Genero Report Designer
(Standalone) User Guide
## Contents

### Genero Report Designer User Guide

- Start on an odd or even page ................................................................................................................. 34
- Specify a different reverse side page ..................................................................................................... 33
- Specify different first and last pages ..................................................................................................... 32
- Print totals at the beginning of a report ................................................................................................. 32
- Display totals on a report....................................................................................................................... 31
- Display an invoice page number ............................................................................................................ 31
- Setting the length of a page number ...................................................................................................... 29
- Using a page number string................................................................................................................... 29
- Display group totals and report totals .................................................................................................... 31
- Print totals at the beginning of a report ................................................................................................. 32
- Specify different pages........................................................................................................................... 32
- Specify different first and last pages..................................................................................................... 32
- Specify a different reverse side page................................................................................................... 33
- Start on an odd or even page................................................................................................................ 34
- Design labels........................................................................................................................................... 35
- Example report structure ........................................................................................................................ 24
- Add headers and footers to a report......................................................................................................... 23
- Switch child and parent nodes................................................................................................................ 22
- Modify an object’s borders, margins, or padding .................................................................................... 17
- Align and format numbers....................................................................................................................... 20
- Center elements....................................................................................................................................... 20
- Position elements at the bottom right ..................................................................................................... 21
- Example report structure........................................................................................................................ 24
- Changing a property value (The Properties view)................................................................................ 25
- Report Design Document metadata........................................................................................................ 26
- Adding report data (Data view)............................................................................................................... 27
- Support for arbitrary XML data sources................................................................................................. 28
- Using page numbers............................................................................................................................... 29
- Display an invoice page number............................................................................................................ 31
- Print group totals and report totals ........................................................................................................ 31
- Display totals on a report....................................................................................................................... 31
- Print totals at the beginning of a report ................................................................................................. 32
- Specify different pages........................................................................................................................... 32
- Specify different first and last pages..................................................................................................... 32
- Specify a different reverse side page................................................................................................... 33
- Start on an odd or even page................................................................................................................ 34
- Design labels........................................................................................................................................... 35
- Using page numbers............................................................................................................................... 29
- Setting the length of a page number ...................................................................................................... 29
- Using a page number string................................................................................................................... 29
- Display group totals and report totals .................................................................................................... 31
- Print totals at the beginning of a report ................................................................................................. 32
- Specify different pages........................................................................................................................... 32
- Specify different first and last pages..................................................................................................... 32
- Specify a different reverse side page................................................................................................... 33
- Start on an odd or even page................................................................................................................ 34
- Design labels........................................................................................................................................... 35
- Example report structure ........................................................................................................................ 24
- Add headers and footers to a report......................................................................................................... 23
- Switch child and parent nodes................................................................................................................ 22
- Modify an object’s borders, margins, or padding .................................................................................... 17
- Align and format numbers....................................................................................................................... 20
- Center elements....................................................................................................................................... 20
- Position elements at the bottom right ..................................................................................................... 21

### GRD 3.10 new features

- Display totals on a report....................................................................................................................... 31
- Display an invoice page number............................................................................................................ 31
- Setting the length of a page number ...................................................................................................... 29
- Using a page number string................................................................................................................... 29
- Display group totals and report totals .................................................................................................... 31
- Print totals at the beginning of a report ................................................................................................. 32
- Specify different pages........................................................................................................................... 32
- Specify different first and last pages..................................................................................................... 32
- Specify a different reverse side page................................................................................................... 33
- Start on an odd or even page................................................................................................................ 34
- Design labels........................................................................................................................................... 35
- Example report structure ........................................................................................................................ 24
- Add headers and footers to a report......................................................................................................... 23
- Switch child and parent nodes................................................................................................................ 22
- Modify an object’s borders, margins, or padding .................................................................................... 17
- Align and format numbers....................................................................................................................... 20
- Center elements....................................................................................................................................... 20
- Position elements at the bottom right ..................................................................................................... 21

### Overview of Genero Report Designer

- Genero reports........................................................................................................................................ 9
- The Report Design window....................................................................................................................... 9
- The Report output.................................................................................................................................. 10
- The work area....................................................................................................................................... 11
- The Tool Box view................................................................................................................................. 12
- Print group totals and report totals ........................................................................................................ 31
- Print totals at the beginning of a report ................................................................................................. 32
- Display totals on a report....................................................................................................................... 31
- Display an invoice page number............................................................................................................ 31
- Using a page number string................................................................................................................... 30
- Setting the length of a page number ...................................................................................................... 29
- Using a page number string................................................................................................................... 29
- Display group totals and report totals .................................................................................................... 31
- Print totals at the beginning of a report ................................................................................................. 32
- Specify different pages........................................................................................................................... 32
- Specify different first and last pages..................................................................................................... 32
- Specify a different reverse side page................................................................................................... 33
- Start on an odd or even page................................................................................................................ 34
- Design labels........................................................................................................................................... 35
- Example report structure ........................................................................................................................ 24
- Add headers and footers to a report......................................................................................................... 23
- Switch child and parent nodes................................................................................................................ 22
- Modify an object’s borders, margins, or padding .................................................................................... 17
- Align and format numbers....................................................................................................................... 20
- Center elements....................................................................................................................................... 20
- Position elements at the bottom right ..................................................................................................... 21

### Designing a Report

- Launch the Report Designer................................................................................................................... 13
- Placing elements on the report page........................................................................................................ 13
  - Absolute or relative positioning........................................................................................................ 13
  - Drag multiple objects........................................................................................................................ 15
  - Choose the right container.................................................................................................................. 15
  - Choose other report elements.......................................................................................................... 16
  - Modify the sizing policy of containers............................................................................................. 16
  - Modify an object’s borders, margins, or padding ............................................................................ 17
  - Align and format numbers................................................................................................................ 20
  - Center elements................................................................................................................................ 20
  - Position elements at the bottom right.............................................................................................. 21
- Organizing the report structure (the Report Structure view).................................................................... 21
  - The tree structure............................................................................................................................ 21
  - Triggers............................................................................................................................................... 22
  - Add headers and footers to a report.................................................................................................. 22
  - Switch child and parent nodes.......................................................................................................... 23
  - Example report structure.................................................................................................................. 24
- Changing a property value (The Properties view)................................................................................ 25
- Report Design Document metadata........................................................................................................ 26
- Adding report data (Data view)............................................................................................................... 27
  - Adding data values and captions....................................................................................................... 27
  - Support for arbitrary XML data sources.......................................................................................... 28
- Using page numbers............................................................................................................................... 29
  - Setting the length of a page number ............................................................................................... 29
  - Using a page number string.......................................................................................................... 30
  - Display an invoice page number.................................................................................................... 31
- Print group totals and report totals....................................................................................................... 31
  - Display totals on a report................................................................................................................ 31
  - Print totals at the beginning of a report............................................................................................. 32
- Specify different pages........................................................................................................................... 32
  - Specify different first and last pages................................................................................................. 32
  - Specify a different reverse side page................................................................................................ 33
  - Start on an odd or even page............................................................................................................ 34
- Design labels........................................................................................................................................... 35
Design address labels................................................................................................................................. 36
Specify the paper settings of a report........................................................................................................ 37
Configure the output...................................................................................................................................... 38
Other Report Designer tasks........................................................................................................................ 39
Change the type of a report element........................................................................................................... 39
Design Documents for pre-printed forms...................................................................................................... 39
Force a page or line break............................................................................................................................ 40
Print a Layout-dependent reference (InfoNodes)........................................................................................... 40
Show or hide a report element....................................................................................................................... 43
Use hyperlinks in a report............................................................................................................................. 43

**Working with tables**................................................................................................................................. 44
Add a table to a report................................................................................................................................... 45
Assign content to a table column................................................................................................................ 45
Set the triggers for a table in a report.......................................................................................................... 45
Merge cells.................................................................................................................................................... 46
Add rows or columns................................................................................................................................... 46
Add table headers and footers..................................................................................................................... 46
Change the width of a table.......................................................................................................................... 47
Change the width of a column....................................................................................................................... 47

**Working with business graphs**................................................................................................................... 48
Types of business graph................................................................................................................................ 48
Map charts.................................................................................................................................................... 48
Category charts........................................................................................................................................... 48
XY charts..................................................................................................................................................... 49
Creating a graph.......................................................................................................................................... 50
Create a map chart....................................................................................................................................... 50
Create a category chart............................................................................................................................... 52
Create an XY chart..................................................................................................................................... 53
Custom keys................................................................................................................................................. 55
Coloring a map chart................................................................................................................................... 56
Modify the report structure.......................................................................................................................... 58
Output charts as tables................................................................................................................................. 59

**Working with pivot tables**........................................................................................................................... 60
What are pivot tables?.................................................................................................................................... 61
Data types in the pivot table.......................................................................................................................... 62
Runtime configurability............................................................................................................................... 62
Performance considerations.......................................................................................................................... 63
Sample pivot table reports.......................................................................................................................... 64
Create a pivot table....................................................................................................................................... 64
Pivot table properties..................................................................................................................................... 65
Add a dimension to a pivot table................................................................................................................ 65
Add a measure to a pivot table..................................................................................................................... 66
Arrange your hierarchies............................................................................................................................... 67
Pivot table elements and the Structure view.............................................................................................. 68

**Expressions in properties**............................................................................................................................. 71
RTL expressions overview............................................................................................................................. 72
Using the expression language..................................................................................................................... 74
Using RTL classes.......................................................................................................................................... 76
Using the PXML expression language ................................................................................................. 78
Substituting variables for constants ........................................................................................................... 79
Expression examples ..................................................................................................................................... 80
  Check a field for a value ............................................................................................................................ 80
  Change the display based on a condition .................................................................................................... 81
  Display an image based on a variable ....................................................................................................... 82
  Size a column based on data .................................................................................................................... 82
  Format a number within a string .............................................................................................................. 83

Debugging your Report Design Document ............................................................................................... 83

Create and manage report templates ......................................................................................................... 84
  Create a report from an existing template .................................................................................................. 84
  Template filters ........................................................................................................................................ 85
  Schema Association page .......................................................................................................................... 86
  Add Fields page ..................................................................................................................................... 87
  Variables page ...................................................................................................................................... 88
  Expanding templates at runtime .............................................................................................................. 90
  Create a new report template .................................................................................................................. 90
  Report template schema definition file (.rsd) ........................................................................................... 91
  Report template (.4rt) file design features .............................................................................................. 93
  Customize the appearance of a new report template in the wizard ....................................................... 96
  Create a report template from an existing report .................................................................................... 98
  Modify an existing report template ........................................................................................................ 99
  Set the report template directory .......................................................................................................... 100
  GenerateReport command options ........................................................................................................ 100
  Example 1: Template with placeholders ............................................................................................... 103
    Example 1: Create the files required for a "Users" report template ...................................................... 103
    Example 1: Use the "Users" template to create a report ......................................................................... 105
  Example 2: Template with data groups and repetitions ......................................................................... 106
    Example 2: Create the files required for a "Triggers" report template ............................................... 106
    Example 2: Use the "Triggers" template to create a report ................................................................... 107
  Example 3: Template with a dynamic list ............................................................................................... 109
    Example 3: Create the files required for an "Items" report template ................................................ 109
    Example 3: Use the "Items" template to create a report ....................................................................... 110

Report schema transformations .................................................................................................................. 111
  Options for data operations in reports ..................................................................................................... 112
  Create a report schema transformation ..................................................................................................... 112
  Duplicate data using a report schema transformation ............................................................................. 113

Report Designer Reference ...................................................................................................................... 114
  Report Design Elements (The Toolbox) .................................................................................................. 114
    Simple Containers ................................................................................................................................ 115
    Propagating Containers (Mini Pages) .................................................................................................... 116
    Drawables ............................................................................................................................................. 117
    Business Graphs ................................................................................................................................... 125
    References ............................................................................................................................................ 130
    Bar Code Boxes ................................................................................................................................... 130
  Report Element Properties .................................................................................................................... 131
    General Properties ............................................................................................................................... 132
    Properties related to charts ................................................................................................................. 153
| Contents |

### The Genero Report Designer framework

- Menus and Toolbars: 238
  - The File > New menu: 239
  - Save / Save As / Save All: 239
  - Open from a different version: 239
- Setting Preferences: 240
  - General Preferences: 240
  - User interface preferences: 240
  - Report Writer preferences: 243
- Access Help: 243

### Upgrading GRD

- Update report design documents to the current version: 243
- GRD 3.10 new features: 244
- GRD 3.00 new features: 245
- GRD 2.50 new features: 245

### Genero Glossary

- A: 246
- B: 247
- C: 247
- D: 248
- E: 249
- F: 249
- G: 249
- L: 251
- M: 251
- N: 252
- O: 252
- P: 253
- R: 253
- S: 255
- T: 256
Copyrights and Trademarks................................................................................................. 257
Index........................................................................................................................................259
## Genero Report Designer User Guide

Manual organization at a glance.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Genero reports on page 9</td>
<td>• Launch the Report Designer on page 13</td>
<td>• Change the type of a report element on page 39</td>
<td>• Add a table to a report on page 45</td>
</tr>
<tr>
<td>• The Report Design window on page 9</td>
<td>• Placing elements on the report page on page 13</td>
<td>• Design Documents for pre-printed forms on page 39</td>
<td>• Assign content to a table column on page 45</td>
</tr>
<tr>
<td>• The Report output on page 10</td>
<td>• Organizing the report structure</td>
<td>• Force a page or line break on page 40</td>
<td>• Set the triggers for a table in a report on page 45</td>
</tr>
<tr>
<td>• The work area on page 11</td>
<td>• Changing a property value</td>
<td>• Print a layout-dependent reference</td>
<td>• Merge cells on page 46</td>
</tr>
<tr>
<td>• The Tool Box view on page 12</td>
<td>• Adding report data</td>
<td>• Show or hide a report element on page 43</td>
<td>• Add rows or columns on page 46</td>
</tr>
<tr>
<td>• Debugging your Report Design Document on page 83</td>
<td>• Using page numbers on page 29</td>
<td>• Use hyperlinks in a report on page 43</td>
<td>• Add table headers and footers on page 46</td>
</tr>
<tr>
<td></td>
<td>• Print group totals and report totals on page 31</td>
<td></td>
<td>• Change the width of a table on page 47</td>
</tr>
<tr>
<td></td>
<td>• Specify different pages on page 32</td>
<td></td>
<td>• Change the width of a column on page 47</td>
</tr>
<tr>
<td></td>
<td>• Design labels on page 35</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Specify the paper settings of a report on page 37</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Working with business graphs on page 48</th>
<th>Working with pivot tables on page 60</th>
<th>Expressions in properties on page 71</th>
<th>Create and manage report templates on page 84</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Types of business graph on page 48</td>
<td>• What are pivot tables? on page 61</td>
<td>• RTL expressions overview on page 72</td>
<td>• Create a report from an existing template on page 84</td>
</tr>
<tr>
<td>• Creating a graph on page 50</td>
<td>• Sample pivot table reports on page 64</td>
<td>• Using the expression language on page 74</td>
<td>• Create a new report template on page 90</td>
</tr>
<tr>
<td>• Output charts as tables on page 59</td>
<td>• Create a pivot table on page 64</td>
<td>• Using RTL classes on page 76</td>
<td>• Create a report template from an existing report on page 98</td>
</tr>
<tr>
<td></td>
<td>• Pivot table properties on page 65</td>
<td>• Using the PXML expression language on page 78</td>
<td>• Modify an existing report template on page 99</td>
</tr>
<tr>
<td></td>
<td>• Add a dimension to a pivot table on page 65</td>
<td>• Substituting variables for constants on page 79</td>
<td>• Set the report template directory on page 100</td>
</tr>
<tr>
<td></td>
<td>• Add a measure to a pivot table on page 66</td>
<td>• Expression examples on page 80</td>
<td></td>
</tr>
</tbody>
</table>
### GRD 3.10 new features

Genero Report Designer v 3.10 includes information about new features and changes in existing functionality.

**Table 1: 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 111.</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 121.</td>
</tr>
<tr>
<td>The new Toolbox object, PDFBOX, can embed an entire PDF.</td>
<td>See <a href="#">PDF Box</a> on page 120.</td>
</tr>
<tr>
<td>You can create a Spider Web chart.</td>
<td>See <a href="#">Category charts</a> and <a href="#">drawAs property</a>.</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 <a href="#">Y-size property</a>.</td>
</tr>
<tr>
<td><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 <a href="#">X-Size Adjustment</a> and <a href="#">Y-Size Adjustment</a> properties.</td>
</tr>
</tbody>
</table>

---

### GRD 3.10 new features

- Arrange your hierarchies on page 67
- Pivot table elements and the Structure view on page 68
- GenerateReport command options on page 100

---

<table>
<thead>
<tr>
<th>Report Designer Reference on page 114</th>
<th>The Genero Report Designer framework on page 237</th>
<th>Upgrading GRD on page 243</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Report Design Elements (The Toolbox) on page 114</td>
<td>• Menus and Toolbars on page 238</td>
<td>• Update report design documents to the current version on page 243</td>
</tr>
<tr>
<td>• Report Element Properties on page 131</td>
<td>• Show, dock, or move a view on page 240</td>
<td>• GRD 3.10 new features on page 8</td>
</tr>
<tr>
<td>• Bar Codes on page 167</td>
<td>• Setting Preferences on page 240</td>
<td></td>
</tr>
<tr>
<td>• RTL Classes Overview on page 202</td>
<td>• Access Help on page 243</td>
<td></td>
</tr>
<tr>
<td>• RTL Class Reference on page 201</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Dimensions on page 232</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overview</td>
<td>Reference</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>For business graphs and pivot tables, the new <code>rangeUpperBound</code> and <code>rangeLowerBound</code> properties define the highest and lowest values on the Y-Axis.</td>
<td>See <code>rangeUpperBound</code> and <code>rangeLowerBound</code> properties.</td>
<td></td>
</tr>
<tr>
<td>For XY charts, the new <code>domainUpperBound</code> and <code>domainLowerBound</code> properties define the highest and lowest values on the X-Axis.</td>
<td>See <code>domainUpperBound</code> and <code>domainLowerBound</code> properties.</td>
<td></td>
</tr>
<tr>
<td>Report elements now include a <code>comment</code> property.</td>
<td>See <code>comment property</code>.</td>
<td></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 9
- The Report Design window on page 9
- The Report output on page 10
- The work area on page 11
- The Tool Box view on page 12

### Related concepts

- Designing a Report on page 12
- Create and manage report templates on page 84

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.

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 84.
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 (\*.rp) 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 (\*.rp) and select **File > Report properties > Output Configuration**.

- Choose an output format, such as SVG, PDF, or Image. 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 12
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.

You design a report by initially dragging and dropping containers (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.

Zoom buttons on the Toolbar allow you to zoom in and out on the report design document.
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 15
- Drawables - the report elements contained by a container; see Choose other report elements on page 16
- Business Graphs - the specific charts and items; see Working with business graphs on page 48
- References - to define layout-specific elements; see Choose other report elements on page 16
- Bar Codes - specific bar codes; see Bar Code type details on page 168

These elements can be dragged and dropped into a report design document.

Related concepts
Report Design Elements (The Toolbox) on page 114
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 13
- Placing elements on the report page on page 13
- Organizing the report structure (the Report Structure view) on page 21
- Changing a property value (The Properties view) on page 25
- Adding report data (Data view) on page 26
- Add headers and footers to a report on page 22
- Using page numbers on page 29
- Print group totals and report totals on page 31
- Specify different pages on page 32
- Design labels on page 35
- Specify the paper settings of a report on page 37
• Other Report Designer tasks on page 39
• Some tips for legacy report designers

Related concepts
Overview of Genero Report Designer on page 9
When you launch the Genero Report Designer, you have the tools you need to design Genero reports.

Create and manage report templates on page 84
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 (4rp) 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 13
• Drag multiple objects on page 15
• Choose the right container on page 15
• Choose other report elements on page 16
• Modify the sizing policy of containers on page 16
• Modify an object's borders, margins, or padding on page 17
• Align and format numbers on page 20
• Center elements on page 20
• Position elements at the bottom right on page 21

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 report adapts 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, 2 cm), from the bottom (for example, max – 3 cm), 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:

![Highlighted container](image)

**Figure 4: Highlighted container**

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 13
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 27
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 116
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 115
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 22
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 27.

**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 118, 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 123 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 125 object (chart or pivot table) for the specific type of graph. See *Working with business graphs* on page 48.
- 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 40.

**Modify the sizing policy of containers**

When a 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

![Figure 9: Examples of borders](image-url)
Figure 10: Examples of borders

Related concepts
Properties related to margins and borders on page 164
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:

- \textbf{X-Size} = \text{max}/2
- \text{leftBorderWidth} = 2\text{mm}
- \text{leftMargin} = 1\text{mm}
- \text{rightBorderWidth} = 1.5\text{mm}
- \text{rightPadding} = 3\text{mm}

X-Size is then changed in the following steps:

1. \textbf{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. \textbf{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 (\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 \textbf{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 \textit{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 \textit{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 \textit{alignment} property specifies how the number is aligned, for example, baseline or centered. It controls the \textit{x} position of this report element in its parent container, unless you have set the \textit{x} property explicitly.

**Table 2: 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 30

For Page Number containers, use the \textbf{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 3: Centering elements on page 21.
Table 3: Centering elements

<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>

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 4: Positioning elements at the bottom right on page 21.

Table 4: 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 21
- Triggers on page 22
- Place a trigger within the report structure on page 22
- Add headers and footers to a report on page 22
- Switch child and parent nodes on page 23
- Example report structure on page 24

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.

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 11: Headers and Footers in Structure View on page 23.
Figure 11: Headers and Footers in Structure View

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 12: WordBox is child of Orders_orderid trigger

While holding the ALT key, click on WordBox, then drag and drop it on top of Orders_orderid.

Figure 13: Orders_orderid trigger is child of WordBox

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.

Figure 14: 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 22. 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 15: 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 131
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 71
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 201  
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 16: Report Design metadata](image)

The metadata is inserted into the final document (PDF, SVG) if the format supports metadata.

The "Keywords" property does not currently work for "xlsx" format.

**Related concepts**

**Properties for Report Metadata** on page 167  
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 arbitrary XML data sources, you can generate an `.xsd` file to describe the data schema. See Support for arbitrary XML data sources on page 28.

• 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.

Figure 17: 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 5: Rules governing element creation based on context

<table>
<thead>
<tr>
<th>Element</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITEM (“key” is set if field isn’t numeric, “value” otherwise)</td>
<td>Parent element is a MAPCHART</td>
</tr>
<tr>
<td>CATEGORYITEM (“key” 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.

### 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>
</input>
```

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.

### 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 18: Page Number Box properties

![Page Number Box properties](image)

### 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 PXML string expression value. See Using a page number string on page 30.

4. **Code Value Expression**: Set the PXML 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();`

```
"Page "+format(getPhysicalPageNumber(),ARABIC)+" of 
"+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:

```
"Page "+format(getPageNumber("pageRoot"),ARABIC)+" of 
"+format(getTotalNumberOfPages("pageRoot"),ARABIC)
```

**Related concepts**

Align and format numbers on page 20
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 6: 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.

**Display totals on a report**

In the Report Designer, add group and report totals using the Stripe container.

1. Open the report design (.rap) 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 19: 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 40.

  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 20: Reverse side pages in Structure and Properties View
What to do next

If you use server-side printing, ensure that your printer supports duplex printing. In your source file, ensure that the printer is set up to print on two sides.

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:

   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.

1. Create a new report.
   
2. In the Data View, specify a Data Schema.
   
   See Adding report data (Data view) on page 26.
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.

   ![Figure 21: Label Report and Report Structure](image)

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.
Save your report.

**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 35 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: `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: `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: `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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Orientation</strong></td>
<td>Select <em>Portrait</em> or <em>Landscape</em> as the layout for the report.</td>
</tr>
<tr>
<td><strong>Units</strong></td>
<td>Specify whether to use <em>centimeters</em> or <em>inches</em> when defining the page size and margins for a report.</td>
</tr>
<tr>
<td><strong>Page Size Format</strong></td>
<td>Set the page size using either <em>Standard sizes</em> or <em>Custom sizes</em>. With standard, select the desired standard size from the list provided in the combobox. For custom, specify the height and width of the report.</td>
</tr>
<tr>
<td><strong>Margins</strong></td>
<td>Each page has four margins: top, bottom, left, right.</td>
</tr>
</tbody>
</table>

**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 design document (.4rp).
2. GRW default values.

**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.
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 37.

To select the output format and other options for the current report, select File > Report Properties > Output Configuration.

![Output Configuration dialog](image)

**Figure 23: 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.

### Other Report Designer tasks

These procedures help you complete specific report design tasks.

- **Change the type of a report element** on page 39
- **Design Documents for pre-printed forms** on page 39
- **Force a page or line break** on page 40
- **Print a layout-dependent Reference (InfoNodes)**
- **Show or hide a report element** on page 43
- **Use hyperlinks in a report** on page 43

### 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 114
  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.
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 report 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
- Ctrl-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 13

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 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 *format* method of the *Numeric class* to convert to a string and also specify the format, as in this example:

  ```
  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 25: Report Structure with reference box and info node* on page 42 shows an example report structure (from the OrderReport.4rp report in the Reports demo). *Figure 26: Report showing total in a reference box* on page 43 shows how this report looks when run.
Figure 25: Report Structure with reference box and info node

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.
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:

   ```java
   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></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>SeePaymentTermsForDetails</td>
<td>Name</td>
<td>PaymentTerms</td>
</tr>
<tr>
<td>Type</td>
<td>WordBox</td>
<td>Type</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>href</td>
<td>#paymentTerms</td>
<td>href</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td></td>
<td>id</td>
<td>paymentTerms</td>
</tr>
</tbody>
</table>

**Figure 27: Hyperlink objects**

In the report, when the user clicks the `SeePaymentTermsForDetails` hyperlink, they are sent to the `PaymentTerms` object.

**Working with tables**

A table object displays data in columns and rows.

- Add a table to a report on page 45
- Assign content to a table column on page 45
- Set the triggers for a table in a report on page 45
- Merge cells on page 46
- Add rows or columns on page 46
- Add table headers and footers on page 46
- Change the width of a table on page 47
- Change the width of a column on page 47

**Related concepts**

Table on page 123
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 45.
5. Set the report trigger.
   - See **Set the triggers for a table in a report** on page 45.
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 135
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 233.

**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 48
- **Creating a graph** on page 50
- **Output charts as tables** on page 59

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 28: Pie Chart example on page 48 presents the total revenue (the value) for each customer (the key), based on the OrderReport.rdd file in the Report Writer sample programs.

![West Coast Wholesalers, Inc.](image)

**Figure 28: 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 50

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 29: 3-D Bar Chart example: Revenue by Customer and Categories on page 49 presents total revenue (the value) by the Customer Name and Product Category (the keys).
Figure 29: 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 52
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 30: 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 53
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 31: Map chart object in the work area on page 51, 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 31: Map chart object in the work area on page 51, 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 56.

You must **modify the report structure** to ensure the chart displays as required.

**Related concepts**

Map charts on page 48

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).

![Category chart object in the work area](image)

**Figure 32: Category chart object in the work area**

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 32: Category chart object in the work area** on page 52, the key is the category name. The key must be a String.
Working with business graphs

- **Value** - The data item that contains the numbers to be charted. For example, in *Figure 32: Category chart object in the work area* on page 52, 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 32: Category chart object in the work area* on page 52, 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 48

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 49
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:

```
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><strong>Items</strong></td>
<td></td>
</tr>
<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 34: Properties panel for the chart Items (where key uses trimmed last name)**

![Chart showing Revenue Distribution by Customers (where key uses trimmed last name)](image)

**Figure 35: Chart showing Revenue Distribution by Customers (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 36: Properties panel for the chart Items (where key uses substring)

Figure 37: Revenue Distribution by Customers (where key uses substring)

Related tasks

Create a map chart on page 50
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 52
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:

```
Color.BLUE
```

The map chart displays as in **Figure 38: Chart showing shades of blue** on page 57.

![Pie chart with slices colored in shades of blue](image)

**Figure 38: Chart showing shades of blue**

The first slice, Fred Bloggs, is displayed in blue (RGB 0,0,255). The other slices are colored in compatible shades.

**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:

```
item.fruitname.trim()="Bananas"?Color.YELLOW:item.fruitname.trim()="Apples"?Color.GREEN:
item.fruitname.trim()="Strawberries"?Color.RED:item.fruit.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:

```
Color.fromRGBA(item.color.redValue, item.color.greenValue, item.color.blueValue)
```

The map chart displays as in **Figure 39: Chart showing colored slices** on page 58.
Figure 39: Chart showing colored slices

Related concepts
The Color Class on page 203
The Color class provides methods for specifying the color of an object.

Related tasks
Create a map chart on page 50
Map charts have one key value and one data value. For example, you could map the revenue distribution by customer.

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 40: One chart per each row on page 58. 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 41: One chart, one page** on page 59, the Structure view results in a page containing one chart; the item node is a child of OnEveryRow:

![Figure 41: One chart, one page](image)

**Figure 41: One chart, one page**

With **Figure 42: One chart for each unique userid** on page 59, 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:

![Figure 42: One chart for each unique userid](image)

**Figure 42: One chart for each unique userid**

**Output charts as tables**

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 44
A table object displays data in columns and rows.
Table on page 123
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 61
• Sample pivot table reports on page 64
• Create a pivot table on page 64
• Pivot table properties on page 65
• Add a dimension to a pivot table on page 65
• Add a measure to a pivot table on page 66
• Arrange your hierarchies on page 67
• Pivot table elements and the Structure view on page 68
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 7: 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 128
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 129
A MEASURE element represents both the declarative aspects of the column as well as the data value.

**Working with business graphs** on page 48
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 Order Specifier. 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 descending order on the first column and a ascending 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 128.
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.

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 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 43: Properties of the Pivot Table element

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 65.
5. Add measures under the Fact node. See Add a measure to a pivot table on page 66.
6. Arrange the dimensions and measures in the Structure View. See Arrange your hierarchies on page 67.
7. Set additional properties as needed for the Pivot Table. See Pivot table properties on page 65.
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.

## 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.
Figure 44: Hierarchy Value/Dimension in the Report Structure

You can now edit the Hierarchy Value properties in the Properties View.

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**.
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 HIERACHIES 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.
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 48: 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 49: 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 50: 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).
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 72
- Using the expression language on page 74
  - Operators
  - Conditional Expressions
  - Operands
  - 4GL Variables
  - Examples
- Using RTL classes on page 76
  - Classes
  - Members
  - Examples
- Using the PXML expression language on page 78
• Units of Measure
• Variables
• Functions
• Substituting variables for constants on page 79
• Expression examples on page 80

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.
• 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 25
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 201
Reference information for Report Template Language (RTL) classes.

Dimensions on page 232
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:
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 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.

**Related concepts**

- **Report Element Properties** on page 131
  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 74
  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 76
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 75
- Conditional Expressions on page 75
- Operands on page 76
### Operators

#### Table 8: 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 than 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:

\[
x < 10 \text{? Color.RED}else\null
\]

**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”.

**Examples**

For the purpose of these examples, `order_line` has been replaced with `order`.

1. To add 10% to the item price: `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.

**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 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 variable `order_line.customer_name` has a string 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:

   ```plaintext
   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:

   ```plaintext
   (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 string 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 demo projects.

---

**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 232.

**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 9: 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 most commonly used functions are:

- `max(valueA, valueB)` - this is a function, not the variable listed in Variables on page 78!
- `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 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 \texttt{max} or \texttt{width}) or the unit names (such as \texttt{cm}) are dynamic.

Expression examples

Examples of common expressions used by report designers.

Related concepts

Using a page number string on page 30
For Page Number containers, use the \textbf{Text Expression} property to create a page number string, such as the standard "Page N of M".

Custom keys on page 55
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 \texttt{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:

\begin{verbatim}
orderline.order.contact.trim().length()>0
\end{verbatim}

This expression will evaluate to either \texttt{TRUE} or \texttt{FALSE}.

To explicitly test for an empty string, there are two options:

\begin{verbatim}
orderline.order.contact.trim().length()==0
orderline.order.contact.isEmpty()==TRUE
\end{verbatim}

To explicitly test for null:

\begin{verbatim}
orderline.order.contact.isNull()==TRUE
\end{verbatim}

Use in the Visibility Condition property

Expressions that evaluate to \texttt{TRUE} or \texttt{FALSE} can be used in the \textbf{Visibility Condition} (\texttt{visibilityCondition}) property. If the expression evaluates as \texttt{TRUE}, then the instance of the element will appear in the report. If the expression evaluates as \texttt{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 \textbf{Text} (\texttt{text}) property. The text property specifies the text to be drawn.

\begin{verbatim}
orderline.orders.contact.trim().length()>0?orderline.orders.contact:""
\end{verbatim}

In this expression, the expression evaluates to \texttt{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:

```plaintext
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.

```plaintext
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:

```plaintext
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:

```plaintext
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:

```plaintext
lineitemprice<100?LayoutDirection.UpsideDown:lineitemprice<1000?
LayoutDirection.TurnRight:LayoutDirection.Unturned
```

The report displays as in the following figure.
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:

```
"/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 55: 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.

![Image of Overfull Box message](image.png)

**Figure 56: Overfull Box message**

## 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 9
- **Designing a Report** on page 12

## 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).

1. Select **File > New > Report From Template**.
   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 85. 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 86.
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 combo-box, 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 87.

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 88.

6. Select **Finish**.
   The report is created as an independent .4rp 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 (.4rp) 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 10: List of default filters** on page 85 provides a list of the default filters and a description of each filter.

**Table 10: List of default filters**

<table>
<thead>
<tr>
<th>Filter</th>
<th>Description</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>Filter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------</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>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 96.

**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.

**Figure 57: 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 page](image-url)
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 59: Aggregate fields on page 88.

![Select fields](image)

**Figure 59: 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 60: Variables page** on page 89.

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 100.

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"
       gvsVersion="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 91 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 93 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 11: Available elements in the .rsd document on page 91.

Table 11: 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.</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;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(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>
</tbody>
</table>
### Element in .rsd | Required syntax | Description | In New Report from Template wizard
--- | --- | --- | ---
TemplateFieldsTrigger | groupName="" | 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. | Each uniquely-named `TemplateFieldsTrigger` group becomes an Add Fields page. See also Template field triggers on page 93.

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.

#### 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 61: The template fields trigger in the Report Structure view** on page 93.

![Figure 61: 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 62: New Report from Template dialog box: Add fields** on page 94.
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 63: The Template Field element in the Tool Box view on page 94.

Figure 63: The Template Field element in the Tool Box view
Add template fields to the report template in the same way as other elements. Figure 64: The template field in the Report Structure view on page 95 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.
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 100.

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 85.
Figure 65: 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 100.
- 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 a 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.

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 96.
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 100 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 96.
**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 Report Designer 3.00.22-149130\tpl).

   **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**.

**GenerateReport command options**

The GenerateReport command creates report design files (.4rp) based on a predefined template and schema.

**Table 12: GenerateReport options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-help or -h</td>
<td>Displays a usage text and then exits.</td>
</tr>
<tr>
<td>-schemaFileName</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. 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><strong>Important:</strong></td>
<td>Mandatory option.</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>
<tr>
<td><strong>Important:</strong></td>
<td>Mandatory option.</td>
</tr>
<tr>
<td></td>
<td>The map syntax is as follows:</td>
</tr>
<tr>
<td></td>
<td>map: map-item (',' map-item)*</td>
</tr>
<tr>
<td></td>
<td>map-item: '{' element-name ',' trigger-name '}’</td>
</tr>
<tr>
<td></td>
<td>As an example, consider a report template designed against the schema defined by SimpleListTEmplate.rsd.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Report template schema definition (.rsd) files are located in GREDIR/templates.</td>
</tr>
<tr>
<td></td>
<td>The schema defines the mappable triggers &quot;outerGroups&quot;, &quot;innerGroups&quot; and &quot;rows&quot;, where each is a descendant of its predecessor. If the input schema defines the elements &quot;ProductGroups&quot;, &quot;Areas&quot;, &quot;Orders&quot; and &quot;Items&quot;, then the following are valid maps:</td>
</tr>
<tr>
<td></td>
<td>{ProductGroups,outerGroups},{Areas,innerGroups},{Items,rows}</td>
</tr>
<tr>
<td></td>
<td>{ProductGroups,outerGroups},{Orders,innerGroups},{Items,rows}</td>
</tr>
<tr>
<td></td>
<td>{ProductGroups,outerGroups},{Orders,rows}</td>
</tr>
<tr>
<td></td>
<td>{ProductGroups,rows}</td>
</tr>
<tr>
<td></td>
<td>{Orders,rows}</td>
</tr>
<tr>
<td></td>
<td>The following example mappings are invalid because they violate the ancestry:</td>
</tr>
<tr>
<td></td>
<td>{ProductGroups,innerGroup},{Areas,outerGroups},{Items,rows}</td>
</tr>
<tr>
<td></td>
<td>{Orders,outerGroups},{ProductGroups,rows}</td>
</tr>
</tbody>
</table>
-placeholderMapping

Specifies the mapping between field 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:

```
StringMap:   MapEntry (',', MapEntry)*
MapEntry:    '{' Key ',', Value? '}'
Key:         IdentifierStartChar* IdentifierFollowChar*
Value:       '"' StringToken* '"'
StringToken: [^"\] | EscapedQuote | EscapedBackslash | ExtraEscapes
EscapedQuote: '\\ '"'
EscapedBackslash: '\\ 'n' | '\\ 't' | '\\ 'r'
ExtraEscapes: '\\ 'n' | '\\ 't' | '\\ 'r'
```

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:

```
-placeldnerMapping {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 '.4rp' or '.4rt' 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>Note: Report template schema definition (.rsd) files are located in GREDIR/templates.</td>
<td></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</td>
</tr>
<tr>
<td></td>
<td>outerGroupFields:product_id,product_description</td>
</tr>
<tr>
<td></td>
<td>-fieldNamePatterns</td>
</tr>
<tr>
<td></td>
<td>rowFields:order_details.*</td>
</tr>
<tr>
<td>-outputFileName</td>
<td>Specifies the name of the resulting .4rp file.</td>
</tr>
<tr>
<td>Important: Mandatory option.</td>
<td></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:

• Report template file: Users.4rt.
• Properties file: Users.4rt.prop.
• Image file: Users.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">
  <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 66: Report template example on page 104.

![Figure 66: 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)
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 68: Report template example on page 107.
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:

```
tag: 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 70: Report template example on page 110.
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.

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 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 26
The Data View specifies the structure of the data record for the report.

Options for data operations in reports on page 112
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.

**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.

- 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 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 71: Report Structure with duplicated reports** on page 114 shows a sample structure (from the RevenueAnalysis.4rp report in the Reports.4pw demo).
Figure 71: Report Structure with duplicated reports

Report Designer Reference

Reference information for Report Designer.

- Report Design Elements (The Toolbox) on page 114
- Report Element Properties on page 131
- Bar Codes on page 167
- RTL Class Reference on page 201
- Dimensions on page 232
- Report Writer preferences on page 243

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 118
  • Decimal Format Box on page 118
  • Page Number (PageNoBox) on page 119
  • Image Box on page 121
  • Table on page 123
• Business Graphs
  • Map Chart on page 125
  • Map Chart Item on page 125
  • Category Chart on page 126
  • Category Chart Item on page 126
  • XY Chart on page 127
  • XY Chart Item on page 127
  • Pivot Table on page 127
  • Pivot Table Hierarchy Value on page 128
  • Pivot Table Fact on page 129
  • Pivot Table Measure on page 129
• References
  • Reference Box on page 130
  • Info Node on page 130
• Bar Codes
  • Bar Code Boxes on page 130

Related concepts
The Tool Box view on page 12
The Tool Box view provides report elements to place on a report design document.

Related tasks
Change the type of a report element on page 39
Change a report element from one type to another, for example, from a Word Box to a Word Wrap Box.

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 toptobottom; 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:

```
orderbyline.account.addr1+(10).toChar()+orderbyline.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](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](https://en.wikipedia.org/wiki/Base64)), while other data such as HTML content is typically “Percent encoded” ([Wikipedia link](https://en.wikipedia.org/wiki/Percent_encoding)).

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 72: Embed in document checkbox

**Populating HTML content from text variables**

Enter the code in the Expression Editor for the **Location** property of the HTML Box, including the name of the text variable.

![Edit RTL Expression](image)

Figure 73: 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 29.

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, followed by the base 64 encoded data of the pdf document

To automatically construct this URL, click the ... button for the Location property. Choose the file and select the Embed in document check box:
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()
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 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:

![Image of embed in document checkbox]

**Figure 75: 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:

![Image of expression editor]

**Figure 76: 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).

**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 (4rp) file.

**Related concepts**

Display an image based on a variable on page 82

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 77: Table Object as viewed in the Report Structure](image)

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 44
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 125
  A Map Chart Item defines the data value items for a Map Chart.

- **Map charts** on page 48
  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 50
  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 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 126
- A Category Chart Item defines the data value items for a Category chart.

**Related tasks**

- Create a category chart on page 52
- 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 to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

- categoryKey, 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 127
- XY charts on page 49

Related tasks

- Create an XY chart on page 53

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 as 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 outweighs 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 60

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 60
The pivot table element defines a table element with fixed roles and types for its columns.

Arrange your hierarchies on page 67
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 60
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 on page 60
The pivot table element defines a table element with fixed roles and types for its columns.

Related tasks
Add a measure to a pivot table on page 66
A measure is used to aggregate numeric values in 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) - "MMMMMMMM"

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) on page 40
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 40
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:
The currently supported types are listed in the topic Bar Code type listing on page 167. 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 158
- These are the properties for bar code elements.

**Related reference**

- Bar Code type listing on page 167
- 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 132
- Properties related to margins and borders on page 164
- Properties related to charts on page 153
- Properties related to bar codes on page 158
- Properties for Report Metadata on page 167

**Related concepts**

- Expressions in properties on page 71
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 25
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 133
- anchorX (Anchor X) on page 134
- anchorY (Anchor Y) on page 134
- baselineType (Baseline Type) on page 134
- bBorder (Bottom Border) on page 134
- bgColor (Background Color) on page 134
- border (Border) on page 135
- class (Class) on page 135
- clip (Clip) on page 135
- color (Color) on page 135
- colspan (Column Span) on page 135
- computeAverage (Compute Average) on page 136
- computeCount (Compute Count) on page 136
- computeDistinctCount (Compute Distinct Count) on page 137
- computeMaximum (Compute Maximum) on page 137
- computeMinimum (Compute Minimum) on page 137
- computeTotal (Compute Totals) on page 138
- enumValues (Enum Values) on page 138
- fidelity (Text Fidelity) on page 138
- fill (Fill) on page 138
- floatingBehavior (Floating Behavior) on page 139
- fontBold (Bold) on page 139
- fontItalic (Italic) on page 139
- fontName (Name) on page 139
- fontSize (Size) on page 139
- format (Format) on page 140
- fWidth (Fix Width) on page 140
- hAlign (Horizontal Alignment) on page 140
- hidePageHeaderOnLastPage (Hide Page Header On Last Page) on page 140
- hidePageFooterOnLastPage (Hide Page Footer On Last Page) on page 141
- href (href) on page 141
- hPadding (Horizontal Padding) on page 141
- hRule (Horizontal Rule) on page 141
- id (id) on page 141
- indent (Indent) on page 142
- intendedResolution (Intended Resolution) on page 142
- isNumeric (Numeric Column) on page 142
- layoutDirection (Layout Direction) on page 142
- localizeText (Localize Text) on page 143
- location (Location) on page 144
- name (Name) on page 144
- padding (Padding) on page 144
• pageName (Name) on page 144
• pageNoOffset (Offset) on page 145
• pageNoFormat (Format) on page 145
• pageRanges (Page Ranges) on page 145
• password (Password) on page 145
• pWidth (Proportional Width) on page 145
• rBorder (Right Border) on page 146
• referenceDefault (Default) on page 146
• referenceName (InfoNode Name) on page 146
• rule (Rule) on page 146
• ruleColor (Rule Color) on page 146
• section (Section) on page 146
• scaleX (Scale X) on page 147
• scaleY (Scale Y) on page 147
• splitOversizedItem (Split Oversized Items) on page 148
• strikethrough (Strikethrough) on page 148
• swapX (Swap X) on page 148
• text (Text) on page 148
• textAlignment (Text Alignment) on page 148
• textExpression (Text Expression) on page 148
• tile - replaced by fill (Fill) on page 138
• title (Title) on page 149
• trimText (Trim Text) on page 149
• objectType (Type) on page 150
• underline (Underline) on page 150
• URL (Location) on page 150
• vAlign (Vertical Alignment) on page 150
• value (Value) on page 150
• visibilityCondition (Visibility Condition) on page 151
• vPadding (Vertical Padding) on page 151
• vRule (Vertical Rule) on page 151
• x (X) on page 151
• X-Size (X-Size) on page 151
• X-Size Adjustment (X-Size Adjustment) on page 152
• y (Y) on page 152
• Y-Size (Y-Size) on page 152
• Y-Size Adjustment (Y-Size Adjustment) on page 152

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 = max, anchorX = 1; that is, aligns the most remotely from the origin of x within the parent container
• center - shortcut for x = max/2, 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 13

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) property on page 135 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 27 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 78

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 79: 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 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**

**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 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**
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 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**

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 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**

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
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

Type: Boolean
Default: TRUE. Unlike the other aggregation options, totals are computed by default.

**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.

**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="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 20
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 233.

Type: PXML, point value.
Default value: None.

**Related tasks**
Change the width of a column on page 47
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 22
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 22

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 43

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 144 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 146 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 43
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.

![Figure 80: A top-to-bottom layout direction](image)

In this example, the layout direction is top-to-bottom (as illustrated by the arrow). The box is not closed at the bottom.

**IBorder (Left Border)**

The IBorder 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 IBorder property overrides the more general border (Border) on page 135 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.
**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 121
  An Image Box displays images in GIF, JPEG, PNG, BMP, WBMP, PDF, and SVG formats.

- **HTML Box** on page 118
  An HTML Box displays an image of an HTML document in the report.

- **PDF Box** on page 120
  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 `pageRanges` 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 120
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 120
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 47
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 135 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 141 and `vRule (Vertical Rule)` on page 151 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 22

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 30.

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 167.

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 135 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 142
  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 148
  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 60
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 43
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 144 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 146 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 max. See Modify the sizing policy of containers on page 16.
Tip: The X-Size property can include the oddPhysicalPage() or evenPhysicalPage() functions. See Start on an odd or even page on page 34.
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 Geometry category.

A value of shrinkToChildren shrinks the X-Size as much as possible without clipping any of the children. A value of 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 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: Enum. Choices are shrinkToChildren, expandToParent.

y (Y)
The y property specifies the y-value of a X/Y coordinate pair defined by the element.

In the Properties view, this property is the Y property in the 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: 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 Geometry category.

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 WordWrapBoxes, because the element should typically grow based on its content.

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.

Type: 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.

Tip: If you want a box to have the same height as its parent, set Y-Size to max. See Modify the sizing policy of containers on page 16.

Tip: To ensure that part of the report starts on an odd page, set Y-Size to oddPhysicalPage()?0:max. See Start on an odd or even page on page 34.

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 Geometry category.

A value of shrinkToChildren shrinks the length of Y-Size as much as possible without clipping any of the children. A value of 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 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: Enum. Choices are shrinkToChildren, 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 153
- `categoryTitle (Categories Title)` on page 153
- `color (Color) -- Map Charts` on page 153
- `computeAggregatesInnermostDimension (Compute aggregates on the innermost dimension)` on page 154
- `displayFactRows (Display Fact Rows)` on page 154
- `displayRecurringDimensions (Display Recurring Dimension Values)` on page 154
- `displaySelection (Display Selection)` on page 154
- `drawAs (Draw As)` on page 155
- `drawLabels (Draw Labels)` on page 156
- `drawLegend (Draw Legend)` on page 156
- `hierarchiesInputOrder (Hierarchies input order)` on page 156
- `key (Key)` on page 156
- `keysTitle (Keys Title)` on page 156
- `outputOrder (Output Order)` on page 156
- `seriesTitle (Series Title)` on page 157
- `sortAscending (Sort Ascending)` on page 158
- `sortBy (Sort By)` on page 158
- `topN (Top N)` on page 158
- `valuesTitle (Values Title)` on page 158
- `xAxisTitle (xAxisTitle)` on page 158
- `yAxisTitle (yAxisTitle)` on page 158

Related concepts

- **Working with business graphs** on page 48
  A business graph allows you to represent data visually on the report.

- **Business Graphs** on page 125
  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 155
The domainUpperBound property specifies the highest value on the X-axis in an XY chart.
rangeUpperBound (Range Upper Bound) on page 157
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 157
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 49
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 154
The domainLowerBound property specifies the lowest value on the X-axis in an XY chart.

**rangeUpperBound (Range Upper Bound)** on page 157
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 157
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 49
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 48
A business graph allows you to represent data visually on the report.

**Output charts as tables** on page 59
All chart types now support the drawing of the data as tables via the Draw As property.

**Business Graphs** on page 125
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 157
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 48
A business graph allows you to represent data visually on the report.

**Working with pivot tables** on page 60
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 157
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 48
A business graph allows you to represent data visually on the report.

**Working with pivot tables** on page 60
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 158
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 159
• codeType (Code Type) on page 159
• codeValue (Code Value) on page 159
• codeValueExpression (Code Value Expression) on page 160
• controlCharacters (Control Characters) on page 160
• dataSymbolsPerLine (Data Symbols per Line) on page 160
• encoding (Encoding) on page 160
• errorCorrectionDegree (Error Correction Degree) on page 161
• noCheckDigits (Number Check Digits) on page 162
• noDigits (Number Digits) on page 162
• noText (Hide Text) on page 162
• preferRectangularSymbols (Prefer Rectangular Symbols) on page 162
• rawCodeValue (Raw Code Value) on page 163
• smartParse (Smart Parse) on page 163
• thinToGapRelation (Thin To Gap Relation) on page 163
• thinToThickRelation (Thin To Thick Relation) on page 164

Related concepts
Bar Codes on page 167
The report element container for a Bar Code is a Bar Code Box. This flow object draws bar codes.

Bar Code Boxes on page 130
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 167
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 167
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

**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 196 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 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

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 a 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 196 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 \( \text{GAPWIDTH} = \frac{\text{THINBARWIDTH}}{\text{thinToGapRelation}} \). 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
\[ \text{THICKBARWIDTH} = \frac{\text{THINBARWIDTH}}{\text{thinToThickRelation}} \]

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 17

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 120
  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 (.4rp).

For compatibility reports, which have no .4rp 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 26
  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 13: 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><code>codabar-18</code> on page 170</td>
<td>varies</td>
<td>(calculated width x 20mm h)</td>
</tr>
<tr>
<td><code>codabar-2</code> on page 168</td>
<td>varies</td>
<td>(calculated width x 20mm h)</td>
</tr>
<tr>
<td><code>code-11-matrix</code> on page 171</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td><code>code-128</code> on page 171</td>
<td>varies</td>
<td>(calculated width x 6.5mm h)</td>
</tr>
<tr>
<td><code>code-2-5-datalogic</code> on page 176</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td><code>code-2-5-IATA</code> on page 176</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td><code>code-2-5-industrial</code> on page 176</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td><code>code-2-5-interleaved</code> on page 176</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>Bar Code Type</td>
<td>Number of Digits Supported</td>
<td>Normal size</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>code-2-5-inverted on page 177</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-2-5-matrix on page 177</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-BCD-matrix on page 177</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-32 on page 177</td>
<td>9</td>
<td>(calculated w x h)</td>
</tr>
<tr>
<td>code-39 on page 179</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-39-extended on page 184</td>
<td>varies</td>
<td>(calculated width x 1in h)</td>
</tr>
<tr>
<td>code-93 on page 188</td>
<td>varies</td>
<td>(calculated w x h)</td>
</tr>
<tr>
<td>code-93-extended on page 189</td>
<td>varies</td>
<td>(calculated w X h)</td>
</tr>
<tr>
<td>data-matrix on page 193</td>
<td>varies</td>
<td></td>
</tr>
<tr>
<td>ean-8 on page 194</td>
<td>8</td>
<td>26.73mm x 21.64 mm (w x h)</td>
</tr>
<tr>
<td>ean-13 on page 194</td>
<td>13</td>
<td>37.29mm x 26.26mm (w x h)</td>
</tr>
<tr>
<td>ean-code-128 on page 194</td>
<td>varies</td>
<td>(calculated width x 6.5mm h)</td>
</tr>
<tr>
<td>ean-data-matrix on page 195</td>
<td>varies</td>
<td></td>
</tr>
<tr>
<td>ean-supplemental-2 on page 195</td>
<td>2</td>
<td>6.6mm x 26.26mm (w x h)</td>
</tr>
<tr>
<td>ean-supplemental-5 on page 195</td>
<td>5</td>
<td>15.5mm x 26.26mm (w x h)</td>
</tr>
<tr>
<td>gs1-8 on page 195</td>
<td>8</td>
<td>26.73mm x 21.64 mm (w x h)</td>
</tr>
<tr>
<td>gs1-13 on page 195</td>
<td>13</td>
<td>37.29mm x 26.26mm (w x h)</td>
</tr>
<tr>
<td>gs1-code-128 on page 195</td>
<td>varies</td>
<td>(calculated width x 6.5mm h)</td>
</tr>
<tr>
<td>gs1-data-matrix on page 195</td>
<td>varies</td>
<td></td>
</tr>
<tr>
<td>gs1-supplemental-2 on page 195</td>
<td>2</td>
<td>6.6mm x 26.26mm (w x h)</td>
</tr>
<tr>
<td>gs1-supplemental-5 on page 195</td>
<td>5</td>
<td>15.5mm x 26.26mm (w x h)</td>
</tr>
<tr>
<td>intelligent-mail on page 196</td>
<td>varies, up to 31 digits.</td>
<td></td>
</tr>
<tr>
<td>pdf-417 on page 196</td>
<td>varies</td>
<td>(calculated w x h)</td>
</tr>
<tr>
<td>qr-code on page 197</td>
<td>varies</td>
<td></td>
</tr>
<tr>
<td>upc-a on page 200</td>
<td>12</td>
<td>1.469in x 1.020in (w x h)</td>
</tr>
<tr>
<td>upc-e on page 200</td>
<td>8</td>
<td>0.897in x 1.020in (w x h)</td>
</tr>
<tr>
<td>upc-supplemental-2 on page 201</td>
<td>2</td>
<td>0.26in x 1.02in (w x h)</td>
</tr>
<tr>
<td>upc-supplemental-5 on page 201</td>
<td>5</td>
<td>(0.611in x 1.02in 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:
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:  
$$CS=16-(\text{Sum}(i=1 \text{ to } n \text{ of } \text{Ref}(i))) \mod 16$$  
where Ref(i) is the reference number of the character i, and n is the total number of characters. Example:  
```
value="a37859b"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+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 THINBARWIDTH=0.165mm. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/\text{thinToThickRelation} where \text{thinToThickRelation} should take values between 1/3 and 1/2. Between digits a gap having the width GAPWIDTH=THINBARWIDTH/\text{thinToGapRelation} is drawn. The padding on both sides measures 10\*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’

<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>
<tr>
<td>17</td>
<td>b</td>
</tr>
<tr>
<td>18</td>
<td>c</td>
</tr>
<tr>
<td>19</td>
<td>d</td>
</tr>
</tbody>
</table>

Table 14: Character set for Codabar 2
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 15: 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>
<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>
<tr>
<td>17</td>
<td>b</td>
</tr>
<tr>
<td>18</td>
<td>c</td>
</tr>
<tr>
<td>19</td>
<td>d</td>
</tr>
<tr>
<td>20</td>
<td>t</td>
</tr>
<tr>
<td>21</td>
<td>n</td>
</tr>
<tr>
<td>22</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 '+'

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:

\[ C = \left( \sum_{i=1}^{n} ((i-1) \mod 9 + 1) \cdot \text{Ref}(n-i+1) \right) \mod 11 \]

And the 'K' checksum is calculated using:

\[ K = \left( \sum_{i=1}^{n} ((i-1) \mod 9 + 1) \cdot \text{Ref}(n-i+1) \right) \mod 11 \]

where n specifies the number of characters to the left of the particular check digit and \( \text{Ref}(i) \) specifies the value of the character at position i, starting with the leftmost character having the value 1. For digits \( \text{Ref}(i) \) yields the digit value itself and for the '-' character \( \text{Ref}(i) \) yields the value 10.

Example calculating the 'C' checksum: codeValue="12-12345-67890", noDigits="16", noCheckDigits="2", n=14

\[ C = \left( 1 \cdot 0 + 2 \cdot 9 + 3 \cdot 8 + 4 \cdot 7 + \ldots + 10 \cdot 2 + 1 \cdot 1 + 2 \cdot 10 + 3 \cdot 2 + 4 \cdot 1 \right) \mod 11 = 305 \mod 11 = 8 \]

The 'K' checksum can then be calculated as: \( n=15 \), \( \text{Ref}(15)=C=8 \)

\[ K = \left( 1 \cdot 8 + 2 \cdot 0 + 3 \cdot 9 + 4 \cdot 8 + \ldots + 9 \cdot 4 + 1 \cdot 3 + 2 \cdot 2 + 3 \cdot 1 + 4 \cdot 10 + 5 \cdot 2 + 6 \cdot 1 \right) \mod 11 = 350 \mod 11 = 9 \]

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 between1/3 and1/2. 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 16: Available characters in the character sets A, B, and C on page 171 lists the available characters in the character sets A, B and C.

Table 16: 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>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>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>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>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>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>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>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>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>
<tr>
<td>101</td>
<td>FNC4</td>
<td>CODEA</td>
<td>CODEA</td>
</tr>
<tr>
<td>102</td>
<td>FNC1</td>
<td>FNC1</td>
<td>FNC1</td>
</tr>
<tr>
<td>103</td>
<td>STARTA</td>
<td>STARTA</td>
<td>STARTA</td>
</tr>
<tr>
<td>104</td>
<td>STARTB</td>
<td>STARTB</td>
<td>STARTB</td>
</tr>
<tr>
<td>105</td>
<td>STARTC</td>
<td>STARTC</td>
<td>STARTC</td>
</tr>
<tr>
<td>-</td>
<td>STOP</td>
<td>STOP</td>
<td>STOP</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 194 and gs1-code-128 on page 195 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 17: EAN 128 Bar Code Common AIs**

<table>
<thead>
<tr>
<th>AI</th>
<th>No of Data Characters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>18</td>
<td>Identification of a delivery unit</td>
</tr>
<tr>
<td>01</td>
<td>14</td>
<td>An EAN 13 Number including check digit</td>
</tr>
<tr>
<td>10</td>
<td>up to 20 alphanumeric characters</td>
<td>Shipping batch identifier</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>Production date in the format YYMMDD</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 18: 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: \(CS=(\text{Ref}(1)+\text{Sum}(i=2 \text{ to } n \text{ of } (i-1)\times \text{Ref}(i))) \mod 103\) where \(\text{Ref}(i)\) is the reference number of the character \(i\), and \(n\) is the total number of characters. Example: \(\text{codeValue}=\text{"STARTB,A,B,C"}\) \(CS=(104+(1*33)+(2*34)+(3*35)) \mod 103, CS=310 \mod 103, CS=1\) 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: \(\text{codeValue}=\text{"STARTB,A,B,C,!,STOP"}\)

**Height and width**

The nominal height is 6.5mm. The width of the bar can be calculated using this formula: \(L/mm=(5.5Nc+11Nab+35)*0.19\) 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 \( \text{THINBARWIDTH} = 0.0236 \text{in} \). The width of a thick bar is \( \text{THICKBARWIDTH} = \text{THINBARWIDTH} \div \text{thinToThickRelation} \) where \( \text{thinToThickRelation} \) 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 \( \text{THINBARWIDTH} = 0.0236 \text{in} \). The width of a thick bar is \( \text{THICKBARWIDTH} = \text{THINBARWIDTH} \div \text{thinToThickRelation} \) where \( \text{thinToThickRelation} \) should take values between 1/3 and 1/2. Between digits a gap of width \( \text{GAPWIDTH} = \text{THINBARWIDTH} \div \text{thinToGapRelation} \) 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 \( \text{THINBARWIDTH} = 0.0236 \text{in} \). The width of a thick bar is \( \text{THICKBARWIDTH} = \text{THINBARWIDTH} \div \text{thinToThickRelation} \) where \( \text{thinToThickRelation} \) should take values between 1/3 and 1/2. Between digits a gap of width \( \text{GAPWIDTH} = \text{THINBARWIDTH} \div \text{thinToGapRelation} \) on page 163 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 \( \text{THINBARWIDTH} = 0.0236 \text{in} \). The width of a thick bar is \( \text{THICKBARWIDTH} = \text{THINBARWIDTH} \div \text{thinToThickRelation} \) where \( \text{thinToThickRelation} \) should take values between 1/3 and 1/2. Between digits a gap of width \( \text{GAPWIDTH} = \text{THINBARWIDTH} \div \text{thinToGapRelation} \) 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 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-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=Sum(i=1 to 8 of CT((i-1)%2*2*Ref(i))) mod 10 where Ref(i) denotes the value of the digit at position i (the leftmost digit has the index 1) and CT() denotes the cross total of its argument (e.g. 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 19: 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>Reference Number</td>
<td>Character</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</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>
<tr>
<td>24</td>
<td>S</td>
</tr>
<tr>
<td>25</td>
<td>T</td>
</tr>
<tr>
<td>26</td>
<td>U</td>
</tr>
<tr>
<td>27</td>
<td>V</td>
</tr>
<tr>
<td>28</td>
<td>W</td>
</tr>
<tr>
<td>29</td>
<td>X</td>
</tr>
<tr>
<td>30</td>
<td>Y</td>
</tr>
<tr>
<td>31</td>
<td>Z</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 1 in. 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=0.0197 in. 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 20: 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>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>
</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: \( \text{codeValue} = "DATALOGIC" \Rightarrow CS = (13 + 10 + 29 + 10 + 21 + 24 + 16 + 18 + 12) \mod 43, CS = 153 \mod 43, CS = 24 \) Looking up reference number 24 yields the character 'O'. The full code value including checksum and stop character is therefore: \( \text{codeValue} = "DATALOGICO" \)

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 21: 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>ASCII code</td>
<td>Code 39 sequence</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</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>/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>ASCII code</td>
<td>Code 39 sequence</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</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>V</td>
<td>V</td>
</tr>
<tr>
<td>ASCII code</td>
<td>Code 39 sequence</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------</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>
<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>
</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 22: Code 39 Extended names and the textual representation that is used in printout on page 184 lists the names and the textual representation that is used in printout.

Table 22: 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>{</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>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>ASCII code</td>
<td>Textual representation</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</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>!</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>%</td>
<td>%</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</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>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>ASCII code</td>
<td>Textual representation</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</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>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>ASCII code</td>
<td>Textual representation</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</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>`</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>
<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>
</tbody>
</table>
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 23: 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>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>
</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:

\[ C = (\text{Sum}(i=1 \text{ to } n \text{ of } ((i-1 \mod 20)+1) \times \text{Ref}(n-i+1))) \mod 47 \]

and the 'K' checksum is calculated as follows:

\[ K = (\text{Sum}(i=1 \text{ to } n \text{ of } ((i-1 \mod 15)+1) \times \text{Ref}(n-i+1))) \mod 47 \]

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*12+2*18+3*16+4*24+....7*29+8*10+9*13) \mod 47=757 \mod 47=5` The K checksum can then be calculated as: `n=10, Ref(10)=C=5K=(1*5+2*12+3*18+4*16+....8*29+9*10+10*13) \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.
The nominal height is 1in. The nominal width of a bar is BARWIDTH=0.022in. The width of the entire bar is (1+(noDigits+2)*9)*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 24: 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>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>ASCII code</td>
<td>Code 93 sequence</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</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>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>ASCII code</td>
<td>Code 93 sequence</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</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>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>ASCII code</td>
<td>Code 93 sequence</td>
</tr>
<tr>
<td>------------</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>z</td>
<td></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>

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 195 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 195 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 171 bar code.

**Note:** The ean-code-128 and gs1-code-128 on page 195 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 195 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 195 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 195 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 194.

gs1-13
A synonym of the ean-13 bar code type.
See ean-13 on page 194.

gs1-code-128
A synonym for the ean-code-128 bar code type.
See ean-code-128 on page 194.

gs1-data-matrix
A synonym for the ean-data-matrix bar code type.
See ean-data-matrix on page 195.

gs1-supplemental-2
A synonym of the ean-supplemental-2 bar code type.
See ean-supplemental-2 on page 195.

gs1-supplemental-5
A synonym of the ean-supplemental-5 bar code type.
See ean-supplemental-5 on page 195.
**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 25: 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 160:
  - 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 163:

- 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 161:

- 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 160:

- 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 (&quot;&quot;) 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>
</tbody>
</table>
## Encoding Type

<table>
<thead>
<tr>
<th>Encoding Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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

**Figure 81: Hello World: size not specified (default error correction (3))**

![QR Code Example 1](image1)

**Figure 82: Hello World: size not specified (error correction degree (2))**

![QR Code Example 2](image2)

**Figure 83: Hello World: size not specified (error correction degree (1))**

![QR Code Example 3](image3)
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.

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 26: 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>sNNNNNN0c</td>
<td>00000</td>
<td>3</td>
<td>sNN000000NNNc</td>
</tr>
<tr>
<td>sNNNNNN1c</td>
<td>10000</td>
<td>3</td>
<td>sNN100000NNNc</td>
</tr>
<tr>
<td>sNNNNNN2c</td>
<td>20000</td>
<td>3</td>
<td>sNN200000NNNc</td>
</tr>
<tr>
<td>UPC-E Number</td>
<td>Digits to insert</td>
<td>Position of Insertion</td>
<td>Resulting UPC-A Number</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
<td>-----------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>sNNNNNN3c</td>
<td>00000</td>
<td>4</td>
<td>sNNN00000NNc</td>
</tr>
<tr>
<td>sNNNNNN4c</td>
<td>00000</td>
<td>5</td>
<td>sNNNN00000Nc</td>
</tr>
<tr>
<td>sNNNNNN5c</td>
<td>00005</td>
<td>6</td>
<td>sNNNN0000005c</td>
</tr>
<tr>
<td>sNNNNNN6c</td>
<td>00006</td>
<td>6</td>
<td>sNNNN0000006c</td>
</tr>
<tr>
<td>sNNNNNN7c</td>
<td>00007</td>
<td>6</td>
<td>sNNNN0000007c</td>
</tr>
<tr>
<td>sNNNNNN8c</td>
<td>00008</td>
<td>6</td>
<td>sNNNN0000008c</td>
</tr>
<tr>
<td>sNNNNNN9c</td>
<td>00009</td>
<td>6</td>
<td>sNNNN0000009c</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 202
- The Color Class on page 203
- The Date Class on page 207
- The Enum Classes on page 208
- The Numeric Class on page 215
- The Runtime Class on page 227
- The String Class on page 230
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 27: 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>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 76
The Report Template Language (RTL) expressions are typed; that is, the expression must return a particular class.

Expressions in properties on page 71
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 28: 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 71
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 29: 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>
</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:

```
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 30: 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 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 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.</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 alpha value.</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 31: 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 Color object.</td>
</tr>
<tr>
<td></td>
<td>Creates a brighter version of the Color object by applying an arbitrary scale factor to each 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 Color object.</td>
</tr>
<tr>
<td></td>
<td>Creates a darker version of the Color object by applying an arbitrary scale factor to each 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 color 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 Color object.</td>
</tr>
<tr>
<td>toString()</td>
<td>Returns a string representation in the form &quot;#argb&quot;.</td>
</tr>
</tbody>
</table>

### Example - Color Object Method

```
Color.RED.darker()
```

**Related concepts**

[Expressions in properties](#) on page 71
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:

```java
Color.fromHSBA(h Numeric, s Numeric, b Numeric)
```

To return a Color object based on hue, saturation, brightness, and alpha:

```java
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**

```java
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 206
  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:

```java
fromRGBA(r Numeric, g Numeric, b Numeric)
```

To return an sRGB Color object based on red, green, blue, and alpha values:

```java
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)
\[ a \text{ (alpha) is within the range (0.0 - 1.0)} \]

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**

fromHSBA() on page 206

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 32: 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 in iso format (&quot;YYYY-MM-DD&quot;). Using the function guarantees that Date values are constructed correctly without the danger of runtime parse errors due to changing 4GL date formats or locale settings.</td>
</tr>
<tr>
<td>parseString(String value, String format)</td>
<td>Constructs a Date from value. The format is used for parsing. 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 33: 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</td>
</tr>
<tr>
<td>ddd</td>
<td>Three-letter English-language</td>
</tr>
<tr>
<td>mm</td>
<td>Month as a two-digit integer.</td>
</tr>
<tr>
<td>mmm</td>
<td>Three-letter English-language</td>
</tr>
<tr>
<td>yy</td>
<td>Year, as a two-digit integer</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":

```java
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:

```java
Date.fromIsoValue(orderline.orders.orderdate.isoValue).format("DDD DD MMM YYYY")
```

**Related concepts**

Expressions in properties on page 71

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 209
- The TextAlignment Class on page 209
- The BaselineType Class on page 209
- The LayoutDirection Class on page 210
- The Y-SizeAdjustment Class on page 210
- The PageNoFormat Class on page 211
- The TrimText Class on page 211
- The X-SizeAdjustment Class on page 211
- FloatingBehavior Class on page 212
- Section Class on page 212
- XYChartDrawAs Class on page 212
- MapChartDrawAs Class on page 213
- CategoryChartDrawAs Class on page 213
- CodeType Class on page 214
- BorderStyles Classes on page 215
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 34: 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 35: 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 36: 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

```
LayoutDirection
```

The class consists of a set of public static member variables that represent the valid values for the `layoutDirection` property.

Table 37: Member Objects (Static Member Variables)

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rightleft</td>
</tr>
<tr>
<td>RighttoRight</td>
</tr>
</tbody>
</table>

The Y-SizeAdjustment Class
Public static member variables that represent the valid values for the Y-SizeAdjustment property.

Syntax

```
Y-SizeAdjustment
```

The class consists of a set of public static member variables that represent valid values for the `Y-SizeAdjustment` property.

Table 38: 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>
**The PageNoFormat Class**
Public static member variables that represent the valid values for the pageNoFormat property.

**Syntax**

```java
PageNoFormat
```

The class consists of a set of public static member variables that represent the valid values for the `pageNoFormat` property.

**Table 39: 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**

```java
TrimText
```

The class consists of a set of public static member variables that represent the valid values for the `trimText` property.

**Table 40: 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**

```java
X-SizeAdjustment
```

The class consists of a set of public static member variables that represent valid values for the `X-SizeAdjustment` property.

**Table 41: 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**

```java
FloatingBehavior
```

The class consists of a set of public static member variables that represent the valid values for the property.

**Table 42: 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**

```java
Section
```

The class consists of a set of public static member variables that represent the valid values for the property.

**Table 43: 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**

```java
XYChartDrawAs
```

The class consists of a set of public static member variables that represent the valid values for the property.
Table 44: 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**

```mapchartdrawas```

The class consists of a set of public static member variables that represent the valid values for the property.

Table 45: 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**

```categorychartdrawas```

The class consists of a set of public static member variables that represent the valid values for the property.

Table 46: 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>
**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 47: Member Objects (Static Member Variables)**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line3D</td>
</tr>
<tr>
<td>SpiderWeb</td>
</tr>
<tr>
<td>StackedArea</td>
</tr>
<tr>
<td>StackedBar</td>
</tr>
<tr>
<td>Waterfall</td>
</tr>
<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>
BorderStyles Classes
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 48: 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 165
The border style properties control the style of the borders for the report object.

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 49: 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>cbart()</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>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>isInfinite()</td>
<td>Returns TRUE if the value has infinite 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>rint()</td>
<td>Returns the Numeric value that is closest in value to the value and is equal to a mathematical integer.</td>
</tr>
<tr>
<td>round()</td>
<td>Returns the closest integer to the value.</td>
</tr>
<tr>
<td>signum()</td>
<td>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.</td>
</tr>
<tr>
<td>sinh()</td>
<td>Returns the hyperbolic sine of a numeric 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>sqrt()</td>
<td>Returns the correctly rounded positive square root of a double 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, 65.toChar() 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>
<tr>
<td>toString()</td>
<td>Converts the value to a string representation.</td>
</tr>
<tr>
<td>getPhysicalPageNumber()</td>
<td>Gets the current page number of the physical page.</td>
</tr>
<tr>
<td>getTotalNumberOfPhysicalPages()</td>
<td>Gets the total number of physical pages.</td>
</tr>
<tr>
<td>getPageNumber(String pageName)</td>
<td>Gets the page number of the specified page.</td>
</tr>
<tr>
<td>getTotalNumberOfPages(String pageName)</td>
<td>Gets the total number of pages for the specified page.</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 71

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**

```plaintext
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**

```plaintext
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, cbrt(-x) == -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 226
Returns the correctly rounded positive square root of a double value.

ceil()
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

ceil()  

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 Numeric.ceil(x) is exactly the value of -Numeric.floor(-x).

Related concepts
floor() on page 221
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.

**cos()**

Returns the trigonometric cosine of an angle.

**Syntax**

```java
    cos()
```

**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 225
  Returns the trigonometric sine of an angle.
- **tan()** on page 226
  Returns the trigonometric tangent of an angle.

**cosh()**

Returns the hyperbolic cosine of a double value.

**Syntax**

```java
    cosh()
```

**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 225
  Returns the hyperbolic sine of a double value.
- **tanh()** on page 226
  Returns the hyperbolic tangent of a double value.

**exp()**

Returns Euler's number \( e \) raised to the power of a double value.

**Syntax**

```java
    exp()
```

**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 219
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 50: 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 223
Returns the base 10 logarithm of a double value.

log10()
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 222
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 223
Returns the smaller of two int values.

round() on page 224
Returns the closest int to the value.

min()
Returns the smaller of two int values.

Syntax

min(Numeric b)
**Parameters**

- *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 223
  Returns the greater of two int values.

- `round()` on page 224
  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:

```java
(int)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 223
  Returns the greater of two int values.

- `min()` on page 223
Returns the smaller of two int values.

**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.

**Syntax**

```
signum()
```

**Usage**

Special cases:
- If the value is NaN, then the result is NaN.
- If the value is positive zero or negative zero, then the result is the same as the value.

**sin()**

Returns the trigonometric sine of an angle.

**Syntax**

```
sin()
```

**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 220
  Returns the trigonometric cosine of an angle.

- **tan()** on page 226
  Returns the trigonometric tangent of an angle.

**sinh()**

Returns the hyperbolic sine of a double value.

**Syntax**

```
sinh()
```

**Usage**

The hyperbolic sine of x is defined to be (ex - 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 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 2.5 ulps of the exact result.
**Related concepts**

cosh() on page 220  
Returns the hyperbolic cosine of a double value.

tanh() on page 226  
Returns the hyperbolic tangent of a double value.

**sqrt()**  
Returns the correctly rounded positive square root of a double value.

**Syntax**

```java
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 219  
Returns the cube root of a double value.

tan()  
Returns the trigonometric tangent of an angle.

**Syntax**

```java
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 220  
Returns the trigonometric cosine of an angle.

sin() on page 225  
Returns the trigonometric sine of an angle.

tanh()  
Returns the hyperbolic tangent of a double value.

**Syntax**

```java
tanh()
```
**Usage**

The hyperbolic tangent of $x$ is defined to be $(e^x - e^{-x})/(e^x + e^{-x})$, in other words, $\text{sinh}(x)/\text{cosh}(x)$.

**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 220
  Returns the hyperbolic cosine of a double value.
- **sinh()** on page 225
  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**

```
Runtime
```

**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.</td>
</tr>
<tr>
<td>getPrinterMediaName()</td>
<td>Returns a STRING indicating the media name specified.</td>
</tr>
<tr>
<td>getPrinterMediaSizeName()</td>
<td>Returns a STRING indicating the media size name specified.</td>
</tr>
<tr>
<td>getPrinterMediaTray()</td>
<td>Returns a STRING indicating the media tray name specified.</td>
</tr>
<tr>
<td>getPrinterName()</td>
<td>Returns a STRING indicating the printer name specified.</td>
</tr>
<tr>
<td>getSVGPaperSource()</td>
<td>Returns a STRING indicating the paper source.</td>
</tr>
<tr>
<td>getSVGPrinterName()</td>
<td>Returns a STRING indicating the printer name.</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 &quot;Browser&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingExcelOutput()</td>
<td>BOOLEAN; returns TRUE if the device name &quot;XLS&quot; or &quot;XLSX&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingHTMLOutput()</td>
<td>BOOLEAN; returns TRUE if the device name &quot;HTML&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingImageOutput()</td>
<td>BOOLEAN; returns TRUE if the device name &quot;Image&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingOORTFOutput()</td>
<td>BOOLEAN; returns TRUE if the device name &quot;OORTF&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingPDFOutput()</td>
<td>BOOLEAN; returns TRUE if the device name &quot;PDF&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingPostscriptOutput()</td>
<td>BOOLEAN; returns TRUE if the device name &quot;Postscript&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingRTFOutput()</td>
<td>BOOLEAN; returns TRUE if the device name &quot;RTF&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingSVGOutput()</td>
<td>BOOLEAN; returns TRUE if the device name &quot;SVG&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingXLSOutput()</td>
<td>BOOLEAN; returns TRUE if the device name &quot;XLS&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingXLSXOutput()</td>
<td>BOOLEAN; returns TRUE if the device name &quot;XLSX&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingPrinterOutput()</td>
<td>BOOLEAN; returns TRUE if the device name &quot;Printer&quot;; otherwise FALSE.</td>
</tr>
<tr>
<td>producingForPreview()</td>
<td>BOOLEAN; returns TRUE if preview; 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 71
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.

Syntax

```
getOutputDeviceName()
```

Possible values

• PDF
• XLS
• XLSX
• HTML
• Image
• Printer
• Postscript
• SVG
• Browser
• RTF
• OORTF

getSVGPaperSource()
Runtime class method that returns a STRING indicating the paper source.

Syntax

```
getSVGPaperSource()
```

Possible values

• Auto
• Cassette
• Envelope
The String Class
The String class provides methods for working with text.

- Syntax
- Methods
- Usage and Examples

Syntax

```cpp
String
```
### Methods

#### Table 52: Object Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>charAt(index Numeric)</code> RETURNING result String</td>
<td>Returns a String containing the character at the specified <code>index</code> in the current String.</td>
</tr>
<tr>
<td><code>contains(s String)</code> RETURNING result Boolean</td>
<td>Returns a Boolean value (TRUE/FALSE) specifying whether <code>s</code> is contained within the current String.</td>
</tr>
<tr>
<td><code>endsWith(s String)</code> RETURNING result Boolean</td>
<td>Returns a Boolean value (TRUE/FALSE) specifying whether the current String ends with <code>s</code>.</td>
</tr>
<tr>
<td><code>equals(s String)</code> RETURNING result Boolean</td>
<td>Returns a Boolean value (TRUE/FALSE) specifying whether <code>s</code> matches the current String. If one of the Strings is null the method returns FALSE.</td>
</tr>
<tr>
<td><code>equalsIgnoreCase(s String)</code> RETURNING result Boolean</td>
<td>Returns a Boolean value (TRUE/FALSE) specifying whether <code>s</code> 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> RETURNING result String</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> RETURNING result Numeric</td>
<td>Returns a Numeric value representing the index of <code>s</code> within the current String.</td>
</tr>
<tr>
<td><code>indexOf(s String , index Numeric)</code> RETURNING result Numeric</td>
<td>Returns a Numeric value representing the index of <code>s</code> within the current String, starting from byte position <code>index</code>. Returns zero if the substring was not found. Returns -1 if the current String is null.</td>
</tr>
<tr>
<td><code>isEmpty()</code> RETURNING result Boolean</td>
<td>Returns a Boolean value. Returns true if the current string has a length of zero (<code>length() = 0</code>), otherwise false.</td>
</tr>
<tr>
<td><code>isNull()</code> RETURNING result Boolean</td>
<td>Returns a Boolean value. Returns true if the current string has a length of zero (<code>length() = 0</code>) 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> RETURNING result Numeric</td>
<td>Returns a Numeric value representing the index within the current String of the last occurrence of <code>s</code>, searching backward.</td>
</tr>
<tr>
<td><code>lastIndexOf(s String , index Numeric)</code> RETURNING result Numeric</td>
<td>Returns a Numeric value representing the index within the current String of the last occurrence of <code>s</code>, searching backward. starting at the specified position <code>index</code>.</td>
</tr>
<tr>
<td><code>length()</code> RETURNING result Numeric</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 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 71
  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 71
  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 78
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 53: 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</td>
<td>pt</td>
<td>Point value (72.27point = 1in)</td>
<td>1</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</td>
<td>pc</td>
<td>Pica value (12point = 1pica)</td>
<td>12</td>
</tr>
<tr>
<td>inch</td>
<td>in</td>
<td>Inch value (72.27point = 1in)</td>
<td>72.27</td>
</tr>
<tr>
<td>bigpoint</td>
<td>bp</td>
<td>Big point (72bigpoint = 1in)</td>
<td>72.27/72</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</td>
<td>dd</td>
<td>Didot point (1157dd = 1238pt)</td>
<td>1238/1157</td>
</tr>
<tr>
<td>cicero</td>
<td>cc</td>
<td>Cicero value (1cc = 12dd)</td>
<td>12*1238/1157</td>
</tr>
</tbody>
</table>

**Paper Format Abbreviations**
A list of paper format abbreviations supported in Genero Studio, for example, ISO 4A0 and Letter.

**Table 54: 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>(dinlang)/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>(folio</td>
<td>germanlegalfanfold)</td>
<td>Folio/German legal fanfold</td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(invoice</td>
<td>statement)(width</td>
<td>length)</td>
<td>Invoice/Statement</td>
</tr>
<tr>
<td>(ledger</td>
<td>tabloid</td>
<td>11x7</td>
<td>Ledger/Tabloid</td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(naletter</td>
<td>letter</td>
<td>note)</td>
<td>Letter</td>
</tr>
<tr>
<td>(width</td>
<td>length)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(nalegal</td>
<td>legal)(width</td>
<td>length)</td>
<td>Legal</td>
</tr>
<tr>
<td>(quarto(width</td>
<td>length)</td>
<td>Quarto</td>
<td>215mm</td>
</tr>
<tr>
<td>a(width</td>
<td>length)</td>
<td>Engineering A</td>
<td>8.5in</td>
</tr>
<tr>
<td>b(width</td>
<td>length)</td>
<td>Engineering B</td>
<td>11in</td>
</tr>
<tr>
<td>c(width</td>
<td>length)</td>
<td>Engineering C</td>
<td>17in</td>
</tr>
<tr>
<td>d(width</td>
<td>length)</td>
<td>Engineering D</td>
<td>22in</td>
</tr>
<tr>
<td>e(width</td>
<td>length)</td>
<td>Engineering E</td>
<td>34in</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>(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>(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>(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>(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>(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>(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>(nanumber9envelope</td>
<td>Envelope number 9</td>
<td>3+7/8in</td>
<td>8+7/8in</td>
</tr>
<tr>
<td>number9envelope</td>
<td>env9)(width</td>
<td>length)</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>Envelope number 10</td>
</tr>
<tr>
<td>(nanumber11envelope</td>
<td>number11envelope</td>
<td>env11) (width</td>
<td>Envelope number 11</td>
</tr>
<tr>
<td>(nanumber12envelope</td>
<td>number12envelope</td>
<td>env12) (width</td>
<td>Envelope number 12</td>
</tr>
<tr>
<td>(nanumber14envelope</td>
<td>number14envelope</td>
<td>env14) (width</td>
<td>Envelope number 14</td>
</tr>
<tr>
<td>(envinvite</td>
<td>inviteenvelope</td>
<td>invite)(width</td>
<td>Invite envelope</td>
</tr>
<tr>
<td>(envitaly</td>
<td>italyenvelope</td>
<td>italy)(width</td>
<td>Italy envelope</td>
</tr>
<tr>
<td>(envmonarch</td>
<td>monarchenvelope</td>
<td>monarch)(width</td>
<td>Monarch envelope</td>
</tr>
<tr>
<td>(envpersonal</td>
<td>personalenvelope</td>
<td>personal)(width</td>
<td>Personal Envelope</td>
</tr>
<tr>
<td>(usstandardfanfold</td>
<td>usstdfanfold)(width</td>
<td>US standard fanfold</td>
<td>14.875in</td>
</tr>
<tr>
<td>(germanstandardfanfold</td>
<td>germanstdfanfold)(width</td>
<td>German standard fanfold</td>
<td>8.5in</td>
</tr>
<tr>
<td>(iso0</td>
<td>a0</td>
<td>dina0)(width</td>
<td>ISO/DIN &amp; JIS A0</td>
</tr>
<tr>
<td>(iso1</td>
<td>a1</td>
<td>dina1)(width</td>
<td>ISO/DIN &amp; JIS A1</td>
</tr>
<tr>
<td>(iso2</td>
<td>a2</td>
<td>dina2)(width</td>
<td>ISO/DIN &amp; JIS A2</td>
</tr>
<tr>
<td>(iso3</td>
<td>a3</td>
<td>dina3)(width</td>
<td>ISO/DIN &amp; JIS A3</td>
</tr>
<tr>
<td>(iso4</td>
<td>a4</td>
<td>dina4)(width</td>
<td>ISO/DIN &amp; JIS A4</td>
</tr>
<tr>
<td>(iso5</td>
<td>a5</td>
<td>dina5)(width</td>
<td>ISO/DIN &amp; JIS A5</td>
</tr>
<tr>
<td>(iso6</td>
<td>a6</td>
<td>dina6)(width</td>
<td>ISO/DIN &amp; JIS A6</td>
</tr>
<tr>
<td>(iso7</td>
<td>a7</td>
<td>dina7)(width</td>
<td>ISO/DIN &amp; JIS A7</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>(isoal0</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>
</tbody>
</table>
### The Genero Report Designer framework

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

- **Menus and Toolbars** on page 238
- **Show, dock, or move a view** on page 240
- **Setting Preferences** on page 240
- **Access Help** on page 243

---

<table>
<thead>
<tr>
<th>Unit abbreviations</th>
<th>Unit Description</th>
<th>Width value</th>
<th>Length value</th>
</tr>
</thead>
<tbody>
<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)</td>
<td>ISO/DIN C10</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>
Figure 86: The Genero Report Designer framework

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.

Menus and Toolbars

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

- File menu
- Edit menu
- Search menu
- View menu
- Projects menu
- Diff menu
- Build menu
- Debug menu
- Database menu
- SCM menu
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.

The File > New menu

**File > New** creates a new file. When created, the corresponding Genero Studio component is opened.

Table 55: Reports Category

<table>
<thead>
<tr>
<th>Section</th>
<th>File Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report Designs</td>
<td>• Empty Report (.4rp) - Opens the Report Designer, populated with an empty report.</td>
</tr>
<tr>
<td></td>
<td>• List Report (.4rp) - Opens the Report Designer, populated with a basic list report.</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 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.
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.

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.

**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.

Setting Preferences

Customize Genero Studio to meet your needs.

- **General Preferences** on page 240
- **User interface preferences** on page 240

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

Save, test, or cancel changes you have made.

- **Load from default**
  Reloads the initial default configuration.

- **OK**
  Save and apply all modifications, then exit Preferences window.

- **Cancel**
  Undo all modifications and exit the Preferences window. The last saved values are restored.

- **Apply**
  Confirms your updates, allowing you to test the new configuration. (No save is performed.) If you want this configuration to become permanent, save it by pressing the OK button.

**General Preferences**

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

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

**Note:** The General Preferences settings are not used for Genero Report Designer.

**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:

<table>
<thead>
<tr>
<th>Visibility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The toolbar displays when any action assigned to the toolbar is within the scope of the current module or by the global framework. By default, all toolbars have their visibility set to Automatic.</td>
</tr>
<tr>
<td>Always visible</td>
<td>When set to &quot;Always visible&quot;, Genero Studio displays the toolbar. The scope of any toolbar actions are not considered.</td>
</tr>
<tr>
<td>Always hidden</td>
<td>When set to &quot;Always hidden&quot;, Genero Studio hides the toolbar. The scope of any toolbar actions are not considered.</td>
</tr>
</tbody>
</table>

**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.
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
Specify the paper settings of a report on page 37
Paper settings set the paper orientation, the page size format, and the report margins for a report.

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

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 56: gsreport options

<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>
</tbody>
</table>
Upgrading GRD 244

<table>
<thead>
<tr>
<th>Option</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 -h option to display command help, the default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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 use 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 57: 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 111.</td>
</tr>
<tr>
<td>The Image Box can now embed the first page of a PDF file.</td>
<td>See Image Box on page 121.</td>
</tr>
<tr>
<td>The new Toolbox object, PDFBOX, can embed an entire PDF.</td>
<td>See PDF Box on page 120.</td>
</tr>
<tr>
<td>You can create a Spider Web chart.</td>
<td>See Category charts and drawAs property.</td>
</tr>
</tbody>
</table>

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.
### Overview

Report elements now include a **comment** property.

### Reference

See **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 58: 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 <strong>section (Section)</strong> on page 146.</td>
</tr>
<tr>
<td>Support of Intelligent Mail bar code type.</td>
<td>See <strong>intelligent-mail</strong> on page 196.</td>
</tr>
<tr>
<td>New <strong>smartParse</strong> 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 <strong>smartParse (Smart Parse)</strong> on page 163 and <strong>code-128</strong> on page 171.</td>
</tr>
<tr>
<td>New gs1* bar code aliases.</td>
<td>See <strong>Bar Code type listing</strong> on page 167.</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 59: 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 44</td>
</tr>
<tr>
<td>Pivot tables support.</td>
<td>See Working with pivot tables on page 60.</td>
</tr>
<tr>
<td>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.</td>
<td>No further reference.</td>
</tr>
<tr>
<td>Null value support.</td>
<td>See The String Class on page 230 and The Numeric Class on page 215, Conditional Expressions.</td>
</tr>
<tr>
<td>Improved trigger updates. Algorithm improved to remove the need for frequent manual adjustments for each change within the data schema (rdd) file.</td>
<td>See Triggers on page 22.</td>
</tr>
<tr>
<td>QR code barcode support.</td>
<td>See <strong>qr-code</strong> on page 197.</td>
</tr>
<tr>
<td>Display position of footers. Layout nodes designated as footers display at the bottom of the Mini Page, providing a WYSIWFYG experience for the report designer.</td>
<td>See Add headers and footers to a report on page 22.</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 27.</td>
</tr>
</tbody>
</table>
Overview

Splitting of oversized elements across pages to prevent overfill.

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.

Support for reverse side printing.

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.

Fallback image support when the requested image for an Image Box is not found.

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.

Class property added for report elements.

Display and modify the sizing policy of containers.

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.

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.

Preference added to control the appearance of RTL expressions in the document view.

Added options to facilitate the mass generation of images that are sized by their content (e.g. for web sites).

Reference

See splitOversizedItem (Split Oversized Items) on page 148.

See Transform transparently (transformTransparently) on page 149.

See Specify a different reverse side page on page 33.

See sortBy (Sort By) on page 158 and sortAscending (Sort Ascending) on page 158.

See Image Box on page 121.

See Place a trigger within the report structure on page 22.

See class (Class) on page 135.

See Modify the sizing policy of containers on page 16.

See Business Graphs on page 125.

See layoutDirection (Layout Direction) on page 142.


---

**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.

---

**B**

**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)

**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

---

**C**

**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 254.

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:

```
ATTRIBUTES(STYLE="dialog",COMMENT="Do you want to delete the record?")
```

dialog

dynamic layout
 See relative positioning on page 253.

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® systems, 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 250.

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 246.

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 255.

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 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 248.

**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 87: Scrollgrid as a responsive tile list**
By default, the tiles are displayed linearly.
Runtime system

See Dynamic Virtual Machine on page 249.

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 246.

**T**

**trigger**
A node in the report structure that specifies what happens when data repeats.

*Note:* Used in Genero Report Writer

**U**

**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.

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

Numerics

3D charts
   category charts 48
   map charts 48

A

absolute positioning 13
address labels 36
aggregating data
   with transformations 111
area charts
   category charts 48
   XY charts 49

B

backside printing
   in reports 33
bar charts
   category charts 48
   map charts 48
bar code
   4-CB 196
   4CB 196
codabar-18 170
codabar-2 168
code-11-matrix 171
code-128 171
code-2-5-datalogic 176
code-2-5-IATA 176
code-2-5-industrial 176
code-2-5-interleaved 176
code-2-5-inverted 177
code-2-5-matrix 177
code-32 177
code-39 179
code-39-extended 184
code-93 188
code-93-extended 189
code-BCD-matrix 177
data-matrix 193
ean-13 194
ean-8 194
ean-code-128 194
ean-data-matrix 195
ean-supplemental-2 195
ean-supplemental-5 195
gs1-13 194, 195
gs1-8 194, 195
gs1-code-128 195
gs1-data-matrix 195
gs1-supplemental-2 195
gs1-supplemental-5 195
Intelligent Mail 196
intelligent-mail 196
list of 167

pdf-417 196
qr-code 197
report objects, properties 158
upc-a 200
upc-e 200
upc-supplemental-2 201
upc-supplemental-5 201
USPS 4-State Customer Barcode 196
USPS OneCode Solution 196
USPS4CB 196

bar codes
   in Tool Box 130

borders
   report objects
      color properties 166
      examples 17
      size expressions 19
      style classes 215
      style properties 165
      width properties 165
   table objects
      border property 135
      bottom border 134
      left border 143
      right border 146
      top border 149

business graphs
   creating 50
   custom keys 55
   in Tool Box 125
   outputting as tables 59
   types 48

C

captions
   adding to report 27

category charts
   adding to report structure 58
   creating 52
   custom keys 55
   in Tool Box 126

child node
   in report structure 21
   switch with parent 23

command options
   GenerateReport 100

conditional expressions
   in RTL 74

custom keys
   business graphs 55

D

data
   choosing methods 112

data schema
   adding to report design 26
data source
  arbitrary XML input source 28
data transformations 111
data values
  adding to report 27
Data view
  in report designer 26
debugging
  report design document 83
decimals
  formatting 118
design file, See report designer
designer, See report designer
designing a report, See report designer
duplex printing
  different reverse side 33
  start on odd page 34
duplicating data
  with transformations 113
dynamic layout 13

E
even page
  in reports 34
evenPhysicalPage() 34
expressions
  in properties 71

F
first page
  in reports 32
fonts
  bold 139
  italic 139
  name 139
  size 139
footers
  in reports 22
  in tables 46

G
Geometry
  Alignment 133
  Anchor X 134
  Anchor Y 134
  Scale X 147
  Scale Y 147
  X 151
  X-Size 151
  X-Size Adjustment 152
  Y 152
  Y-Size 152
  Y-Size Adjustment 152

H
headers
  in reports 22

in tables 46
HTML
  displaying in report 118
hyperlinks
  in reports 43

I
images
  based on variables 82
  displaying in report 121
  location 144
  report templates 96
Info Node
  layout-dependent reference 40
  Tool Box object 130
internationalization
  using translate() 82
invoices
  as Genero reports 9
  displaying page numbers 31
  report templates 85

L
labels
  address 36
  designing 35
last page
  in reports 32
layout-dependent reference
  printing 40
line breaks 40
line charts
  category charts 48
  XY charts 49

M
map charts
  adding to report structure 58
  color 56
  creating 50
  custom keys 55
  in Tool Box 125
margins
  PDFs
    crop properties 166
  report objects
    examples 17
    width properties 164
max
  numeric class function 223
PXML 78
menu
  New 239
  Save 239
  Save All 239
  Save As 239
metadata
  report design document 26
  report document properties 167
min
  numeric class function 223
  PXML 78

N
  numbers
    formatting decimals 118
    formatting within strings 83
    Numeric RTL class 215
    positioning in reports 20

O
  odd page
    in reports 34
  oddPhysicalPage() 34
  operators
    in RTL expressions 74

P
  padding
    report objects
      examples 17
      width properties 166
    table objects
      horizontal padding 141
      padding property 144
      vertical padding 151
  page breaks 40
  page numbers
    displaying in report 119
    displaying invoice page 31
    overview 29
    setting length 29
    using a string 30
  paper formats
    supported in Genero 233
  paper settings
    designing reports 37
    in Report Writer preferences 243
  parent node
    in report structure 21
    switch with child 23
  PDF
    display in reports 120
  pie charts
    color 56
    creating 50
  pivot tables
    adding dimensions 65
    adding measures 66
    arranging hierarchies 67
    creating 64
    data types 62
    definition 61
    in report structure 68
    in Tool Box 127
    performance considerations 63
    properties 65
    runtime configuration 62
    sample reports 64
    polar charts 49
    positioning in reports
      absolute or relative 13
      at bottom right 21
      in center 20
      multiple objects 15
      numbers 20
    pre-printed forms
      designing 39
    preferences
      accelerators 242
      general 240
      paper settings 243
      report writer preferences 243
      Toolbar Configuration view 241
      toolbars 241
      user interface 240
    properties
      expressions 71
    Properties view
      in report designer 25
  PXML expressions
    embedding RTL 79
    overview 78

R
  radar charts, See spider web charts
  reference
    access help 243
    menu configuration 241
  Reference Box
    layout-dependent reference 40
    Tool Box object 130
  relative positioning 13
  report design document, See report designer
  report design templates
    .art 93
    .rsd 90, 91
    Add Fields page 87
    creating from existing reports 98
    creating new templates 90
    creating reports from 84
    design features 93
    expanding at runtime 90
    filters 96
    filters list 85
    images 96
    modifying 99
    overview 84
    Schema Association page 86
    schema definition 91
    setting directory 100
    Variables page 88
  report designer
    business graphs 48
    Data view 26
    debugging 83
    design a report 12
    launching 13
overview 9
pivot tables 60
placing elements 13
Properties view 25
reference 114
Report Design window 9
report output 10
Report Structure view 21
set properties 25, 71
tables 44
Tool Box view 12
work area 11

Report Element Properties
alignment 133
anchorX 134
anchorY 134
author 167
Background Color (Color) 134
Baseline Type (Layout) 134
baselineType 134
bBorder 134
bgColor 134
Bold (Font) 139
border 135
Border (Table) 135
border color properties 17, 166
border style properties 17, 165
border width properties 17, 165
Bottom Border (Table) 134
Categories 131
Categories Title (Chart) 153
Category Key (Items) 153
categoryKey 153
categoryTitle 153
check 159
Check (Bar Code) 159
class 135
Class (Object) 135
clip 135
Clip (Layout) 135
Code Type (Bar Code) 159
Code Value (Bar Code) 159
Code Value Expression (Bar Code) 160
codeType 159
codeValue 159
codeValueExpression 160
color 135, 153
Color (Color) 135, 153
colsSpan 135
Column Span (Table) 135
comment 136
Compute aggregates on the innermost dimension (Chart) 154
Compute Average (Value) 136
Compute Count (Value) 136
Compute Distinct Count (Value) 137
Compute Maximum (Value) 137
Compute Minimum (Value) 137
Compute Totals (Value) 138
computeAggregatesInnermostDimension 154
computeAverage 136
computeCount 136
computeDistinctCount 137
computeMaximum 137
computeMinimum 137
computeTotal 138
Control Characters (Bar Code) 160
creator 167
Data Symbols per Line (Bar Code) 160
dataSymbolsPerLine 160
Default (Reference) 146
Display Fact Rows (Chart) 154
Display Recurring Dimension Values (Chart) 154
Display Selection (Chart) 154
displayFactRows 154
displayRecurringDimensions 154
displaySelection 154
domainLowerBound 154
domainUpperBound 155
Draw Labels (Chart) 156
Draw Legend (Chart) 156
drawAs 155
drawLabels 156
drawLegend 156
encoding 160
Encoding (Bar Code) 160
Enum Values (Value) 138
enumValues 138
Error Correction Degree (Bar Code) 161
errorCorrectionDegree 161
fidelity 138
fill 138
Fill (Image) 138
Fix Width (Table) 140
Floating Behavior (Layout) 139
floatingBehavior 139
fontBold 139
fontItalic 139
fontName 139
fontSize 139
format 140
Format (Page Number) 145
Format (Value) 140
fWidth 140
hAlign 140
Hide Page Footer On Last Page (Mini Page) 141
Hide Page Header On Last Page (Mini Page) 140
Hide Text (Bar Code) 162
hidePageFooterOnLastPage 141
hidePageHeaderOnLastPage 140
Hierarchies input order (Chart) 156
hierarchiesInputOrder 156
Horizontal Alignment (Table) 140
Horizontal Padding (Table) 141
Horizontal Rule (Table) 141
hPadding 141
href 141
href (Hyperlink) 141
hRule 141
id 141
id (Hyperlink) 141
indent 142
Indent (Text) 142
InfoNode Name (Reference) 146
Intended Resolution (Image) 142
intendedResolution 142
isNumeric 142
Italic (Font) 139
key 156
Key (Items) 156
Keys Title (Chart) 156
keysTitle 156
keywords 167
Layout Direction (Orientation) 142
layoutDirection 142
lBorder 143
Left Border (Table) 143
Localize Text (Text) 143
localizeText 143
location 144
Location (Image) 144, 150
margin width properties 17, 164
name 144
Name (Font) 139
Name (Object) 144
Name (Page Number) 144
noCheckDigits 162
noDigits 162
noText 162
Number Check Digits (Bar Code) 162
Number Digits (Bar Code) 162
objectType 150
Offset (Page Number) 145
Output Order (Chart) 156
outputOrder 156
padding 144
Padding (Table) 144
padding width properties 17, 166
page ranges 145
pageName 144
pageNoFormat 145
pageNoOffset 145
password 145
PDF crop properties 166
Prefer Rectangular Symbols (Bar Code) 162
preferRectangularSymbols 162
Proportional Width (Table) 145
pWidth 145
rangeLowerBound 157
rangeUpperBound 157
Raw Code Value (Bar Code) 163
rawCodeValue 163
rBorder 146
referenceDefault 146
referenceName 146
report document metadata properties 167
Right Border (Table) 146
roundedCorners 165
rule 146
Rule (Table) 146
Rule Color (Table) 146
ruleColor 146
scaleX 147
scaleY 147
section 146
Section (Layout) 146
Series Title (Items) 157
seriesTitle 157
Size (Font) 139
smartParse 163
Sort Ascending (Chart) 158
Sort By (Chart) 158
sortAscending 158
sortBy 158
Split Oversized Items (Mini Page) 148
splitOversizedItem 148
strikethrough 148
strikeThrough (Font) 148
subject 167
Swap X (Orientation) 148
swapX 148
tBorder 149
text 148
Text (Text) 148
Text Alignment (Text) 148
Text Expression (Text) 148
Text Fidelity (Font) 138
textAlignment 148
textExpression 148
Thin To Gap Relation (Bar Code) 163
Thin To Thick Relation (Bar Code) 164
thinToGapRelation 163
thinToThickRelation 164
title 167
title (general property) 149
Title (Metadata) 149
Top Border (Table) 149
Top N (Chart) 158
topN 158
Transform transparently (Orientation) 149
transformTransparently 149
Trim Text (Text) 149
trimText 149
Type (Object) 150
underline 150
Underline (Font) 150
URL 150
vAlign 150
value
pivot table 150
Value (Miscellaneous) 150
Values Title (Chart) 158
valuesTitle 158
Vertical Alignment (Table) 150
Vertical Padding 151
Vertical Rule (Table) 151
Visibility Condition (Object) 151
visibilityCondition 151
vPadding 151
vRule 151
x 151
X-Size 151
X-Size Adjustment 152
xAxisTitle 158
xAxisTitle (Chart) 158
y 152
Y-Size 152
Y-Size Adjustment 152
Index

yAxisTitle 158
yAxisTitle (Chart) 158
report schema transformations
   creating 112
   duplication 113
report structure
   business graphs 58
   example 24
   headers and footers 22
   pivot tables 67, 68
   switch child and parent 23
   tree 21
   triggers 22
Report Structure view
   in report designer 21
report templates, See report design templates
rest
   line breaks 40
   page breaks 40
   PXML 78
reverse side printing
   in reports 33
ring charts 48
RTL Class Reference
   Alignment class 209
   BaselineType class 209
   Boolean class 202
   BorderStyles class 215
   CategoryChartDrawAs class 213
   CodeType class 214
   Color class 203
   Date class 207
   Enum class 208
   FloatingBehavior class 212
   LayoutDirection class 210
   MapChartDrawAs class 213
   Numeric class 215
   Overview 201
   PageNoFormat class 211
   Runtime class 227
   Section class 212
   String class 230
   TextAlignment class 209
   TrimText class 211
   X-SizeAdjustment class 211
   XYChartDrawAs class 212
   Y-SizeAdjustment class 210
RTL classes 76
RTL expressions
   embedding in PXML 79
   examples
      changing display 81
      checking for values 80
      displaying images 82
      formatting numbers 83
      sizing columns 82
   language 74
   overview 72

S
scatter charts 49
smart parse
   bar codes 163
spider web charts 48
static layout 13
step charts 49
strings
   displaying in report 117
   formatting numbers 83
   including page numbers 30
   String RTL class 230
T
tables
   from business graphs 59
   in reports
      adding content 45
      changing width 47
      columns 46, 47
      creating 45
      headers and footers 46
      merging cells 46
      rows 46
      setting triggers 45
   in Tool Box 123
   pivot tables 60
template field triggers 93
template fields 94
templates
   report design, See report design templates
time series charts 49
Tool Box objects
   bar codes 130
   business graphs 125
   Business Graphs 48
   Category Chart 126
   Category Chart Item 126
   Decimal Format Box 118
drawables 117
   HTML Box 118
   Image Box 121
   Info Node 130
   Map Chart 125
   Map Chart Item 125
   Mini Pages 116
   Page Number (PageNoBox) 119
   PDF Box 120
   Pivot Table 127
   Pivot Table Fact 129
   Pivot Table Hierarchy Value 128
   Pivot Table Measure 129
   Pivot Table sample reports 64
   Pivot Tables 60
   Pivot Tables data types 62
   Pivot Tables definition 61
   Pivot Tables performance considerations 63
   Pivot Tables runtime configuration 62
   Pivot Tables, create 64
   propagating containers 116
   Reference Box 130
   references 130
   simple containers 115
Table 123
Word Box 117
Word Wrap Box 117
XY Chart 127
XY Chart Item 127

Tool Box view
in report designer 12

triggers
business graphs 58
in report structure 22
pivot tables 67, 68
report tables 45

U

units of measure
PXML expressions 78
supported in Genero 233

upgrade report design documents
gsreport command-line utility 243

V

variables
expressions in properties 71
report templates 88
views
managing views 240

W

waterfall charts 48

X

XML
arbitrary data source 28

XY charts
adding to report structure 58
creating 53
in Tool Box 127