473
  search and replace 475
  selection 473
  text editing 474

keywords
  set color in Code Editor 456
  set font in Code Editor 456
labels
  address 806
  creating in BDL 692
  designing 805
language support
  add text encoding plug-in 191
  character sets 190
  overview 189
last page
  in reports 803
layout-dependent reference
  printing 811
legacy reports
  report design documents 815
Liberation fonts 658
line breaks 811
line charts
  category charts 822
  XY charts 823
List devices
  Android devices 149
  iOS devices 160
  Multiple devices 149, 160
localized strings 716
localizing mobile apps 1091
logs
  program logs 621, 622, 1088, 1089
  viewing the program logs on Android 621, 1088
  viewing the program logs on iOS 622, 1089
  viewing the program logs on the Genero Development Client 622, 1089
Lucida fonts 658
map charts
  adding to report structure 833
  color 830
  creating 824
  custom keys 829
  in Tool Box 869
margins
  PDFs
    crop properties 910
  report objects
    examples 786
    width properties 908
max
  numeric class function 968
  PXML 854
menu
  New 98
  Save 100
  Save All 100
  Save As 100
menus in Genero Studio
  Build menu 108
  Database menu 110
  Debug menu 109
  Dependency Diagram 486
  Edit menu 106
  File menu 96
  Help menu 116
  Projects menu 105
  SCM menu 111
  Search menu 104
  Tools menu 113
  View menu 104
  Windows menu 115
meta-schema
  copy column definition 361
  copy table definition 361
Meta-schema Manager
  overview 94
meta-schemas
  adding to projects 281, 349
  comparing 358
  constraints and indexes 350
  creating 344
  data types 363
  diagram context menu 376
  dialogs 367
  error messages 376
  field information 355
  foreign keys 351
  generate HTML schema documentation 361
  generating database script 359
  opening DB Explorer 395
  overview 343
  preferences 375
  properties 362
  relationships 354
  tables and columns 349
  updating from database 359
  viewing 356
  views 373
metadata
  report design document 796
  report document properties 910
migrating
  BAM template sets 1215
min
  numeric class function 968
  PXML 854
mobile applications
  debugging 615, 1083
  demos 1082
  error messages 1092
  localizing 1091
  packaging 1175
  using Genero Mobile for iOS 152
  using Genero Studio 1080
mobile devices
  packaging 1173
modeling an application
  add and implement a program 235
  add forms 236
  add mobile device features 272
  add relations 270
  add reports 245
  add Web services 250
Business Application (BA) diagram 232
create a BA diagram 235
import files into the diagram 275

new features
version 2.40 1209
version 2.41 1208
version 2.50 1202
version 2.51 1202
version 3.00 1198
version 3.10 12, 1194

numbers
formatting decimals 862
formatting within strings 858
Numeric RTL class 961
positioning in reports 789

odd page
in reports 805
oddPhysicalPage() 805

operators
in RTL expressions 849

options
incognito mode 171

Package node 1180
packaging
application server 1169
directory nodes 1171
Genero Archive (GAR) 1171
mobile app 1175
mobile device 1173
platform 1192
rules 1189

padding
report objects
examples 786
width properties 910
table objects
horizontal padding 885
padding property 889
vertical padding 895

page breaks 811
page numbers
displaying in report 864
displaying invoice page 801
overview 799
setting length 799
using a string 800

paper formats
supported in Genero 978

paper settings
changing with BDL 702
designing reports 807
in Report Writer preferences 982

parent node
in report structure 790
switch with child 793

PDF
configure device 736
display in reports 864
fonts 658
output reports 743

permissions
Android 1180

pie charts
color 830
creating 824

pivot tables
adding dimensions 841
adding measures 842
arranging hierarchies 843
creating 840
data types 837
definition 836
in report structure 844
in Tool Box 872
performance considerations 838
properties 841
runtime configuration 838
sample reports 839

PJL printing 704

POINT
BAM templates 1117, 1168
polar charts 823

pre-printed forms
designing 810

preferences
accelerators 136
Behavior & Display 460
Code Editor 459
Color & Font 461
compilation 137
compiler and runtime 137
database forms 529
DB Explorer 375
form designer 528
form elements 529
general 128
Histories 134
messages 134
Meta-Schema Manager 375
paper settings 982
report writer 982
runtime/debug 137
setting preferences 128
Template 462
Toolbar Configuration view 135
toolbars 135
user actions 132
user interface 135
workspaces 131
XML Schema/DTD 463
preview form 617, 1085
Printer Job Language (PJL) 704
printers
  configuring 656
  driverless 704
  sending report data 703
Profiler 618, 1085
project
  structured 400, 405
project manager
  build rules definition 413
  build rules, add or edit 414
  build rules, usage 415
  command line options for rules 416
  create report 650
  environment variables 418
  execution rules 416
  external dependencies 411
  gsmake 421
  languages 413
  link rules 416
  organizing projects 411
  post-compile commands 420
  post-link commands 421
  pre-compile commands 420
  pre-link commands 421
Project Manager
  locate file 423
  overview 94, 399
properties
  Directory node 1180
  expressions 847
  meta-schema 362
  Package node 1180
Properties view
  in report designer 795
property
  Action (Destination UI Settings) 240
  Destination Field (Filter) 240
  Source Field 296
  Source Field (Filter) 240
  Source Field (Position) 240
Property
  Form Designer
    colorCondition 548
    style 567
PXML expressions
  embedding RTL 855
  overview 854

Q
quick start
  BAM templates 1095
  Form Designer 487
  summary list 48

R
radar charts, See spider web charts
rdd file
  generating 680, 774
reference
  access help 138
  Genero Studio reference 125
  integrate tools 138
  menu configuration 136
  samples directory 126
Reference Box
  layout-dependent reference 811
  Tool Box object 874
regression testing
  framework 481
  GGC 480
  how to ... 481
  quick start 480
  view macro definitions 482
relative positioning 782
rendering hints
  chart style 832
  fgl_report_setRenderingHints 762
  optimizing rendering 715
Report Data wizard 693
report design document, See report designer
report design templates
  .Art 995
  .rsd 992, 993
  Add Fields page 989
  creating from existing reports 1000
  creating new templates 992
  creating reports from 986
  design features 995
  expanding at runtime 992
  filters 998
  filters list 987
  images 998
  modifying 1001
  overview 986
  Schema Association page 988
  schema definition 993
  setting directory 1001
  Variables page 990
report designer
  business graphs 821
  Data view 796
  debugging 858
  design a report 781
  launching 782
  overview 778
  pivot tables 836
  placing elements 782
  Properties view 795
  reference 859
  Report Design window 779
  report output 779
  Report Structure view 790
  set properties 795, 847
  tables 819
  Tool Box view 781
  work area 780
Report Element Properties
  alignment 878
anchorX 878
anchorY 878
author 910
Background Color (Color) 879
Baseline Type (Layout) 878
baselineType 878
bBorder 879
bgColor 879
Bold (Font) 883
border 879
Border (Table) 879
border color properties 786, 909
border style properties 786, 909
border width properties 786, 908
Bottom Border (Table) 879
Categories 876
Categories Title (Chart) 897
Category Key (Items) 897
categoryKey 897
categoryTitle 897
check 903
Check (Bar Code) 903
class 879
Class (Object) 879
clip 880
Clip (Layout) 880
Code Type (Bar Code) 903
Code Value (Bar Code) 903
Code Value Expression (Bar Code) 903
codeType 903
codeValue 903
codeValueExpression 903
color 880, 897
Color (Color) 880, 897
colspan 880
Column Span (Table) 880
comment 880
Compute aggregates on the innermost dimension (Chart) 898
Compute Average (Value) 880
Compute Count (Value) 881
Compute Distinct Count (Value) 881
Compute Maximum (Value) 881
Compute Minimum (Value) 882
Compute Totals (Value) 882
computeAggregatesInnermostDimension 898
computeAverage 880
computeCount 881
computeDistinctCount 881
computeMaximum 881
computeMinimum 882
computeTotal 882
Control Characters (Bar Code) 904
controlCharacters 904
creator 910
Data Symbols per Line (Bar Code) 904
dataSymbolsPerLine 904
Default (Reference) 890
Display Fact Rows (Chart) 898
Display Recurring Dimension Values (Chart) 898
Display Selection (Chart) 898
displayFactRows 898
displayRecurringDimensions 898
displaySelection 898
domainLowerBound 898
domainUpperBound 899
Draw Labels (Chart) 900
drawLegend (Chart) 900
drawAs 899
drawLabels 900
drawLegend 900
encoding 904
Encoding (Bar Code) 904
Enum Values (Value) 882
enumValues 882
Error Correction Degree (Bar Code) 905
errorCorrectionDegree 905
fidelity 883
fill 883
Fill (Image) 883
Fix Width (Table) 884
Floating Behavior (Layout) 883
floatingBehavior 883
fontBold 883
fontItalic 884
fontName 884
fontSize 884
format 884
Format (Page Number) 889
Format (Value) 884
fWidth 884
hAlign 885
Hide Page Footer On Last Page (Mini Page) 885
Hide Page Header On Last Page (Mini Page) 885
Hide Text (Bar Code) 906
hidePageFooterOnLastPage 885
hidePageHeaderOnLastPage 885
Hierarchies input order (Chart) 900
hierarchiesInputOrder 900
Horizontal Alignment (Table) 885
Horizontal Padding (Table) 885
Horizontal Rule (Table) 886
hPadding 885
href 885
href (Hyperlink) 885
hRule 886
id 886
id (Hyperlink) 886
indent 886
Indent (Text) 886
InfoNode Name (Reference) 890
Intended Resolution (Image) 886
intendedResolution 886
isNumeric 887
Italic (Font) 884
key 900
Key (Items) 900
Keys Title (Chart) 900
keysTitle 900
keywords 910
Layout Direction (Orientation) 887
layoutDirection 887
lBorder 888
Left Border (Table) 888
Localize Text (Text) 888
report schema transformations
creating 776
duplication 776
report structure
business graphs 833
element 794
headers and footers 792
pivot tables 843, 844
switch child and parent 793
tree 790
triggers 791
Report Structure view
in report designer 790
report templates, See report design templates
report viewer
customizing 1014
Report Writer
getting started 49
overview 95
reports
creating 650
requirements
iOS 1180
rest
line breaks 811
page breaks 811
PXML 854
REST
JSON Web Service client 263
reverse side printing
in reports 803
ring charts 822
RTF
output reports 713
RTL Class Reference
Alignment class 953
BaselineType class 953
Boolean class 946
BorderStyles class 959
CategoryChartDrawAs class 957
CodeType class 958
Color class 946
Date class 951
Enum class 952
FGLNumericVariable class 960
FGLStringVariable class 960
FloatingBehavior class 956
LayoutDirection class 954
MapChartDrawAs class 957
Numeric class 961
Overview 945
PageNoFormat class 955
Runtime class 972
Section class 956
String class 975
TextAlignment class 953
TrimText class 955
X-SizeAdjustment class 955
XYChartDrawAs class 956
Y-SizeAdjustment class 954
RTL classes 852
RTL expressions
embedding in PXML 855
examples
changing display 856
checking for values 855
displaying images 857
formatting numbers 858
sizing columns 858
language 849
overview 848
S
scatter charts 823
SCH file 349
sequence diagrams 483
settings.agconf
AGSettings element 1123
BasicItem element 1150
BusinessApplication element 1143
BusinessRecord element 1133
Constraint element 1151
Database element 1153
DynamicProperties element 1126, 1136, 1143, 1155
DynamicPropertyGroup element 1132
element listing 1152
File element 1124, 1134, 1153
Form element 1123
Item element 1133, 1142, 1149, 1162
Items element 1149
Messages element 1125, 1135, 1155
Relation element 1150
smart parse
bar codes 907
Source Code Management (SCM)
overview 95
usage 625
Source Code Management module (SCM)
overview 624
reference pages 636
requirements 624
spider web charts 822
SQL queries
BDL 677
DB Explorer 398, 399
static layout 782
step charts 823
strings
displaying in report 861
formatting numbers 858
including page numbers 800
localized 716
String RTL class 975
sub report
coding the application 685
creating the design 691
overview 684
when to use 685
support
reporting an issue 1075
SVG
output reports 708
SVN
access Blame view 634
add files 626
apply patch 632
Blame 634
browse repository 633
checkout files 625
cleanup 631
commit changes 626
copy 631
create patch 632
delete 630
diff 635
integrate bug tracking 635
lock file 629
lock strategies 628
locking 627
log revision range 633
merge and revert 631
move working copy 632
needs-lock property 628
revert changes 629
revert from single revision 631
switch 632
un-add files 629
update 630
view log information 633

T
tables
from business graphs 835
in reports
   adding content 819
   changing width 821
   columns 820, 821
   creating 819
   headers and footers 820
   merging cells 820
   rows 820
   setting triggers 820
in Tool Box 867
meta-schemas 349
pivot tables 836
tcl
   basics and samples 1117
   generating code 1098
   POINT and BLOCK
      adding 1117
tclsh executable 322
template field triggers 995
template fields 996
templates
   BAM
      reference 1121
   report design, See report design templates
time series charts 823
Tool Box objects
   bar codes 875
   business graphs 869
   Business Graphs 822
   Category Chart 870

U
units of measure
   PXML expressions 854
   supported in Genero 978
update
   documentation 124
   software 124
upgrade report design documents
gsreport command-line utility 983
upgrading to a new version
   upgrade to version 2.20 1215
   upgrade to version 2.30 1214
   upgrade to version 2.40 1214
   upgrade to version 2.41 1214
   upgrade to version 2.50 1211
   upgrade to version 3.00 1211
   upgrade to version 3.10 1210
V
variables
expressions in properties 847
predefined node 436
report templates 990
vi editor
commands list 476
view
Bookmarks View 117
central work area 118
Code structure view 466
Database Structure 356
Document Errors view 118, 472
Document view 118
Output View 118, 472
Projects view 119
Properties View 119
Search Results 471
Search/Replace view 467
Structure View 120
Tasks view 120, 471
views
Code Editor 466
Files 588
listing 117
managing views 116
meta-schemas 373
VM output 621, 1088

W
watchpoints 593
waterfall charts 822
web components 515
web configuration
incognito mode 171
Web Services
adding 250, 1076
building application 1079
creating program 1076
overview 96
updating WDSL 1080
Web Services wizard 1076
widget
ComboBox 507
Field 509
widgets
web components 515
Windows
Security Blocking 205
wizard
Report Data 693
Web Services 1076
WSDL
adding web services 1076
updating 1080

X
Xcode
install and configure 152
XML
arbitrary data source 798
catalog files 457
output data file 714
validation error messages 586
XML template files
createables.conf 1121
settings.agconf 1121
xslt
generating code for templates 1118
XY charts
adding to report structure 833
creating 827
in Tool Box 871
Z
zoom form
add zoom 240
pre-select row displayed 296
set relation 240