FOUR Js
The Power of Simplicity

Four Js Genero Report Designer User Guide

Contents

Genero Report Designer User Guide..... 5

What's new in Genero Report Designer, v 3.00.....6

The Report Design Document.....7

Overview of Genero Report Designer.....7
- Genero reports.....7
- Launch the Report Designer..... 7
- The Report Design window..... 7
- The Report output..... 8

Designing a Report..... 9
- The Report Design window..... 9
- The work area..... 10
- The Tool Box view..... 11
- Placing elements on the report page..... 11
- Changing a report element type.....15
- Changing a property value (The Properties view).....15
- Adding report data (Data view)..... 16
- Organizing the report structure (the Report Structure view)..... 19
- Using page numbers..... 23
- Report Design Document metadata.....24
- Configuring the output..... 25

Design How-To.....28
- General design..... 28
- Print headers and footers.....31
- Print group totals and report totals..... 32
- Print totals at the beginning of a report..... 34
- Print a Layout-dependent reference (InfoNodes).....34
- Specify different first and last pages.....35
- Print an invoice page number instead of the physical page number..... 35
- Design Documents for preprinted forms..... 35
- Design labels..... 36
- Design address labels..... 38
- Modify an object's borders, margins, or padding..... 39
- Use hyperlinks in a report..... 41
- Some tips for legacy report designers..... 42
- Backside printing..... 43
- Debugging your Report Design Document..... 43

Working with tables..... 43
- Add a table to a report.....44
- Assign content to a table column.....44
- Set the triggers for a table in a report..... 44
- Merge cells..... 45
- Add rows or columns..... 45
- Add headers and footers.....45
- Change the width of a table.....46
- Change the width of a column..... 46

Working with business graphs..... 46
Report Writer business graphs.....47
Creating a graph.....49
Output charts as tables..... 55
Working with Pivot Tables.....56
What are pivot tables?..... 56
Sample pivot table reports..... 59
Create a static pivot table..... 60
Pivot table properties.....61
Arrange your hierarchies..... 61
Add a dimension.....62
Add a measure..... 62
Pivot table elements and the Structure view.....63
Expressions in properties..... 65
Overview..... 66
Using the expression language.....67
Using RTL classes..... 71
Using the PXML expression language.....72
Substituting 4GL variables for constants.....73
Expression examples.....74

Report Designer Reference..... 76
Report Design Elements (The Toolbox)..... 76
Simple Containers..... 77
Propagating Containers (Mini Pages)..... 77
Drawables..... 78
Business Graphs..... 85
References.....89
Bar Codes.....89
Report Element Properties..... 90
General Properties.....90
Properties related to margins and borders.....115
Properties for Report Document Metadata..... 119
Bar Codes.....119
Bar Code type listing.....120
Bar Code type details.....121
RTL Classes Overview.....156
Dimension Resolver..... 157
Unit Names..... 157
Paper Format Abbreviations.....158
Customize Report Designer: preferences..... 161

RTL Class Reference.....163
The Boolean Class.....163
The Color Class..... 164
The Date Class..... 167
The Enum Classes.....169
The Alignment Class..... 169
The TextAlignment Class..... 169
The BaselineType Class.....170
The LayoutDirection Class..... 170
The Y-SizeAdjustment Class.....171
The PageNoFormat Class..... 171
The TrimText Class..... 171

The X-SizeAdjustment Class....172
FloatingBehavior Class....172
Section Class....172
XYChartDrawAs Class....173
MapChartDrawAs Class....173
CategoryChartDrawAs Class....174
CodeType Class....174
BorderStyles Classes....176
The FGLNumericVariable Class....176
The FGLStringVariable Class....177
The Numeric Class....178
The Runtime Class....188
The String Class....192

Upgrading Genero Report Designer....197
Upgrading reports from prior versions....197
What's new in Genero Report Designer, v 3.00....198
What's new in Genero Report Designer, v 2.50....198

Genero Report Designer User Guide

Manual organization at a glance.

The Report Design Document on page 7	**Report Designer Reference on page 76**	**RTL Class Reference on page 163**	**Upgrading Genero Report Designer on page 197**
• Overview of Genero Report Designer on page 7 • Designing a Report on page 9 • Design How-To on page 28 • Working with tables on page 43 • Working with business graphs on page 46 • Working with Pivot Tables on page 56 • Expressions in properties on page 65	• Report Design Elements (The Toolbox) on page 76 • Report Element Properties on page 90 • Bar Codes on page 119 • RTL Classes Overview on page 156 • Dimension Resolver on page 157 • Customize Report Designer: preferences on page 161	• The Boolean Class on page 163 • The Color Class on page 164 • The Date Class on page 167 • The Enum Classes on page 169 • The FGLNumericVariable Class on page 176 • The FGLStringVariable Class on page 177 • The Numeric Class on page 178 • The Runtime Class on page 188 • The String Class on page 192	• Upgrading reports from prior versions on page 197 • What's new in Genero Report Designer, v 3.00 on page 6 • What's new in Genero Report Designer, v 2.50 on page 198

What's new in Genero Report Designer, v 3.00

This publication includes information about new features and changes in existing functionality.

These changes and enhancements are relevant to this publication.

Table 1: Genero Report Designer, Version 3.00

Overview	Reference
Genero Report Designer provides a LastPageFooter section property.	See section (Section) on page 108.
Support of Intelligent Mail bar code type.	See intelligent-mail on page 150.
New `smartParse` bar code property for bar code Code-128. When enabled, this allows you to enter the bar code value, and the internal code will be computed for you resulting in the shortest visual representation.	See smartParse (Smart Parse) on page 109 and code-128 on page 124.
New gs1* bar code aliases.	See Bar Code type listing on page 120.

The Report Design Document

- Overview of Genero Report Designer on page 7
- Designing a Report on page 9
- Design How-To on page 28
- Working with business graphs on page 46
- Expressions in properties on page 65

Overview of Genero Report Designer

Report Designer provides you with the tools to design Genero reports.

- Genero reports on page 7
- Launch the Report Designer on page 7
- The Report Design window on page 7
- The Report output on page 8

Genero reports

Genero reports can take a variety of formats.

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
- Pre-printed forms, where you define the content for a pre-printed form.
- Labels, where you define the content and label size to be printed on pages of labels.
- Business Graphs, where you specify the type of graph and the data items to graph.
- Compatibility reports, where you output a legacy Genero (`4gl`) report in ASCII text, using Genero Report Writer.

A report can be displayed in various output formats.

Launch the Report Designer

There are several ways to launch the Report Designer.

To open the Report Designer:

- Choose **File >> New, Reports** from the Genero Studio main menu.
- Double-click an existing report design document (`4rp`) in the Project Manager.

The Report Design window

The Report Design window displays when you launch the Report Designer.

In the Report Design window, you create a Report Design document (`.4rp`). The Report Design document (`4rp`) defines the report page as consisting of box-shaped containers, to hold the data objects and text or image objects that make up the report.

The Report Design window consists of the work area and several views. If a view is not visible, you may need to add it using the **Window >> Views** menu.

Work Area

The work area is located in the center of Genero Studio. When you open a report, this area contains the report page.

- Report elements (containers such as Mini Page and Layout Nodes, and their child elements such as Word Box and Image Box) and data values from the Data View can be dragged and dropped onto the report page and rearranged.
- Page Headers and Footers can be defined.

Structure View

The organization of the report can be seen in the Structure View. Containers and their contents can be moved around in the View to insure that they print in the correct order.

Triggers in the Structure View specify what should be printed when a trigger event (change in data) occurs.

Data View

Your Report Design Document will use an `rdd` file to determine the data schema for a report. The same `rdd` file can be used for multiple report definitions.

Your Report Design Document will use a data schema (`.xsd`) file to determine the data schema for a report. The same data schema file can be used for multiple report definitions.

The data schema file is specified in the Data View. Once 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.

Properties View

Each report element has properties, displayed in the Properties View, with values that you can change.

In addition to literal values, expressions can be used to change the value of report elements properties. 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.

Output View

The Output View is used by any module to display its output. The output of the various modules (messages, errors, results of commands, and so on) is displayed as tabs in the view.

Document Errors view

The Document Errors tab displays any error messages for the report design document.

The Report output

A report can be output in various formats and different page sizes, to various output devices.

Your report application can use reporting API functions to specify output details.

Default output and printer options can be set for each report design document (`4rp`).

To set default paper settings for a report, you would open the report (`4rp`) and select **File >> Report properties >> Paper Settings**. Paper settings include:

- The orientation of the page (portrait or landscape)
- The units of measure for the page (centimeter or inch)
- The page size format (standard or custom) as well as the type of paper (letter, legal, and so on)
- Margins.

To set a default output configuration for a report, you would open the report (`4rp`) and select **File >> Report properties >> Output Configuration**.

- Choose an output format, such as SVG, PDF, or Image. If you choose SVG format, it displays in the Genero Report Viewer. The Genero Report Viewer is provided as part of the Genero Desktop Client. It

provides a preview of the report, and allows the user to select a printer. To view a report output as PDF, you must have Acrobat Reader.
- Set rendering defaults.
- Set a default page range.
- If you select Image, you can then set Image Settings, to include the file type, the resolution in dpi, and the image prefix name for the created report file.

Designing a Report

The Report Designer consists of various views, menus and toolbar buttons that enable complex report designs.

- The Report Design window on page 9
- The work area on page 10
- The Tool Box view on page 11
- Placing elements on the report page on page 11
- Changing a report element type on page 15
- Changing a property value (The Properties view) on page 15
- Adding report data (Data view) on page 16
- Organizing the report structure (the Report Structure view) on page 19
- Using page numbers on page 23
- Report Design Document metadata on page 24
- Configuring the output on page 25

See also: Design HowTo

The Report Design window

The Report Design window displays when you launch the Report Designer.

To open the Report Design window:

- Choose **File >> New, Reports** from the Genero Studio main menu.
- Double-click an existing report design document (`4rp`) in the Project Manager.

When creating a new report, you can begin with:

- A blank report.
- A list report template that has a basic structure already in place.
- A report from a template.

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
- Structure View - a tree of the report containers and their contents.
- Properties View - a list of the properties for a selected report element.
- 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 out
- 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 work area

The work area in the Main Window provides a GUI interface to the report.

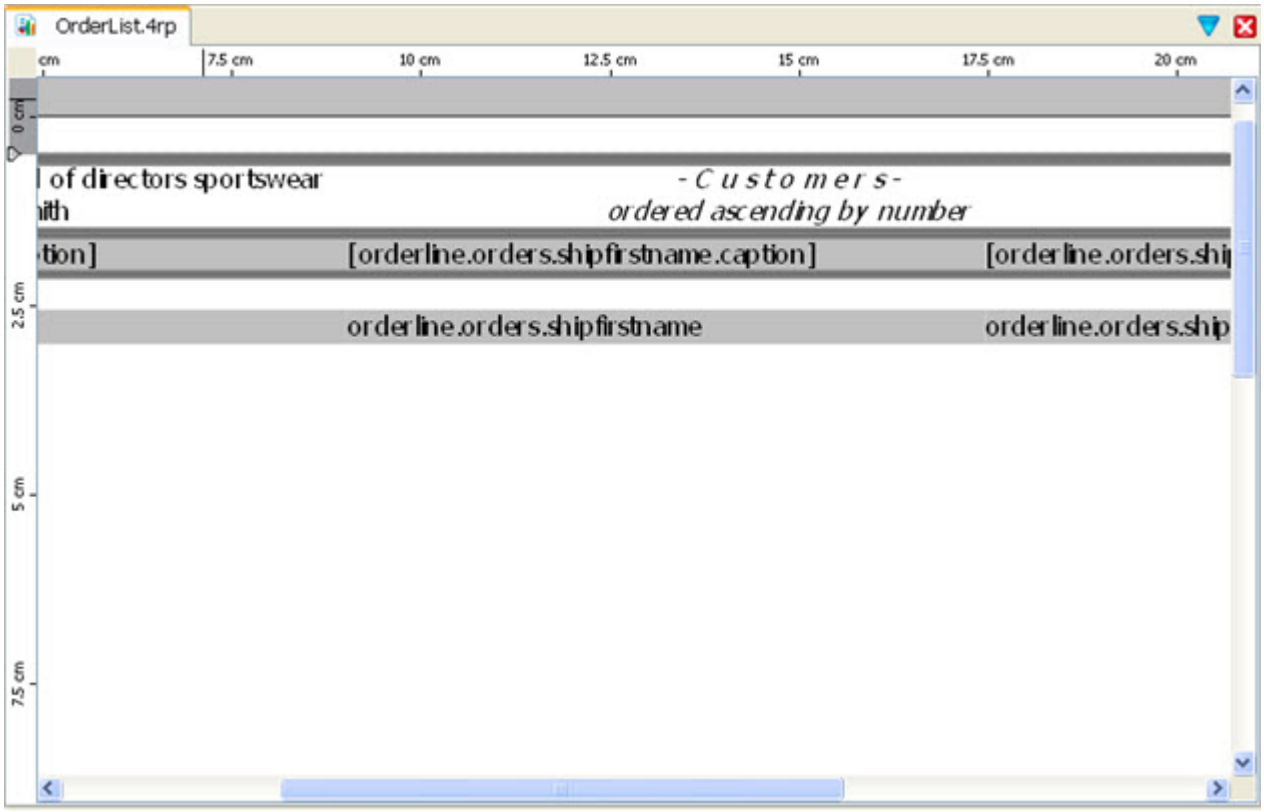


Figure 1: Report Designer work area

You design a report by initially dragging and dropping containers 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 Images are dragged and dropped into the containers.

From the Data View, you can drag and drop Data Items into a container, if you have specified the data schema.

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 WordBox has a text property where you can enter text to be displayed in the report.

If you select multiple elements, all items in the current selection are affected by the current operation, such as moving, sizing, or changing the type or text.

Use the **View>>Toggle View** menu or the Toggle View icon to toggle the work area between the report design and a preview of the report. When you preview a report, sample data is displayed on the page.

Figure 2: The Toggle View icon

Zoom buttons on the Toolbar allow you to zoom in and out on the report design document.

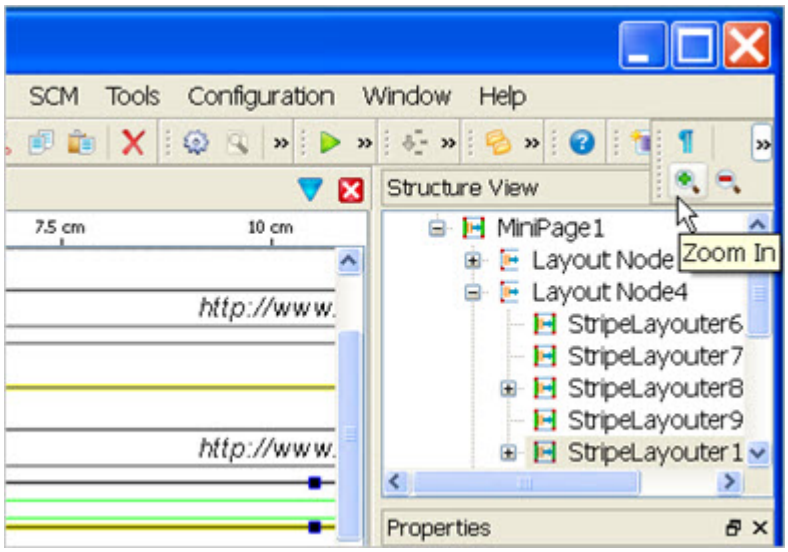


Figure 3: Zoom buttons

The Tool Box view

The Tool Box view provides report elements to place on a report design document.

The Tool Box view, typically displayed as a tab, provides the following report elements:

- Containers - for grouping other elements on the report page; see Choosing Containers
- Drawables - the report elements contained by a container; see Choosing Other Report Elements
- Business Graphs - the specific charts and items; see Working with Business Graphs
- References - to define layout-specific elements; see Choosing Other Report Elements
- Bar Codes - specific bar codes; see Bar Code Values

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

Placing elements on the report page

When placing elements on a report page, you determine whether the positioning is specific or dynamic by setting element properties that determine how the element acts as the report changes.

Specific positioning or automatic layout
The position of a report element can be specific, to enable reports that use pre-printed forms, or dynamic, adjusting as needed based on the length of the report element.

Dynamic Layouting	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 specific report items, this is the recommended method. When you drag the element, a moving red line indicates where the element will be located when you drop it. A colored 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 will be automatically positioned after the last existing child object in the container.
Specific Positioning	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 will snap to the closest point on the grid. A red dot on the element indicates its attachment point.

As you drag an element, a moving thin black line helps you line it up with other elements on the report if desired. The X and Y properties of the element in the Properties view indicate its location relative to its parent. These are automatically calculated when you drop an element into a container, or move it around. When you move it inside a container, the lines of the container are highlighted in yellow :

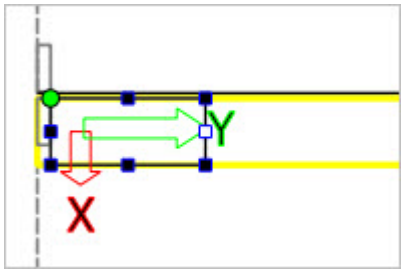


Figure 4: Highlighted container

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 dynamic layouting) or a red dot (for specific 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

You can drag and drop a multiple selection of objects from the Data View onto the report page (work area) or to the Structure View.

Use the CTRL and SHIFT keys to select the objects, then drag them to the desired location. An object will be created for each element selected, following the order in which they were selected.

If you drag to the Design work area using specific 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 will appear in a line. When you create a column object, a vertical container is used so the elements will appear stacked.

If you use dynamic layouting, or drag to the Structure View, the behavior is the same as if each element has been dragged individually.

Choose the right container

Selecting the right containers gives you the ability 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 used for the report 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 together, move the groups around, 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 the report 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 over 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.

Choose other report elements

After placing a container on a report page, elements are added for data and other report objects.

Data View

From the Data View:

Icons at the top of the data view allow you to specify the type of object you wish to create when you drag and drop a data value: whether you wish to drag and drop a reference to a data value, or a data item's caption (title), or to choose to have the values and captions aligned in a table format. See Data Values and Captions.

- For data values passed to the report, when you place the data item on the design page, it is automatically enclosed in a Word Box/ Word Wrap Box or Decimal Format Box container, depending on the data type.
- For captions (titles) for data items, the caption is automatically enclosed in a Word Box.

Tool Box view

From the Tool Box view:

- For additional text - use a Word Box or Word Wrap Box. You can enter the text when you drag the report element onto the report design page, or you can set the value of the text property in the Properties View. You can also double-click on the text in the report design page to edit it.
- 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 define the printed format, including decimal places, by setting the value of the format property in the Properties View. You can also double-click on the text in the report design page to edit it.
- For HTML pages - use an HTML Box on page 79. which displays the image of an HTML document in a report.
- For Images - use an Image Box, which allows you to specify the image to be printed by setting the location property.
- 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 83 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 85 object (chart or pivot table) for the specific type of graph. See Working with Charts.
- For values like "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 Design HowTo for additional information.

Modify the sizing policy of containers

Arrow-shaped controls on the four sizing knobs located at the center of the sides of the item in focus 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" *or*
- X-Size-Adjustment="shrinkToChildren" and Y-Size-Adjustment="shrinkToChildren"

Arrows pointing outward indicate a maximizing policy:

- X-Size="max" and Y-Size="max" *or*
- X-Size-Adjustment="expandToParent" and Y-Size-Adjustment="expandToParent"

Clicking on an arrow toggles its value.

Examples

These images illustrate some common cases.

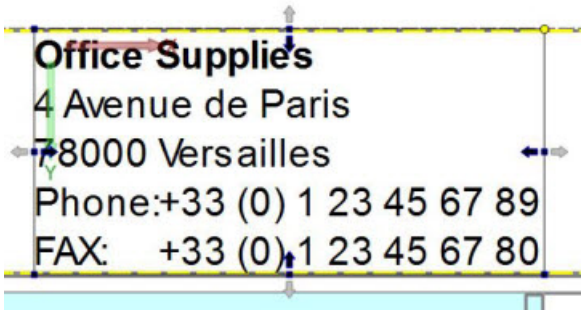


Figure 5: Container packs the content as tight as possible

Figure 6: Container packs content vertically and expands content horizontally

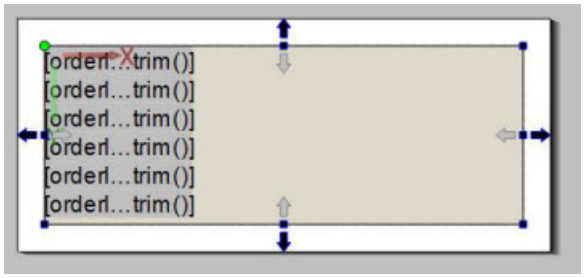


Figure 7: Container expands the content to use up the available space

A page root container typically expands in all directions to use up the available space.

Changing a report element type

To change a report element from one type to another, right-click the element in the report design document work area, and select **Convert To**; choose the new type from the list that displays. To convert the type of multiple report elements at once, Ctrl-click each element, and right-click to display the context menu.

The name property of the element will 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 will have the properties set that it has in common with the old type. For some type conversions, additional properties may be set. For example, when converting a Decimal Format Box to a Word Box, the **value** property is converted to a string value and assigned to the **text** property.

Changing a property value (The Properties view)

Select a report element in the report page (work area) or Structure View to display the property values in the Properties View. Once displayed, edit the property value.

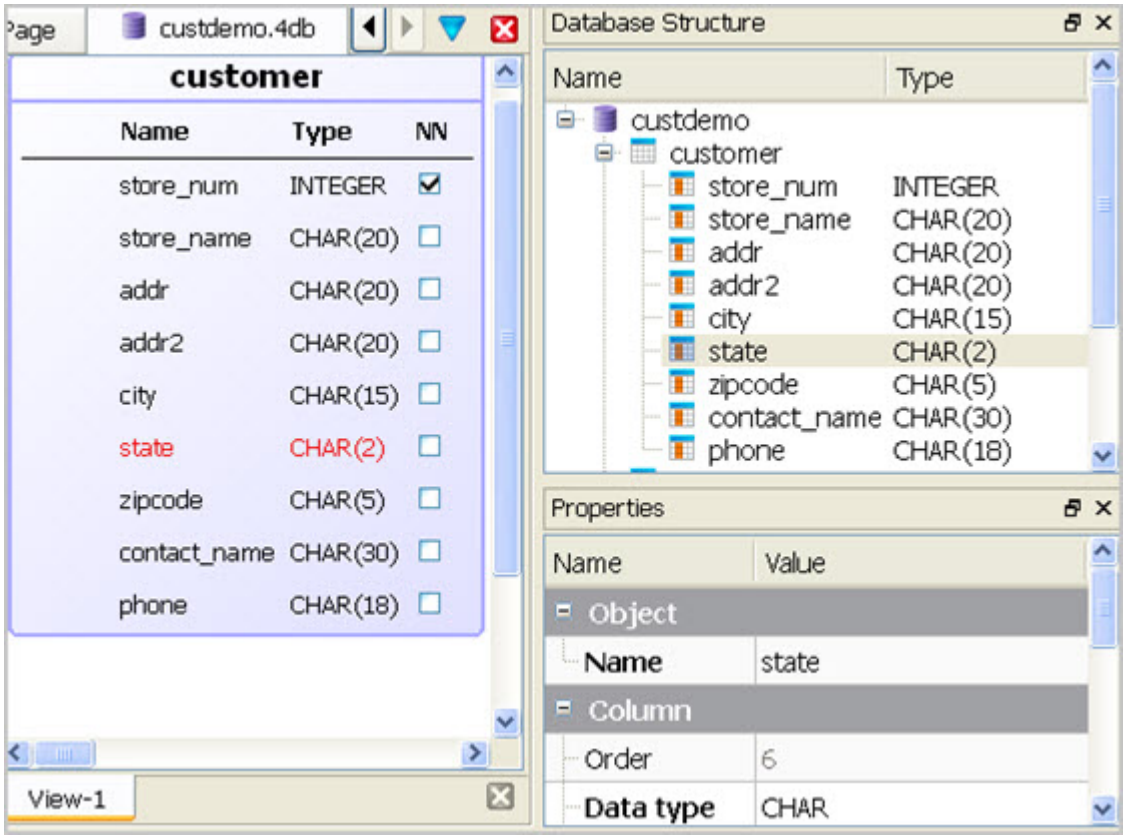


Figure 8: Properties View

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. All the properties are assigned a type, and the values entered must be valid for that type. The type of each property is listed in the Properties page.

> **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.

Using expressions for property values

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 RTL Expression language used in Report Writer closely follows the Java™ syntax for expressions and evaluation semantics.

- Arithmetic formulas can be used.
- Conditional expressions allow you to express IF/ELSE statements.
- Genero BDL variables can be used
- Functions from the Reporting API can be used.

Press the **fx** button to open the Expression Editing Window. See Using RTL Expressions for additional information.

Adding report metadata

Five string properties for Report metadata can be specified for the document root in the Report Designer Properties View.

The metadata fields include title, author, creator, subject and keyword .

The metadata is inserted into the final document, if the output format supports metadata.

In the case of SVG, the **title** property is used as a document caption in GRV.

For reports running in compatibility mode (reports having no associated `4rp` report design document), the values can be set by calls to the corresponding API functions. See the Genero Studio >> Report Writer documentation topic "Reporting API Functions".

The "Keywords" property is currently not working for "xlsx" format.

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 Report Schema file. The Report Schema file is extracted from the application source files:

- For applications written in Genero BDL, an `.rdd` file is extracted.
- For applications written in Java, C# or PHP, an `.xsd` file is extracted. An `.xsd` file defines an XML data source.
- 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 18.

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.

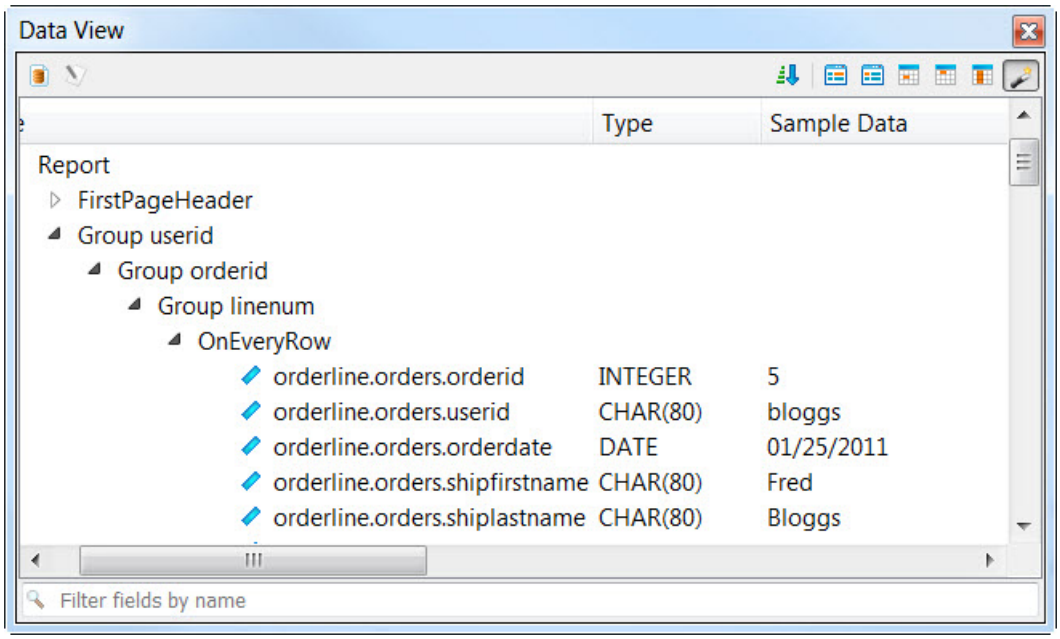


Figure 9: Data View

Click the **Open Data Report File** icon at the top of the Data View to specify the Report 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 2: Rules governing element creation based on context

Element	Condition
ITEM (“key” is set if field isn’t numeric, “value” otherwise)	Parent element is a MAPCHART

Element	Condition
CATEGORYITEM (“key” is set if field isn’t numeric, “value” otherwise)	Parent element is a CATEGORYCHART
Same object as option “Create a table column title object”	Parent element is of class `grwTableHeader`
Same object as option “Create a table column value object”	Parent element is of class `grwTableRow`
Same object as option “Create a table column value object for a column without title”	Parent element is of class `grwHeadlessTableRow`
Same object as option “Create a form fields value object”	If none of the above are applicable.

The data objects are automatically contained in a Word Box if the data type is defined as less than 30 CHAR, and in Wordwrap Boxes if the data type is defined as larger than 30 CHAR. If the data type is Numeric, the data object is contained in a Decimal Format Box.

The text property of the Word Box or Word Wrap Box, or the value property of the Decimal Format Box, specify what will print in the report output. The value property of the Decimal Format Box can be calculated using an Expression. See Using RTL Expressions.

Support for arbitrary XML data sources
Genero Report Writer can produce reports from arbitrary XML input sources.

The xml source is described by an XML Schema. The Open schema file dialog proposes `rdd` and `xsd` file formats.

If an `xsd` file is selected, the designer interprets the file as follows:

- Any XML Attribute is considered a variable.
- Any simple type element with minOccurs=1 and maxOccurs=1 is considered a variable.
- Any complex type elements with minOccurs=0 or maxOccurs>1 produce triggers.

For an example, see the `Table.4rp` report design document in the **Reports** sample project, located under the **OrderReportXML** application node. Open this report to see the `OrderData.xsd` used as data schema for the report (as shown in the Data View tab).

Encoding null values in the data

The attribute “`xsi:nil`” (with “`xsi`” representing the namespace “`http://www.w3.org/2001/XMLSchema-instance`”) can be used on an empty element to denote a null value. The RTL function `isNull()` for input variables will return true for such a variable. For example, consider the following document fragment:

```
<input xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”>
  …
  <productReference xsi:nil=”true”></productReference>
```

The RTL expression `productReference.isNull()` will yield true for this instance of the variable.

Limitations

Recursive references are not supported. Elements involved in a recursive reference will be ignored by the designer.

Optional variables are not supported. A variable is optional if one of these conditions is true:

- It is an XML simple type element with minOccurs="0".
- It is an XML Attribute without use="required" in the schema definition.

Simple type element with minOccurs="0" are discarded by the designer. No variable is created.

The designer creates variables for optional attributes and issues a warning. An error occurs if the variable is not present at runtime.

The Genero application

When you declare the data of a report design document (.4rp) to use an arbitrary XML data sources, you must use the `fgl_report_runFromXML` API function in your report. See fgl_report_runFromXML.

Organizing the report structure (the Report Structure view)

How do you make sure that the various elements print in the correct place in the report? Use containers - by combining horizontal boxes, stripes, word boxes, etc., in the correct order in the parent container, you control the order in which the elements in that parent container print out. Then you can move the containers around in the Structure View to make sure they print out in the correct spot.

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 data for your report is passed from your report program to the report one row at a time, sorted by the criteria specified in your report.

- For a Genero BDL program, the SQL statement defines the sort criteria.
- For a Java, C# or PHP program, the structure of the data model defines how the data is sorted.

Each row streamed to the report contains all the data or text specified by the data schema (the data report file). 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. The *triggers* in the structure of the report specify what should be printed when a change in data occurs. Arrange all the report elements for a single trigger in a parent container.

Triggers and Genero BDL report applications

Trigger nodes are specified by the `ORDER EXTERNAL` statement in the `REPORT` program block in your Genero BDL code, and indicate the data values by which the data is grouped. The final trigger node is `ON EVERY ROW`, specifying what is to be printed for each row of data passed to the report. The trigger nodes display in the Report Structure as red bullets:

Figure 10: The ORDER BY clause and report triggers

Triggers and Java, C# and PHP report applications

Trigger nodes are specified by the structure defined in the data schema (.xsd) file, which aligns with the hierarchy of the data in the data model.

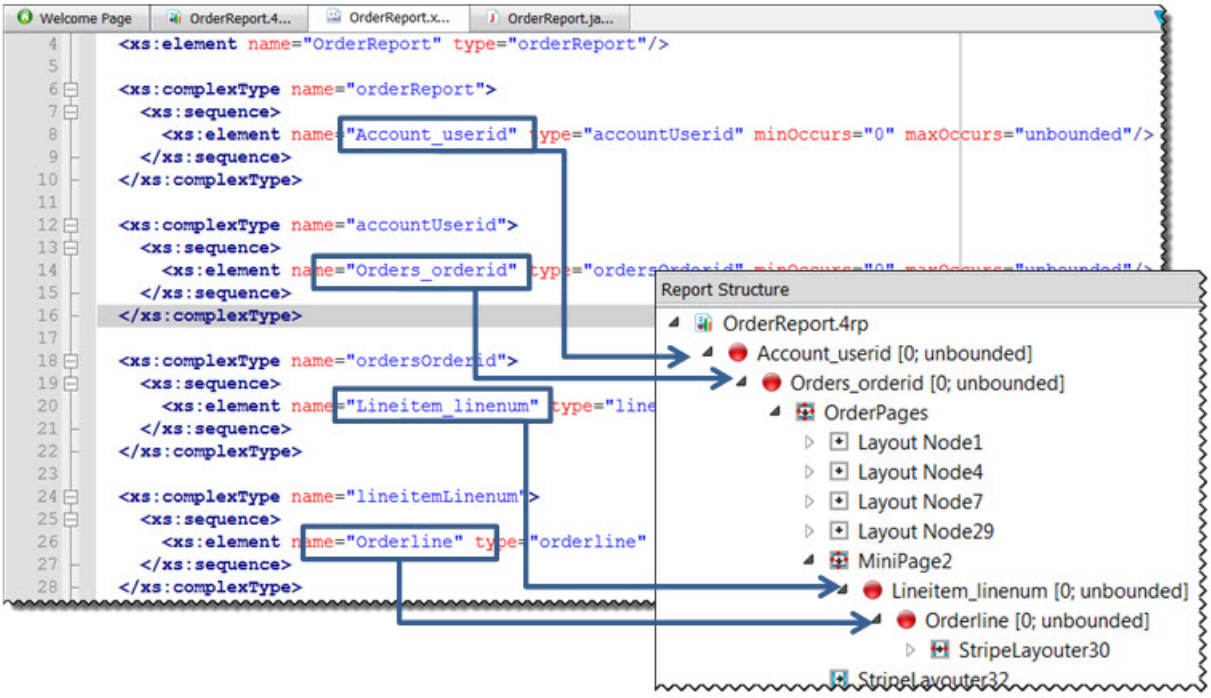


Figure 11: The data schema and report triggers

When the Data Schema changes

When the data schema associated with a report is modified, the Genero Report Designer regenerates the triggers to match the new schema. The Genero Report Designer will attempt to update the Report Structure using the minimum number of modifications required to perform the update. Cases of adding new triggers and cases of removing triggers that do not contain document fragments are handled automatically.

When it is necessary to move or remove a trigger that contains a document fragment, a warning displays in the Document Errors view stating that the fragment may require manual correction. These warnings are stored persistently in the report definition (`4rp`) file. The warning can only be removed by using the context menu **Clear trigger update message (issue is fixed)**.

The warning message `Data schema changed. The document has been updated to match the changes` is only displayed when a new trigger is inserted, a trigger is removed or a trigger is moved.

Place a trigger within the report structure

Report triggers appear in the report structure after you select the data schema for your report design. 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.

Page headers and footers

Arrange all the elements for a Page Header or Page Footer in a parent container that does not propagate.

A Vertical Box (Layout Node) is typically used. A layout node container has a section property. In this property, you specify the header or footer you are defining: first page header, any page header, last page header, first page footer, and so on.

Example report structure
Use the Structure View to examine the structure of your report.

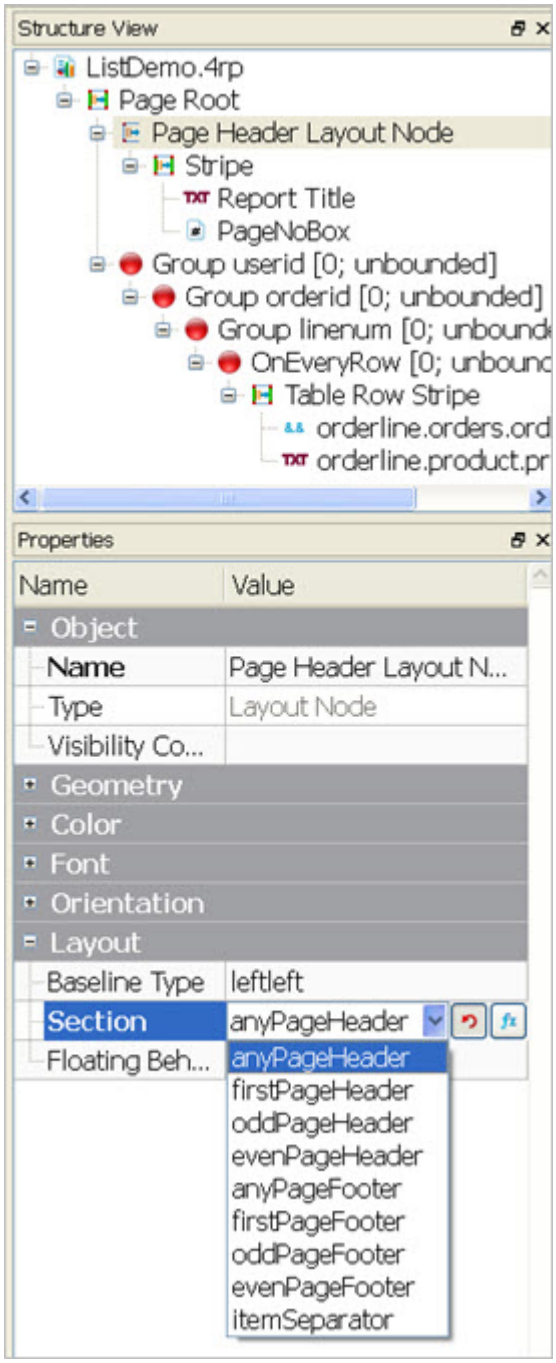


Figure 12: Structure View and Properties of a sample report

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 Page Headers and Footers. 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.

The Printed page
Placement of the header and footer containers take the page size into account.

When a report page is printed, any Page Header (if defined) will print at the top of the page, followed by the content of the containers associated with trigger nodes, followed by the Page Footer (if defined) 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 will be repeated when the report is run for as many pages as are required, based on the data passed to the Report Writer.

Using page numbers

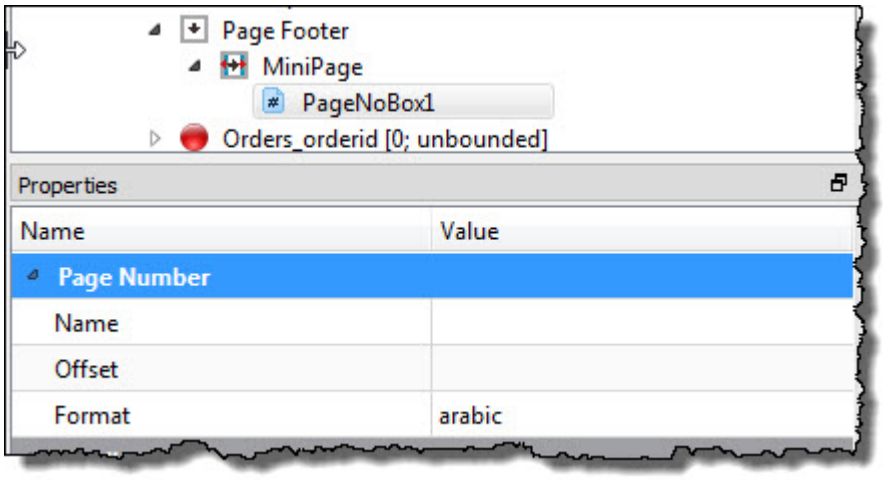
Use the Page Number drawable to print page numbers on a physical page.

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 MiniPage under the report trigger to use as a basis for the page number count. You can add a new MiniPage for this purpose if necessary.

Figure 13: Page Number Box properties



Genero Report Writer automatically calculates a Page Number container size based on a four digit page number or you can set the size with properties such as **X-Size** or **Text**. Set the length explicitly using the **X-Size** property (e.g. "3cm"), or use the **Text** property to hint a smaller size. For example, set the **Text** property to "000" to specify a maximum length of 3 digits. If multiple sizing properties are configured, Genero Report Writer uses the setting with the highest priority as follows (listed in highest to lowest priority):

1. X-Size
2. Text
3. Text Expression

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** (textExpression) property, it's 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 (i.e. "Page 9999 of 9999"). To override the default behavior you can set the **X-Size** to an explicit size or hint a smaller size with the **Text** property (e.g. "Page 99 of 99").

The values of the **Name** (pageName), **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()`, report pages waiting to be updated with the actual page count will be held back if printing is initiated from the viewer.

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 will start 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)
```

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. Enter your values for the properties.

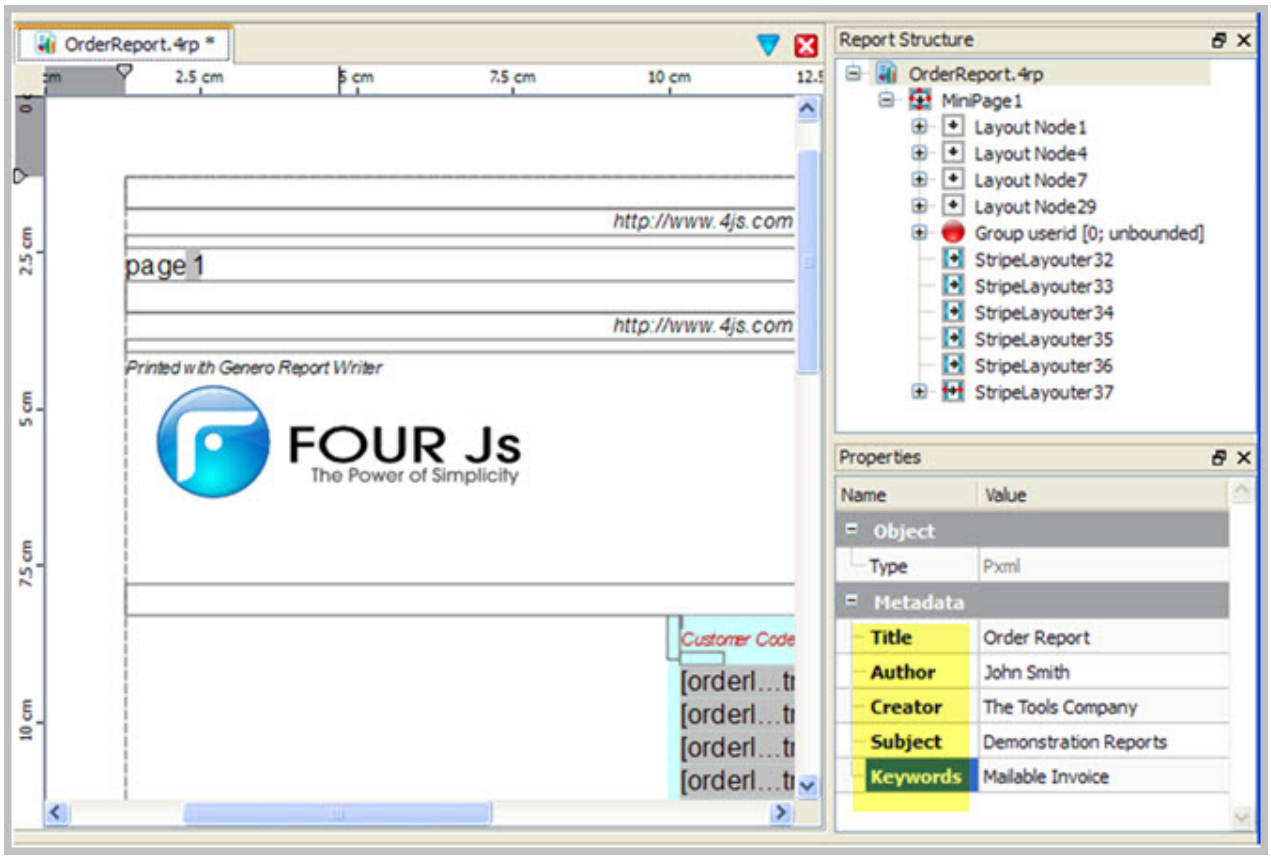


Figure 14: Report Design metadata

The metadata is inserted into the final document (PDF, SVG) if the format supports metadata. In the case of SVG, the **title** property is used as a document caption in the Genero Report Viewer.

If your report will run in compatibility mode (having no `4rp` design document), the values can be set by calls to the corresponding Report API functions: fgl_report_setTitle(), fgl_report_setAuthor(), fgl_report_setSubject() and fgl_report_setKeywords(). See the Genero Studio >> Report Writer documentation topic "Report API Functions".

Configuring the output

The same report can be output in different formats and different page sizes, and to different output devices.

In **Preferences** you can change the default paper settings for all reports: Tools>>Preferences, Report Writer, Paper Settings. See the Genero Studio >> Report Writer documentation topic "Report Writer Preferences".

The **File>>Report Properties** main menu option allows you to change the default report options for the currently open report design document.

- **Paper Settings -** select the page size and other paper settings for a report

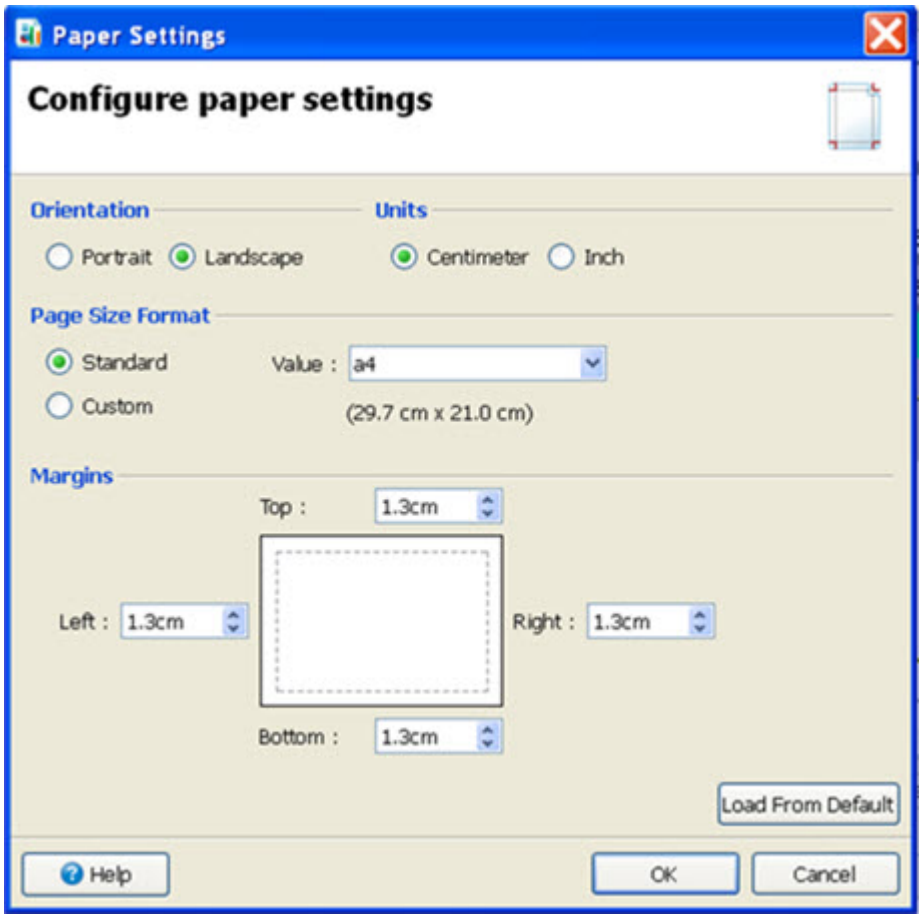


Figure 15: Paper Settings dialog

- **Orientation** - **portrait** or **landscape**
- **Units** - **centimeter** or **inch**
- **Page Size Format** - select from a list of common formats, or enter a custom height and width
- **Margins** - set **left**, **right**, **top**, and **bottom** margins

The **Load from Default button** restores the default values for paper settings as set in the Genero Report Writer Preferences. See the Genero Studio >> Report Writer documentation topic "Report Writer Preferences".

- **Output Configuration** - select the output format and other options

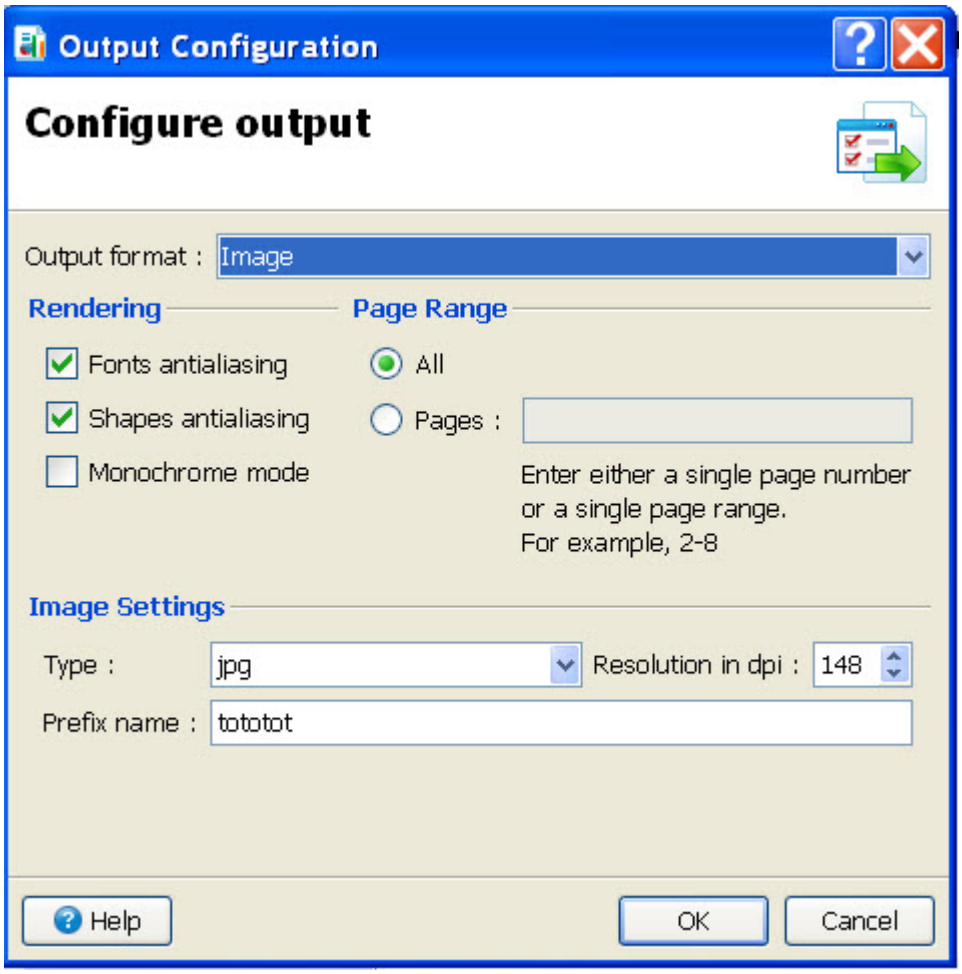


Figure 16: Output Configuration dialog

Options for **Output format**:

- **SVG** (scalable vector graphics - can be displayed using the Report Viewer feature of GDC)
  - **Rendering** - Select options to minimize the aliasing distortion
- **PDF** (Acrobat PDF format, can be displayed using PDF viewer)
  - **Rendering** - Select options to minimize the aliasing distortion or set monochrome mode
  - **Page Range** - output all pages or enter a range
- **Image** (creates an image, such as .jpg. You can select the image type.)
  - **Rendering** - Select options to minimize the aliasing distortion or set monochrome mode
  - **Page Range** - output all pages or enter a range
  - **Image settings** - Select image **Type**, **Resolution**, **prefix for the image** filename

Note: The Rendering options for font antialiasing (SVG or PDF documents) will only take effect if the fidelity property of report text elements is set to "True".

Functions from the Reporting API can be used in your report program to override the default options at runtime. For example, the function fgl_report_selectdevice provides additional output formats. See the Genero Studio >> Report Writer documentation topic "Report API Functions".

Design How-To

These procedures help you complete specific report design tasks.

- General Design
  - Align Numbers (format)
  - Center Elements
  - Set the paper settings of a report on page 29
  - Force a Page Break
- Print Headers and Footers
- Print Group Totals and Report Totals
- Print Totals at the beginning of the report
- Have Different first and last Pages
- Print an Invoice Page Number instead of the physical page number
- Print a layout-dependent Reference (InfoNodes)
- Design Documents for preprinted forms on page 35
- Design labels on page 36
- Design address labels on page 38
- Modify an object's Borders, Margins, and Padding
  - Size Expressions for Bordered Boxes
- Using Hyperlinks in a Report
- Some Tips for Legacy Report Designers

General design

These topics describe some common features of reports and how to do them.

Note: The basics of report design is discussed in Designing a Report.

Align and format numbers

Use proper containers and properties to align and format numbers.

Use the ToolBox object Decimal Format Box as the report element for numbers. It supports different locales and kinds of numbers, including integers, fixed-point numbers, and currency amounts ($123).

The text property specifies the value for the number to be printed. The value of this property may also be edited directly in the report design document by double-clicking the Decimal Format Box. The input cursor will be placed in the document, and the layout of the document is updated on each keystroke.

The format property specifies how the number will print out. The default value for this property is "---,---,---&.&&". You can change this string, using the specified symbols. The - (minus) symbol represent digits (if the number is negative, it will print with a leading -), and the period represents the decimal point. The & symbol fills with zeros any position that would otherwise be blank. If the actual number displayed requires fewer characters than the format string specifies, numbers are right-aligned and padded on the left with blanks.

Table 3: Formatting examples

Value of text property	Appearance in report
123456.1	123,456.10
15.24	15.24
-1600	-1,600.00

Center elements

Use properties to center elements.

To center an element in its parent container you can set its properties as described in Table 4: Centering elements on page 29

Table 4: Centering elements

Property	Value
x	max/2
y	max/2
anchorx	0.5
anchory	0.5

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 **anchorx** and **anchory** value of **0.5** sets the attachment point to the center of the element.

Set the paper settings of a report

Paper settings set the paper orientation, the page size format, and the report margins for a report.

Each report defines these paper settings:

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

Paper Settings order of precedence

There are three areas which determine the paper settings for a report, listed here in the order of precedence:

1. In the report application using reporting APIs.
2. In the report design document (`.4rp`).
3. GRW default values.

Set paper settings using the reporting APIs

A report application can use the reporting APIs to override the paper settings for an individual report.

- For Genero BDL applications, see Change paper settings and output format.
- For Java applications, see Change paper and page settings with Java APIs.
- For C# applications, see Change paper and page settings with .NET APIs.

Set paper settings for a report design document (`.4rp`)

The Genero Report Writer preferences specify the initial paper settings for a new report.

To change the paper settings:

1. Open the report in Genero Report Designer.
2. Select **File >> Report properties**, **Paper Settings...**. The **Paper Settings** dialog opens.
3. Set the desired orientation, units of measure, page size format, and margin settings for the report.

   Tip: To set the paper settings to the current values set in the **Paper Settings Configuration** of the local Genero Report Writer, click **Load from Default**.
4. Click **OK** to save your settings and close the **Paper Settings** dialog.

Set paper setting preferences for GRW

The **Paper Settings Configuration** page sets the default paper settings for all new reports.

1. Select **Tools >> Preferences**, **Report Writer**, **Paper Settings**.
2. Set the desired orientation, units of measure, page size format, and margin settings.

   Tip: To set the paper settings to the installation default values, click **Load from Default**.
3. Click **Apply** to save your preferences.
4. Click **OK** to close the **Preferences** dialog.

Once created, the paper settings for a report are stored as part of the report design document (`.4rp`). Changes to the **Paper Settings Configuration** have no effect on existing reports.

Force a page break

To force a page break, you set the X-size or Y-size property to "rest".

To force a page break, you can insert a container in the page, using the *rest* variable in the X-size or Y-size property of the container; this makes the container consume the remainder of its parent container. The property to be set is determined by the default layout direction of the container.

To force a page break after a MiniPage or Stripe container, set the height (X-size) to "rest".

To force a page break after a Layout Node, set the Y-size to "rest".

Table 5: Forcing a Page Break

Property	Value
X-size	rest
Y-size	rest

Switch child and parent nodes

In the Report Structure, you have two nodes in a parent-child hierarchy. You want to 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:

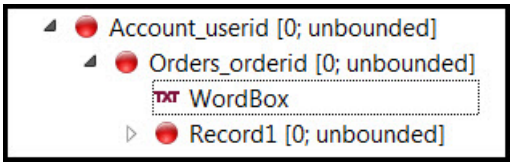


Figure 17: WordBox is child of Orders_orderid trigger

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

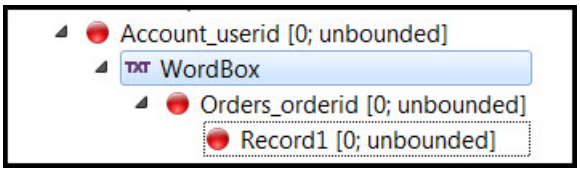


Figure 18: Orders_orderid trigger is child of WordBox

The two nodes have effectively switched places.

Print headers and footers

A MiniPage or PageRoot in your Report Design document defines how the content is to be laid out on the report page. Headers and Footers can be defined for the report pages, which can depend on the relative position of the page in the overall report.

You can use any Simple Container or Drawable as a container for a header or footer. Generally, a Vertical Box (LayoutNode) is used.

Add the header or footer container to a MiniPage or PageRoot. To identify the containers as headers or footers, and to specify where on the report page a container should print, set the container's section property to one of these ports:

firstPageHeader	Page Header, to print on the first page only; if this section is defined, subsequent Page Headers begin printing on the second page.
oddPageHeader	Page Header, to print on odd pages; has precedence over anyPageHeader.
evenPageHeader	Page Header, to print on even pages; has precedence over anyPageHeader.
anyPageHeader	Page Header is for every page, unless separate Odd and Even Page Headers are defined.
firstPageFooter	Page Footer, to print on the first page only; if this section is defined, subsequent page footers begin printing on the second page.
oddPageFooter	Page Footer, to print on odd pages; has precedence over anyPageFooter.
evenPageFooter	Page Footer, to print on even pages; has precedence over anyPageFooter.
anyPageFooter	Page Footer, to print for every page, unless separate Odd and Even Page Footers are defined.

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. In the parent Container, you cannot have multiple header or footer containers set to the same section.

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.* Within the container you can build up the header or footer using various containers and report elements. The Stripe container is useful for report elements that are to be laid out left to right across the page. Use Stripes, WordBoxes, etc. as needed, to arrange the contents of the header or footer within the container in the order in which it should be printed.

The properties HidePageHeaderOnLastPage and HidePageFooterOnLastPage provide flexibility in the printout.

> **Important:** It is an error if any element having the section set is preceded in the same sibling list by one that doesn't. In other words, any sections for the MinPage need to be specified first. Verify in the Structure view that the report structure is correct.

Print group totals and report totals

In the Genero BDL report program

In your Genero BDL report application, define variables for the totals, calculate the values, and output them to the report engine.

In the `REPORT` program block, define variables:

```
DEFINE
 data store_order_data,
 --Add variables for totals
 store_total, order_total, item_total, report_total DECIMAL(10,2)
```

Identify the data columns by which the data is grouped in the `ORDER EXTERNAL` command:

```
ORDER EXTERNAL BY
data.orders.store_num,
data.orders.order_num
```

Calculate the group and report totals, and output them in the `FORMAT` section.

```
FORMAT

  --Add FIRST PAGE HEADER, BEFORE GROUP OFs
  FIRST PAGE HEADER
    LET report_total = 0

  BEFORE GROUP OF data.orders.store_num
    LET store_total = 0

  BEFORE GROUP OF data.orders.order_num
    LET order_total = 0

  ON EVERY ROW
    --Add statements to calculate totals
    LET item_total = data.items.price*data.items.quantity
    LET order_total = order_total + item_total
    LET store_total = store_total + item_total
    LET report_total = report_total + item_total
    PRINT data.*, order_total, store_total, report_total
```

In the Java report program

In your Java report application, define variables for the totals, calculate the values, and output them to the report engine.

> **Note:** Code examples shown here can be found in `OrderReportModel.java`, part of the OrderReportJava demo application. Examine that file to see the code fragments in context of a complete application.

This application accumulates the subtotals on a per-row basis. First, define the appropriate variables:

```
    private double overalltotal;
    private double usertotal;
    private double ordertotal;
```

Reinitialize totals on appropriate iterator:

```
usertotal = nextRow.lineitemprice;
```

```
report.ordertotal = report.nextRow.lineitemprice;
```

Calculate totals on each row of data:

```
    overalltotal += lineitemprice;
    usertotal += lineitemprice;
    ordertotal += lineitemprice;
    nextRow.lineitemprice = lineitemprice;
    nextRow.overalltotal = overalltotal;
    nextRow.usertotal = usertotal;
    nextRow.ordertotal = ordertotal;
```

Specify variables in the `Row` class, with specific XML annotations (for relevant information in the data schema (`.xsd`), and use meaningful variable names:

```
    @XmlElement(name = "overalltotal", required = true, nillable = true)
    public Double overalltotal;
    @XmlElement(name = "usertotal", required = true, nillable = true)
    public Double usertotal;
    @XmlElement(name = "ordertotal", required = true, nillable = true)
    public Double ordertotal;
```

In the report design document (`.4rp`)

- In the Report Design window, define Stripes in the design document, one for each total. Use Decimal Format Boxes to hold the values.
- 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. It is important that the total stripes are the last child element of a report trigger node.

Consider this example:

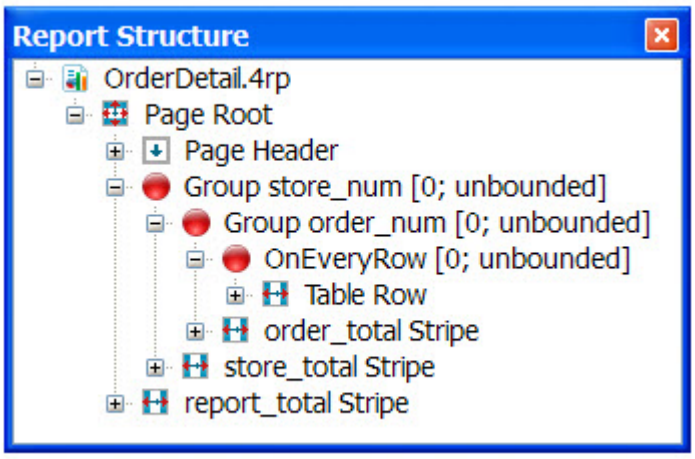


Figure 19: Example Report Structure

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 will place it at the very bottom of the child list for that trigger.

On each change of data, the specified data for every corresponding row will print out, and the appropriate Stripe will print out 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 will be 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

You can print aggregate values before the first detail row is printed.

For example, totals can be printed at the beginning for a report.

When this feature is used, the output has to 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 would be output. If the total number of records is small, the delay will hardly be noticeable; for example, when you print the order total before printing up to a few hundred rows relating to the order.

Print a Layout-dependent reference (InfoNodes)

InfoNodes allow you 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. In order to have a report layout that will work with various page sizes, you can use an InfoNode and a Reference Box.

This example illustrates how to print the total price (`overalltotal`) from a previous page.

In the Genero BDL report program

The `REPORT` block of the Genero BDL file must calculate the desired value and output it to the report. The following example is from the `OrderReport.4gl` file in the demo sample programs:

```
ON EVERY ROW
  LET lineitemprice = orderline.lineitem.unitprice *
 orderline.lineitem.quantity
  LET overalltotal = overalltotal + lineitemprice
  LET ordertotal = ordertotal + lineitemprice
  PRINT orderline.*, lineitemprice, overalltotal, ordertotal
```

The variable `overalltotal` contains the running total price of the lineitems on the report.

In the Java, C# or PHP report program

The report application must calculate the desired value and output it to the report. For example, you can create a variable named `overalltotal` to accumulate and hold the running total price of the line items of a report.

In the report design document (`.4rp`)

You will 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 will create an invisible column in your report line containing the value of the InfoNode.

  The Value property of the InfoNode must be a String. You can use the fglValue member of the FGLNumericVariable class to convert **overalltotal**:

  ```
  overalltotal.fglValue
  ```

This will format the value of **overalltotal** as a String based on the default format set in the Genero DVM. Or, 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 that will only be used to determine the maximum length of the value in the InfoNode, since 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 that is from a numeric data type, or "MMMM" as the maximum length for a value that is 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.

Specify different first and last pages

Use MiniPage containers to specify different first and last pages.

How do you have a different first or last page in a report?

- Add a separate MiniPage container for each page variation (first-page, main-report-page, last-page, for example), as children of the Page Root container.
- Add the report elements that are specific to each container.

In the Structure view the MiniPage containers should be listed in the order in which you want them to appear in the report. For example:

- Page Root
  - first-page - this is a "before" page, to print before the main content
  - main-report-page - this would contain all 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

Print an invoice page number instead of the physical page number

Print an invoice page number instead of the physical page number.

- Use the name property of the MiniPage container of the invoice to assign a name to the page.
- Add a Page Number report object to the page header of this MiniPage container.
- In the **Page Number** section of the properties for the Page Number object, set the **name** property to the name of the MiniPage Container.

The page number of the MiniPage will be printed on the report.

Design Documents for preprinted forms

A simple drag and drop of the report elements will invoke the Positioning method, which allows you to place an element in the desired location on the form.

To make this easier, add an Image box to the report, containing an image of the form to serve as a background.

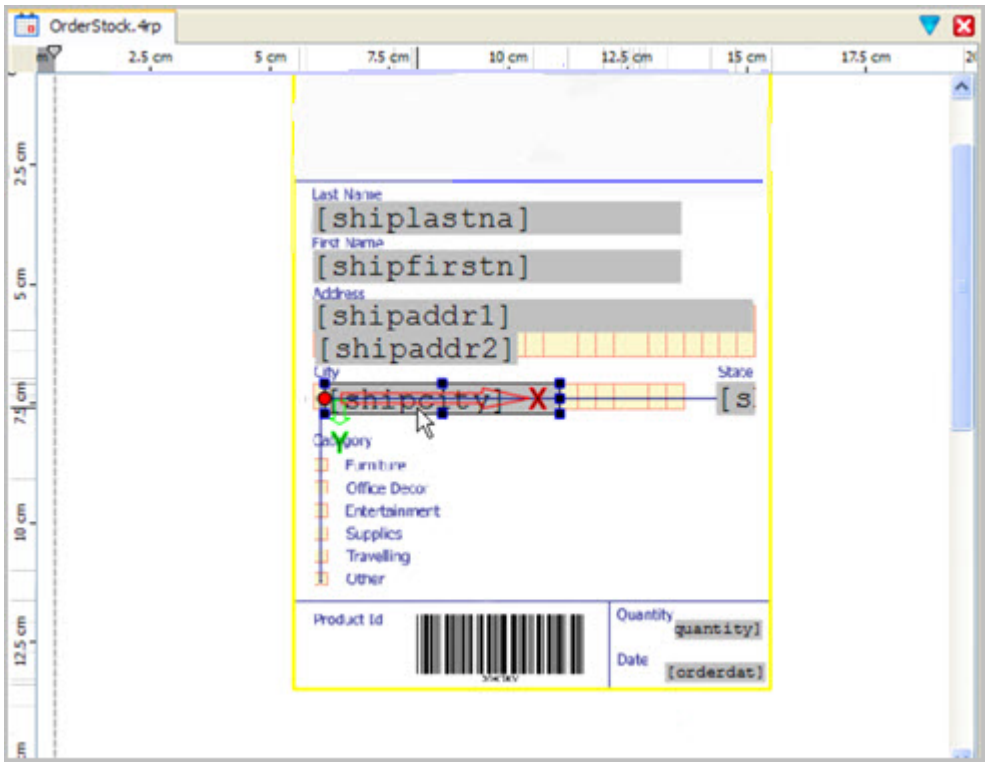


Figure 20: Report with Image

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.

Design labels

For a report application that prints out labels, the report design document (`.4rp`) is the size of a single label.

A report program programmed to output labels expects the report design document to represent a single label.

- To code a Genero report application that creates labels in a report format, see Create labels: the report program (Genero BDL).
- To code a Java report application that creates labels in a report format, see Create labels: the report application (Java).
- To code a .NET report application that creates labels in a report format, see Create labels: the report application (C#)

1. Create a new report.

   Select **File >> New**, **Reports**, **Empty Report (.4rp)**.
2. In the **Data View**, specify a Data Schema.

   See Adding report data (Data view) on page 16.
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.

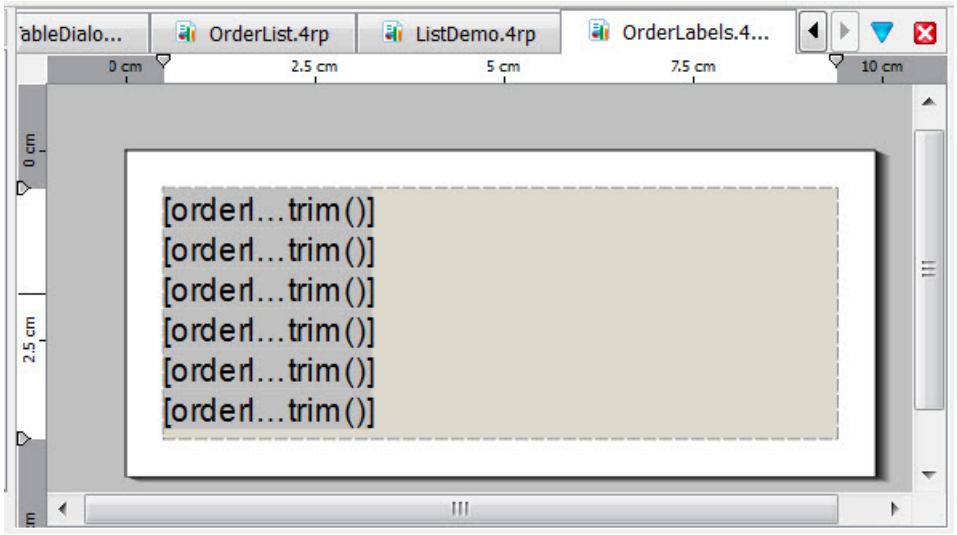


Figure 21: Label Report and Report Structure

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`.

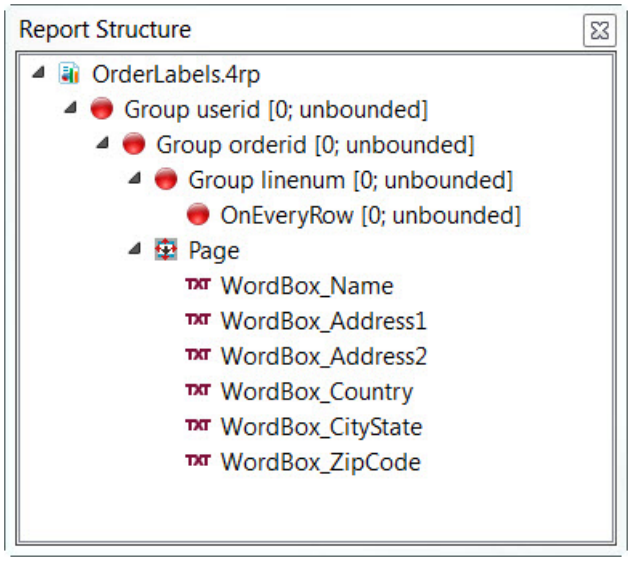


Figure 22: Label Report and Report Structure

6. Save your report.

For an example, see the `orderlabels.4rp` report in the OrderReport demo application.

Design address labels

Design for a label that may contain three to five lines, depending on the data record.

Before reading this procedure, you should be familiar with designing a basic label report. See Design labels on page 36 for more information.

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 ould set the visibilityCondition as follows: `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 non 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.

Modify an object's borders, margins, or padding

Any box object on a report design document can have margins, borders, and padding.

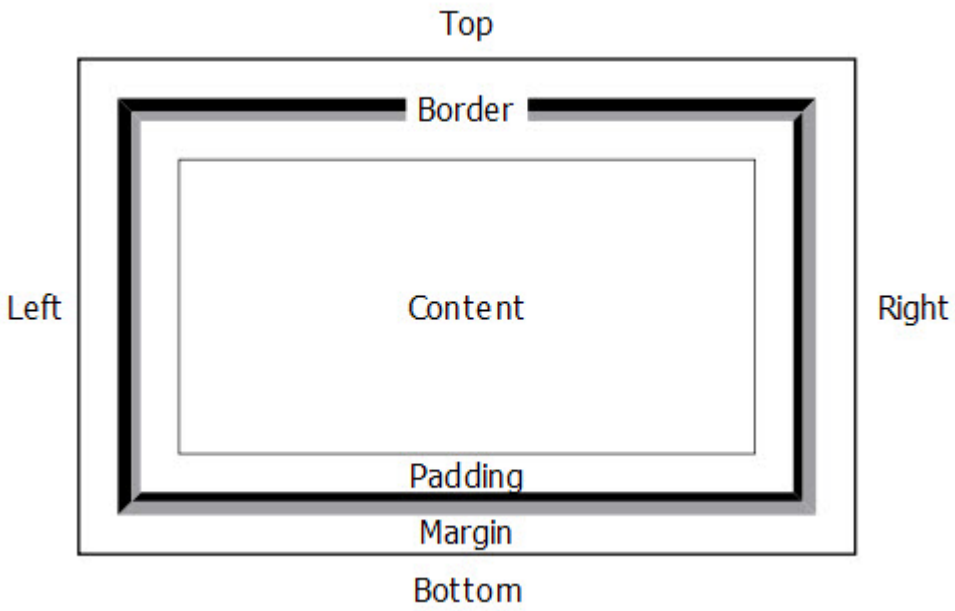


Figure 23: Border, Padding, and Margin

Set an object's specific properties in the Properties View to change:

- the width of a margin, border, or padding - marginWidth, marginRightWidth, marginBottomWidth, marginLeftWidth, marginTopWidth, borderWidth, borderRightWidth, borderBottomWidth, borderLeftWidth, borderTopWidth, paddingWidth, paddingRightWidth. paddingBottomWidth, paddingLeftWidth, paddingTopWidth
- the style of a border: solid, dashed, double, dotted, groove, ridge, inset, outset - borderStyle, borderRightStyle, borderBottomStyle. borderLeftStyle, borderTopStyle
- the color of a border - borderColor, borderRightColor, borderBottomColor, borderLeftColor, borderTopColor
- whether the box will have rounded corners (limited to the border styles solid, dashed, and double) - roundedCorners

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

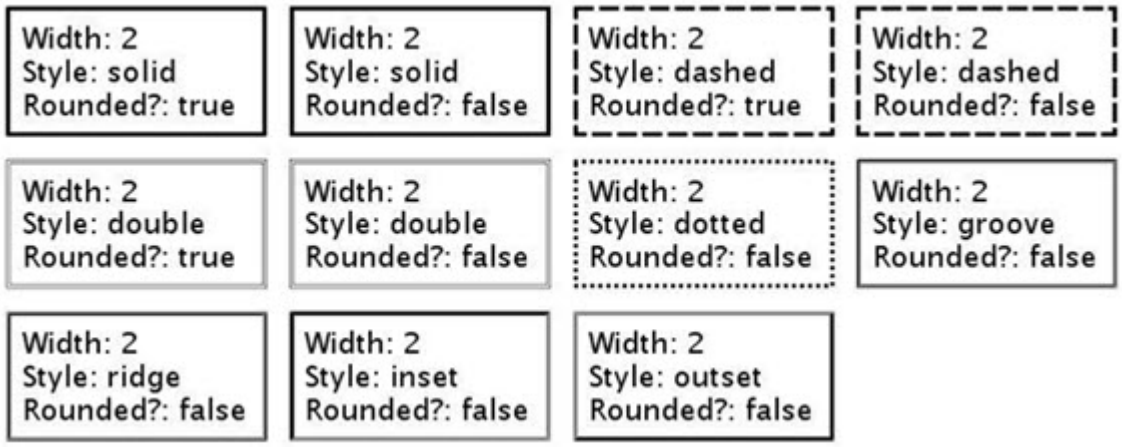


Figure 24: Examples of borders

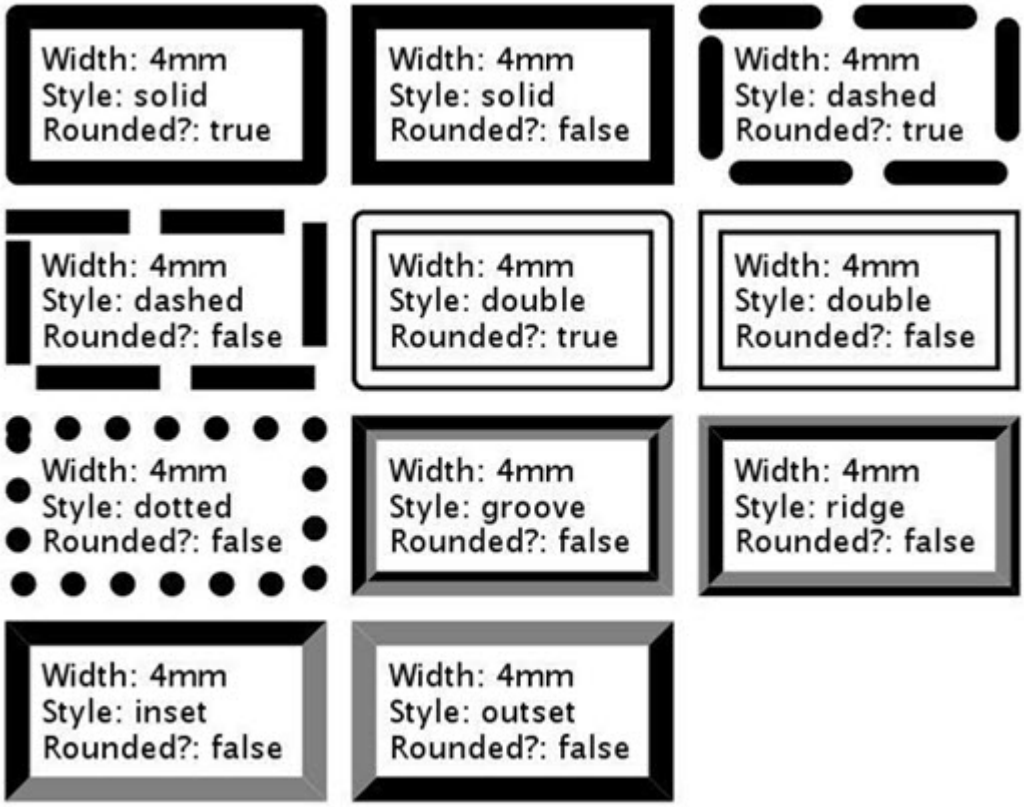


Figure 25: Examples of borders

Size expressions for bordered boxes
You can define the outer bounds of a box.

The x-Size and y-Size properties specify the inner size of the box; if we specify a box to be 3cm wide and have a 1mm thick border on all sides, for example, the box's outer bounds will 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, however, 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 will also be 3cm.

  Note: Do not use expressions that contain **max** as only one of its components, such as max/2, to specify the height and width of bordered boxes. Doing so can have unexpected results.

Size expressions that contain the variable max with other components
With bordered boxes, it may be necessary to customize expressions that use the max variable.

As explained in Size expressions for bordered boxes on page 40, the value of **max** is automatically adjusted by the border values. This causes unexpected results in expressions such as max-2cm"or "max/2", for example, where max is only a component of a more complex expression. Modify such 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".

Example

LAYOUTNODE: x-Size="max/2", leftBorderWidth="2mm", leftMargin="1mm", rightBorderWidth="1.5mm", rightPadding="3mm"

Changing the expression for x-Size:

- Before Step 1: x-Size="max/2"
- After Step 1: x-Size="(max+2mm+1mm+1.5mm+3mm)/2"; this is "f" in the explanation.
- After Step 2: x-Size="(max+2mm+1mm+1.5mm+3mm)/2-2mm-1mm-1.5mm-3mm", which can be consolidated to "(max-7.5mm)/2-7.5mm".

The final property values for the box are:

LAYOUTNODE: x-Size="(max-7.5mm)/2-7.5mm", leftBorderWidth="2mm", leftMargin="1mm", rightBorderWidth="1.5mm", rightPadding="3mm"

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).

id	Can be used to create an anchor in the document. Nodes can be identified with a unique id and then used as the target of a hyperlink.
href	Can be used to define a hyperlink pointing to any resource on the Internet, local disk, or any anchor

inside the document. The href should be defined using the URI syntax. For example:

```
http://www.google.com
```

```
mailto:santa.clauss@northpole.com
```

```
#ref
```

Hyperlinks are not supported in reports output to Image, Printer or Postscript formats.

Some tips for legacy report designers

This table answers some common questions the correlation between Report Designer and traditional 4GL commands in reports:

Table 6: Legacy Report to Genero Report Writer

Legacy Report command	Using Genero Report Writer
SKIP TO TOP OF PAGE	In the report design document, drop a container that will consume the remainder of the page. See Forcing a page break.
BEFORE GROUP OF, AFTER GROUP OF	There is a GROUP trigger for data control breaks in the report structure. The position and contents of the child containers of the trigger determine what is printed out and when. See Group and Report totals.
ON EVERY ROW	In the ON EVERY ROW statement of the BDL file, the PRINT statement just sends the data items to the report engine. The report design document specifies what is to be printed out for every row of data passed to the report.
SPACES, format strings	All of the formatting for the report line is done in the report design document. These keywords are no longer used in a PRINT statement in the BDL file.
PRINT	In the report design document, the Data View tab displays the list of data items in the order in which they are specified in the PRINT statement.
ON LAST ROW	Drop a container positioned as the last child of the page root. The contents of the container will print out after the last report row. See Report Total.
NEED *n* LINES	Put all the report elements that need to be kept together in a Vertical Box Layout Node container. If there is not sufficient room on the page to print all of the elements in the container, the entire container will be printed on the next page.
PAGE HEADER, PAGE FOOTER	Avoid using these control breaks, which are triggered by the line count of the BDL report, which does not correspond with the actual page breaks in the report output by Report Writer. Create page

Legacy Report command	Using Genero Report Writer
	headers and footers in the report design document instead.

Backside printing

You can specify that something print on the back side of each report page.

An even page header (or footer) that uses all of the available space in layout direction (Y-size="max") will be printed “between” all pages.

If a report has the pages 1, 2, 3, 4 then the backside ‘B’ is printed between the pages resulting in the sequence 1, B, 2, B, 3, B, 4.

If a backside is required after the last page, then an additional backside page need to be defined at the end of the report yielding 1, B, 2, B, 3, B, 4, B.

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.

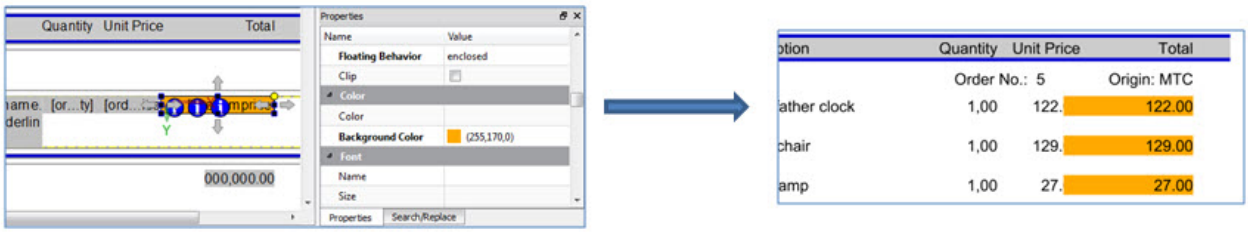


Figure 26: Setting the background color

Using GREDEBUG environment variable

Set GREDEBUG to check overfull boxes. Warning messages regarding any overfull boxes are written to standard output.

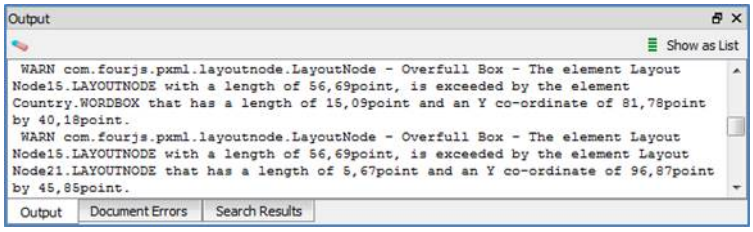


Figure 27: Overfull Box message

Working with tables

A table object has the ability to display data in columns and rows. You can add a table to a report, then manipulate the table to add or size columns and rows and change other display characteristics.

- Add a table to a report on page 44
- Assign content to a table column on page 44

- Set the triggers for a table in a report on page 44
- Merge cells on page 45
- Add rows or columns on page 45
- Add headers and footers on page 45
- Change the width of a table on page 46
- Change the width of a column on page 46

Add a table to a report

This procedure tells you how to quickly add a simple table to a new report.

While this procedure presumes you are starting with an empty report, it should provide 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 44.
5. Set the report trigger.
   See Set the triggers for a table in a report on page 44.
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

A table is comprised of columns and rows. This procedure tells you how to associate a field from the Data View to a column header and body element.

Although this procedure tells you how to associate a field to a column, a column can contain any content - to include another table, a chart, any content.

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, drag the field from the Data View and drop it into the header row for the desired column.
   It creates a column header. 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.
   It creates an expression 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

If you want each data row streamed to your report to result in a table row added to the table in your report, you need to set the appropriate trigger.

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

With a row, you can merge one or more columns (cells) into a single cell. You can also revert the merge and convert the merged cell back to its original number of cells.

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.

Add rows or columns

By default, a column has two rows and three columns. This procedure tells you how to add additional rows and columns.

To add a column or row to a report table object, you do not use the toolbox.

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 headers and footers

By default, a table has a single header (Any Page Header). You can add additional headers or footers as needed.

To add a header or footer to a table, you do not use the Toolbox.

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 would not add two headers of the same type; you would 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 would not add two footers of the same type; you would 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

You can specify the width of a table. If you do not specify a width, the table expands to the width of the parent container.

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

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.

A column width can be proportional or fixed.

When you specify a value in the Proportional Width property, you are specifying its width in proportion to other columns in the same table. Consider the following example: A table has three 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 157.

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

- Report Writer Business Graphs
  - Map Chart
  - Category Chart
  - XY Chart
- Creating a Graph
  - The Data

- The Design
- A Custom Key
- Output Charts as Tables

Report Writer business graphs

All of the Business Graphs in Genero Report Writer map Numeric values, grouping the data as required for the desired result.

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.

The MapChart layout object is defined by the MapChartDrawAs class.

Draw As property: This kind of graph can be drawn as a Pie, Pie3D, Bar, Bar3D or Ring, or as a table. For example, the pie chart in Figure 28: Pie Chart example on page 47 presents the total revenue (the value) for each customer (the key), based on the `OrderReport.rdd` file in the Report Writer sample programs.

Draw Legend and Draw Labels property: These properties have been added to customize the appearance of the plots. In particular the option to remove the legend is useful when more then several 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.

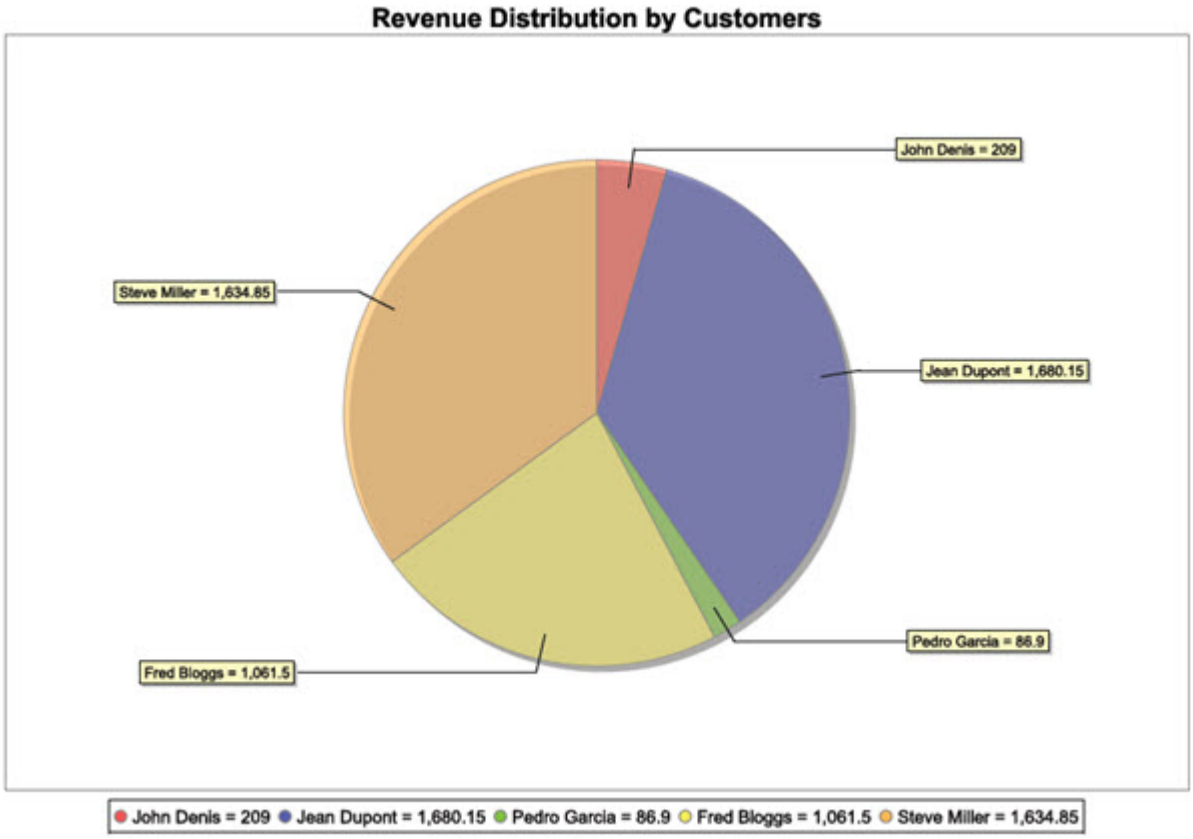


Figure 28: Pie Chart example

The **Map Chart Item** layout object specifies the key data item and the values data item in its properties:

- Key - the data is to be grouped by this property (customer name in the example chart); must be a String.

- Value - the chart will display the total of this property (total unitprice in the example chart) for each key (customer name); must be Numeric.
- Name - name of this report item in the Structure view; must be a String.
- Color - 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
The Category Chart layout object allows you to create a graph that has two keys for each set of values.

Draw As property: This kind of graph can be drawn as a Bar, Bar3D, Stacked Bar, Line, Line3D, Area, Stacked Area, Waterfall, or as a table.

In a waterfall graph, 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 waterfall graph can only have one category.

This Bar chart presents total revenue (the value) by the Customer Name and Product Category (the keys):

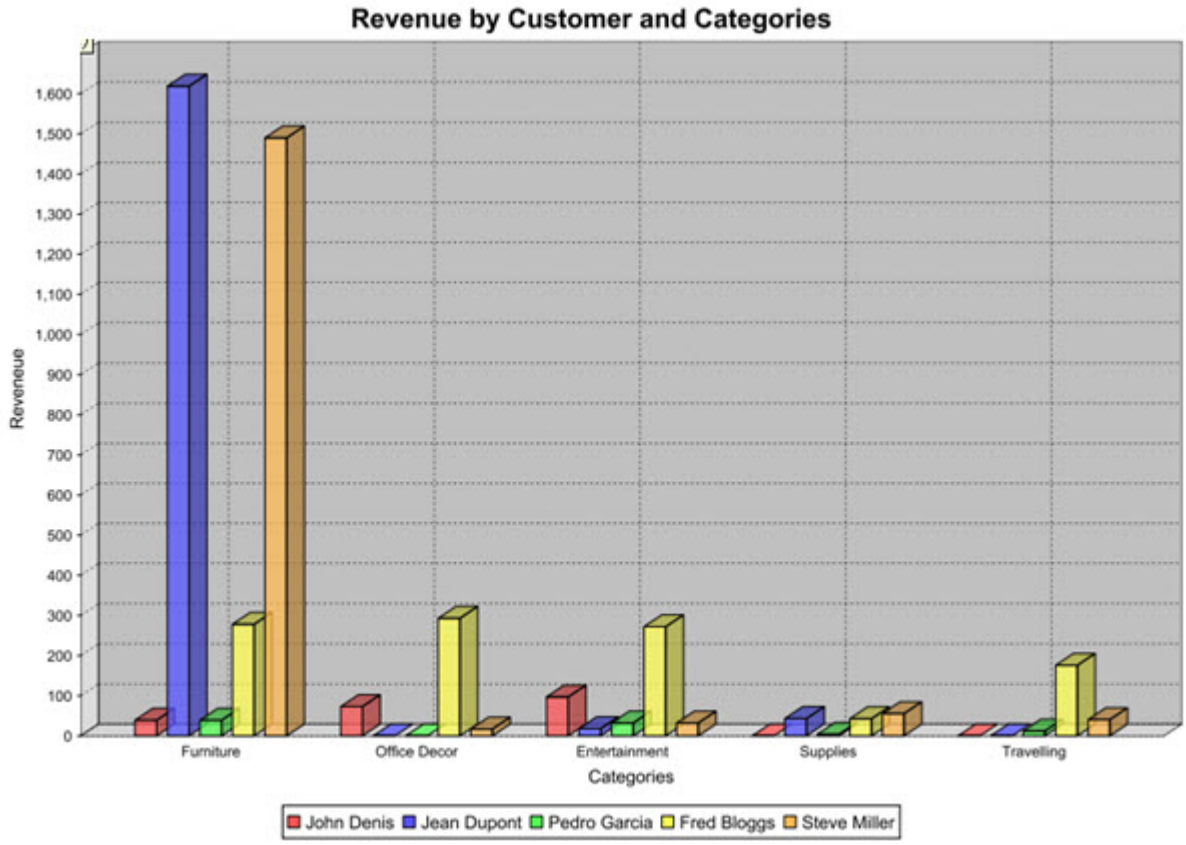


Figure 29: 3-D Bar Chart example: Revenue by Customer and Categories

The **Category Chart Item** layout object specifies the key data items and the values data item in its properties:

- Category Key - the categories for the data (the type of product purchased in the example chart); must be a String.
- Key - the data in each category is grouped by this property (the customer name); must be a String.

- Value - the chart will display the total of this property (the unitprice of order line items) for each category (product) divided into groups by Key (customer name); must be Numeric.
- Name - name of this report item in the Structure view; must be a String.

XY Chart
The XY Chart layout object allows you to create a graph that maps a series that has two values, as an XY-plot. This chart is used most often for scientific data.

Draw As property: This type of graph can be drawn as a Polar, Scatter, Area, Stacked Area, Line, Step, Step Area, a table or sorted table, or Time Series (property.) This graph presents the x and y values of three series of trigonometric functions.

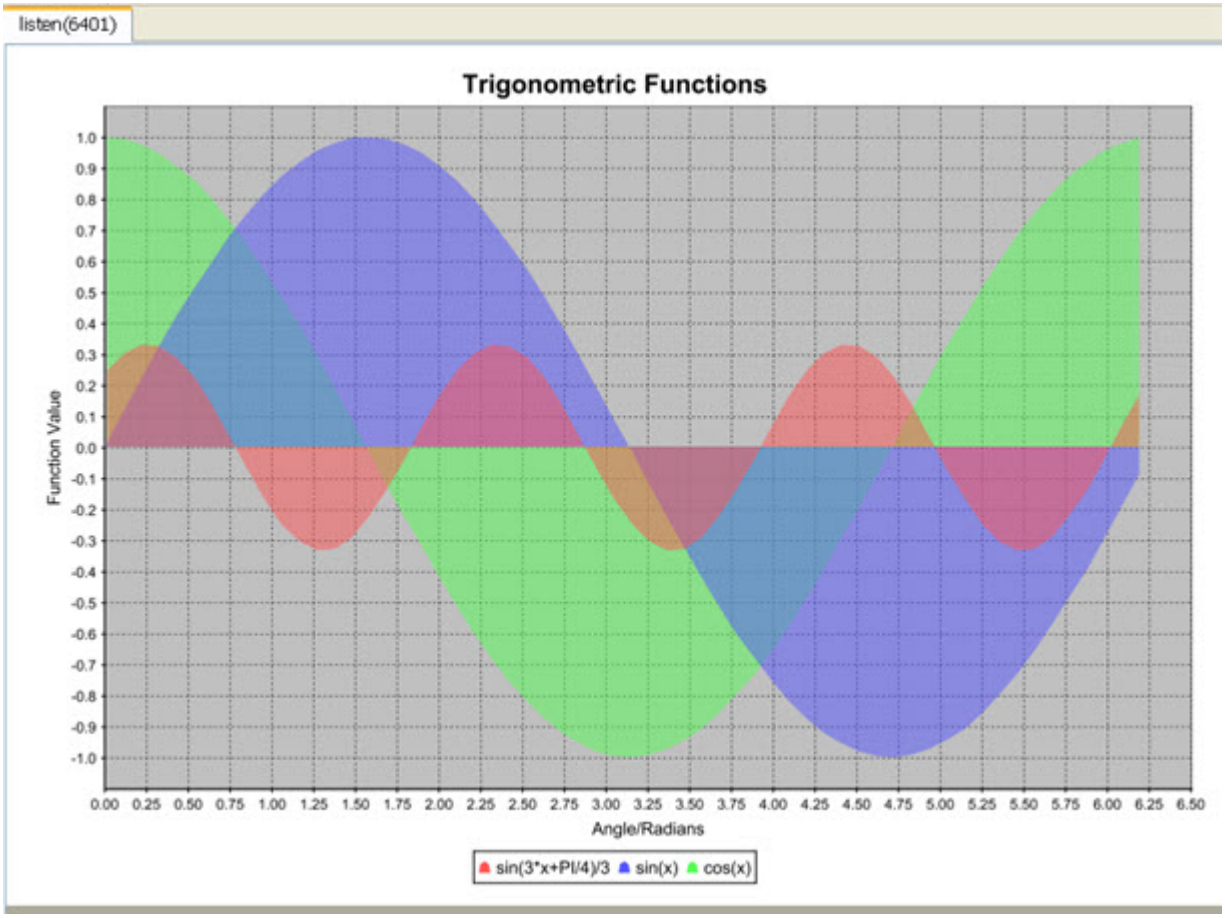


Figure 30: XY Chart example

The **XY Chart Item** element specifies the value data items in its properties:

- Series Title - the caption for the series of values being charted; there can be more than one series (three in Figure 30: XY Chart example on page 49); must be a String.
- X - the values for the x axis; must be Numeric.
- Y - the values for the y axis; must be Numeric.
- Name - name of this report item in the Structure view; must be a String.

Creating a graph

A Business Graph uses the same process as any other type of report.

See Steps to a Report for a list of the steps required to create a report.

The data
As with all reports, the data schema defines the data for the graph.

As in other types of reports, the PRINT statement in the BDL file specifies the data structure of the information that is available for the graph:

```
ON EVERY ROW
 LET lineitemprice = orderline.lineitem.unitprice *
 orderline.lineitem.quantity
 LET overalltotal = overalltotal + lineitemprice
 LET ordertotal = ordertotal + lineitemprice
 PRINTX orderline.*, lineitemprice, overalltotal, ordertotal
```

See Creating the BDL File for additional information.

You extract the data structure as an `rdd` file that you reference in the Data View page of the Report Designer:

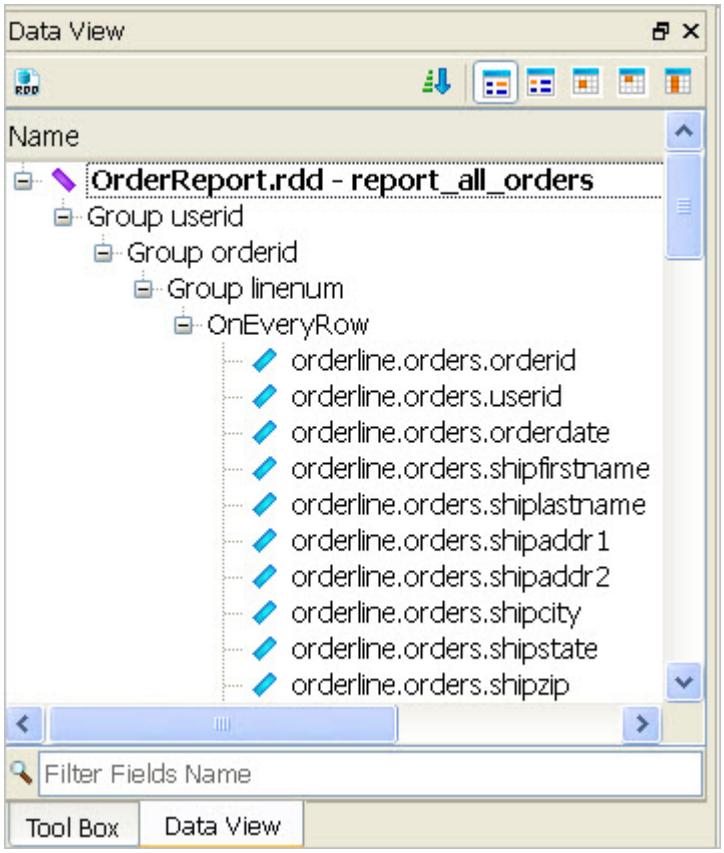


Figure 31: Report Designer Data View

A single `rdd` file can be used to design multiple reports containing the values of various data items from the file. The same data items can also be displayed as different types of charts.

The design

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 desired Business Graph (Map Chart, Category Chart, XY Chart, or Pivot Table) into a container on the report design.

   The Design Window will contain a Chart object and its related Item object (shown as a price tag). If you cannot view the entire chart in the Report Design window, you can re-size the display of the report in

the window: point the mouse at the report and scroll the mouse scroll button while holding down the keyboard Control key.

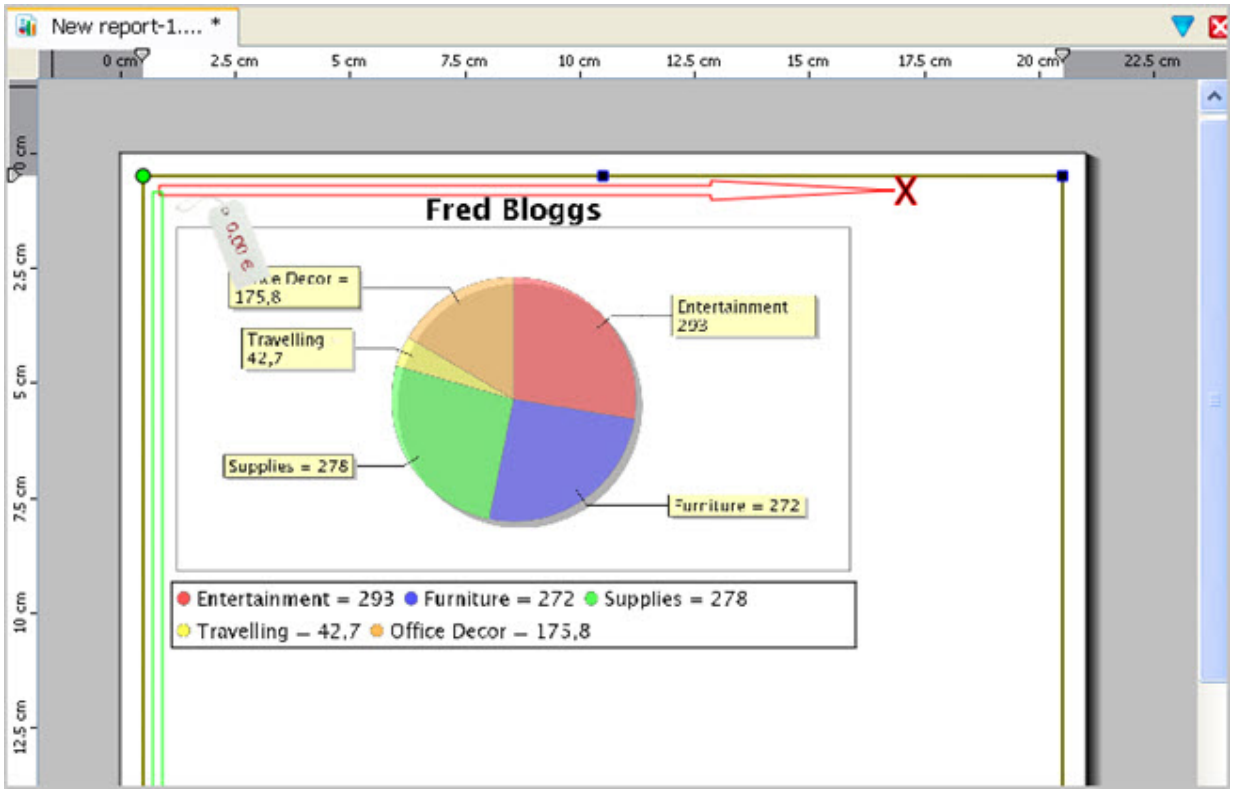


Figure 32: Chart object in the work area

4. Select the chart object and enter the values for its properties in the Properties View. The properties vary depending on the graph chosen:
   - Title - caption at the top of the graph
   - Values Title - caption for the values
   - Keys Title - caption for the keys
   - Category Title - caption for the categories (of a Category Chart)
   - X Axis Title - caption for the X axis (of an XY Chart)
   - Y Axis Title - caption for the Y axis (of an XY Chart)
   - Draw As - the type of chart
   - Sort By - sort alphabetically, numerically, or by input order.
   - Sort Ascending - sort in ascending or descending order.
5. Select the chart's price tag, which represents the item object, and enter values for its properties. The specific properties vary depending on the graph type.
   - **keys** - the data items that are used to group values, These data items must be Strings. You are not limited to existing String data items, however. You can define a custom string for the key, using the data items in a expression.
   - **values** - the data item that contains the numbers to be charted. These data items must be Numeric.
6. Modify the tree in the Structure view as needed:
   - The item object for the chart (Map Chart Item, Category Chart item, XY Chart item) must be a child of the OnEveryRow trigger node.
   - The location of the chart object (Map Chart, Category Chart, XY Chart) specifies how many different charts will be created, based on the resulting subsets of the data.

Drag and drop the nodes in the Structure view to create the correct hierarchy. 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.

Examples

With Figure 33: One chart per each row on page 52, the Structure View that resulted from the creation of a new report containing a chart. It will create a new page and chart for each data row passed to the report, and therefore needs to be modified:

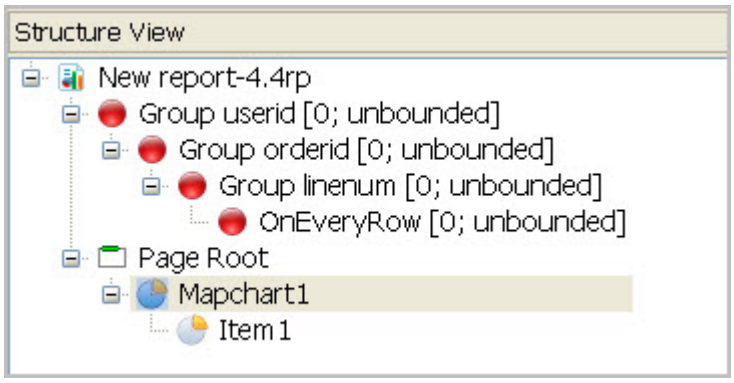


Figure 33: One chart per each row

With Figure 34: One chart, one page on page 52, the Structure view will result in a page containing one chart; the item node is a child of OnEveryRow:

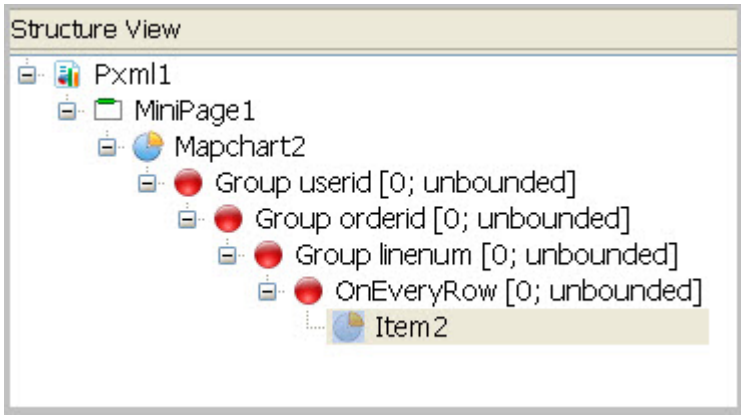


Figure 34: One chart, one page

With Figure 35: One chart for each unique userid on page 53, the Structure View will result in one chart for every unique userid, since the Map Chart is a child of the userid trigger node; the item node is a child of OnEveryRow:

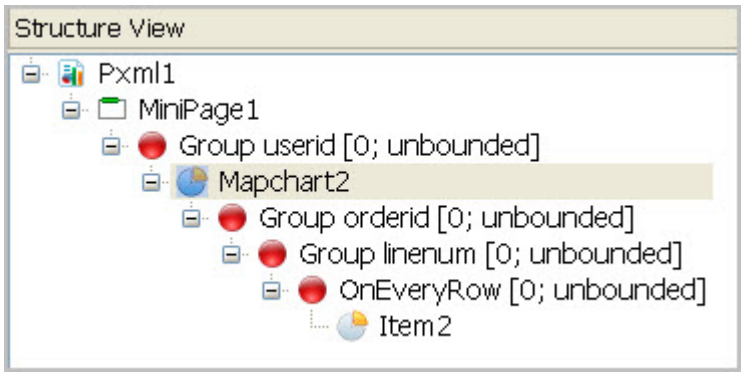


Figure 35: One chart for each unique userid

Custom keys

You can 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:

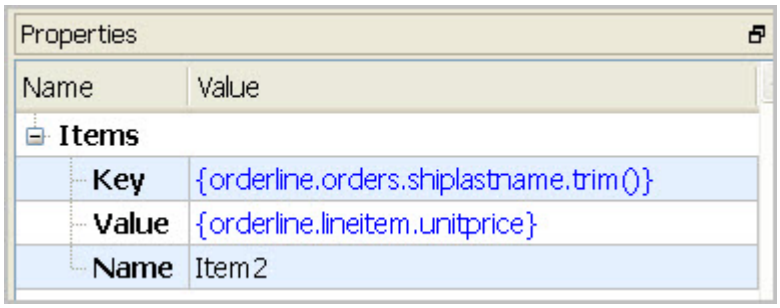


Figure 36: Properties panel for the chart Items (where key uses trimmed last name)

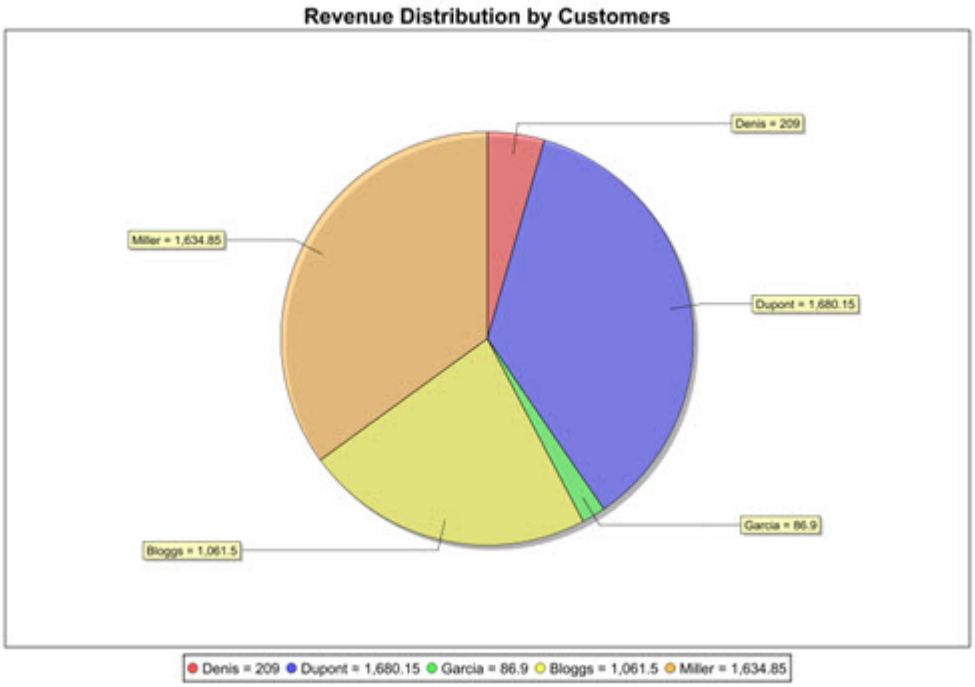


Figure 37: Chart showing Revenue Distribution by Customers (where key uses trimmed last name)

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:

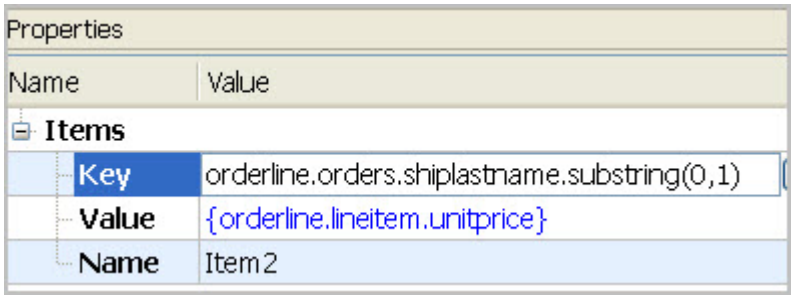


Figure 38: Properties panel for the chart Items (where key uses substring)

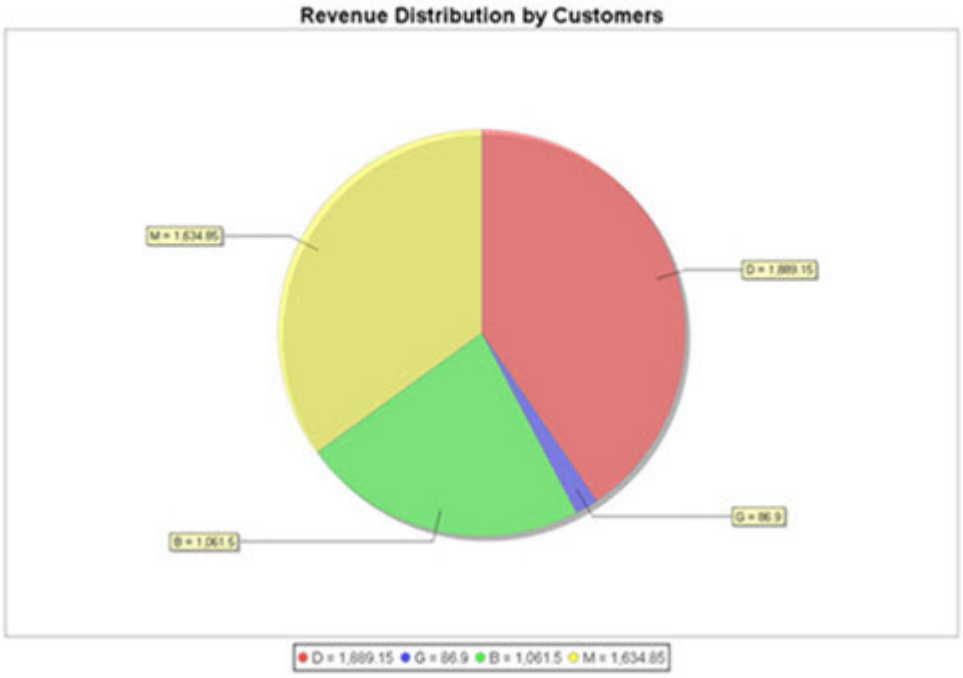


Figure 39: Revenue Distribution by Customers (where key uses substring)

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** (see Mapping of Chart properties).
- **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.

Working with Pivot Tables

The pivot table element defines a table element with fixed roles and types for its columns. Grouping, sorting and summarizing allows results to be displayed in different ways.

- What are pivot tables? on page 56
- Sample pivot table reports on page 59
- Create a static pivot table on page 60
- Pivot table properties on page 61
- Arrange your hierarchies on page 61
- Add a dimension on page 62
- Add a measure on page 62
- Pivot table elements and the Structure view on page 63

What are pivot tables?

The pivot table element is a table element with fixed roles and types for its columns, suitable for processing multi-dimensional data. Grouping, sorting, and summarizing are performed. 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. No other column types exist in a pivot table. A pivot table has one type of row: *fact rows*. 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. There are usually many rows with identical dimension values in a column. The dimensions can be viewed as forming a hierarchy. For this reason dimension can also referred to a *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, and so on.

For example, consider a table with the dimension columns "Country" and "Region". After sorting the data, several rows starting with {"Afghanistan","1 North",..} will be at the top, perhaps followed by some rows starting with {"Afghanistan","3 South",..} again followed by rows starting with {"Albania","1 North",..}. "Country" and "Region" form a hierarchy or tree where a country branch has sub branches for it's regions. The innermost dimension is said to contain the "facts" or "values" (meaning 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 charts

The pivot table is a generalization the chart objects. As an example one can say that a CATEGORYCHART is a PIVOTTABLE with two dimensions (The "categoryKey" and "key" attributes in the CATEGORYITEM element), one measure (the "value" attribute in the CATEGORYITEM element) on which a summarizing aggregation is performed for both dimensions. This table compares the different chart objects to the pivot table.

Table 7: Comparing chart objects and pivot tables

Element type	Number of dimensions	Number of measures	Number of aggregation groups	Aggregation functions	Sorting options
MAPCHART	One (specified by the key attribute)	One (specified by the value attribute)	One (values with the same key value are summarized)	Summarizing	By key, value and input order
CATEGORY CHART	Two (specified by the key and categoryKey attributes)	One (specified by the value attribute)	One (values with the same key + categoryKey value combination are summarized)	Summarizing	By keys, value and input order
XYCHART	None	Two (specified by the x and y attributes)	None	None	None
PIVOTTABLE	N (specified by HIERARCHY elements)	N (specified by MEASURE elements)	N (Aggregation can be performed on all dimensions)	Summarizing and others (such as count, average, maximum, minimum, and so on)	Input order and any combination of measures

Set data types

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.

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 88.

Sample pivot table reports

Two sample pivot table reports are provided with the installation of Genero Studio. One report shows a static pivot table, while the other shows a dynamic pivot table.

The reports are located in the `Reports.4pw` project. This project can be found in the `My Genero Files/samples/Reports` directory.

The reports are located in the non-generated sample applications, OrderReportCSharp and OrderReportJava. These projects can be accessed from the Welcome page, under the Tutorils and Samples tab. The source files are located in the `My Genero Report Writer Files/samples/Reports` directory.

Static pivot table sample report

The report name is `StaticPivotTable.4rp`.

This sample report produces a table of customer data, grouped by customers and orders. The input is presorted. The dimension columns, the userid and orderid, are populated accordingly.

Dynamic pivot table sample report

The report name is `DynamicPivotTable.4rp`.

When this report is selected, a second dialog opens. From this dialog, you select the dimensions and measures included in the report, along with how to sort the measures.

The form `pivotdialog.4fd` defines the dialog. The module `pivotdialog.4gl` contains the Genero code driving the dialog, making use of the pivot table library `libpivot.4gl`.

A new record of type “PivotControlBlock” is shipped in the FIRST PAGE HEADER of the report. The structure of the record:

```
TYPE PivotControlBlock RECORD
  title STRING,
  drawAs STRING,
  dimensionsDisplaySelection STRING,
  measuresDisplaySelection STRING,
  outputOrder STRING,
  topN INTEGER,
  displayFactRows BOOLEAN,
  displayRecurringValues BOOLEAN,
  computeAggregatesOnInnermostDimension BOOLEAN,
  computeTotal BOOLEAN,
  computeCount BOOLEAN,
```

```
  computeDistinctCount BOOLEAN,
  computeAverage BOOLEAN,
  computeMinimum BOOLEAN,
  computeMaximum BOOLEAN
END RECORD
```

The definition of the record is located in the file `globals.4gl`.

The record is populated by a call to the dialog function `promptForPivotDialogIfAny` (from `pivotdialog.4gl`) in `OrderReport.4gl` in the sample application. This is the code fragment:

```
CALL promptForPivotDialogIfAny(filename) RETURNING retval, controlBlock.*

IF NOT retval THEN
  RETURN NULL
END IF

IF NOT fgl_report_loadCurrentSettings(filename) THEN
  EXIT PROGRAM
END IF
```

The function inspects the `4rp` file. If it finds exactly one dynamically configurable pivot table, it will prompt the user to configure it (see `pivotdialog.4gl` for details).

For details about coding for a dynamic pivot table, examine the source code that comes with the sample application.

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"`.

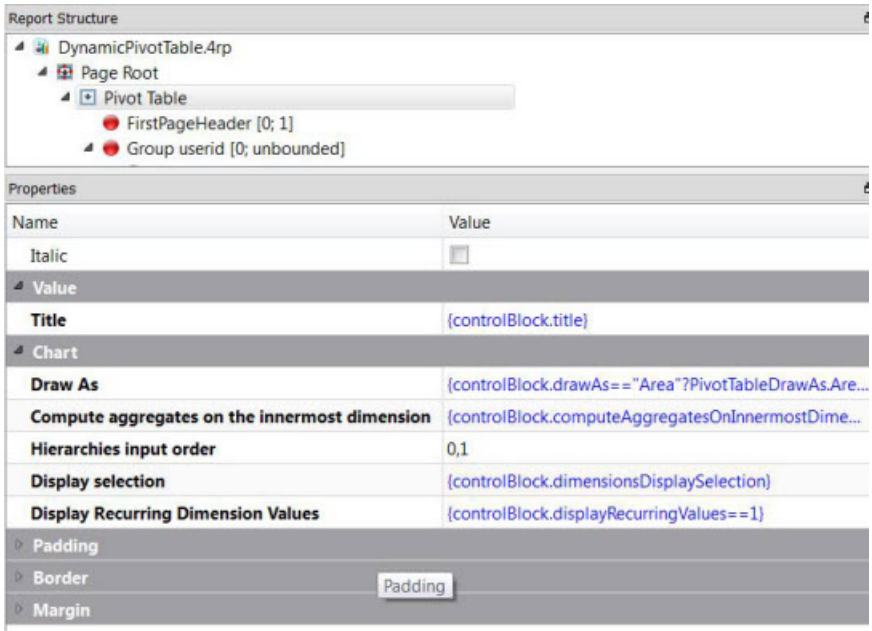


Figure 40: Properties of the Pivot Table element

This figure displays the values of the properties for the pivot table element.

Create a static pivot table

Follow this procedure to create a static pivot table.

1. Create a new, empty report.
2. In the Data View, associate your data schema.

3. From the Tool Box, add a Pivot Table to the report layout under the Page Root.
4. Add dimensions as Hierarchy elements. See Add a dimension on page 62
5. Add measures under the Fact node. See Add a measure on page 62.
6. Arrange the dimensions and measures in the Structure View. See Arrange your hierarchies on page 61.
7. Set additional properties as needed for all elements of the Pivot Table. See Pivot table properties on page 61

An example is provided in the Reports project (`Reports.4pw`), provided as part of the `samples` directory. The report is `StaticPivotTable.4rp`.

Pivot table properties

When you define a pivot table, you set properties specific to 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
- 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. In case that 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 so that if for example a pivot table with 4 dimensions and 3 measures is drawn as a category chart which has 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 Pivot Table: Pie, Pie3D, Ring, Bar, Bar3D, Area, StackedBar, StackedArea, Line, Line3D, Waterfall, Polar, Scatter, XYArea, XYStackedArea, XYLine, Step, StepArea, TimeSeries, Table

Arrange your hierarchies

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.

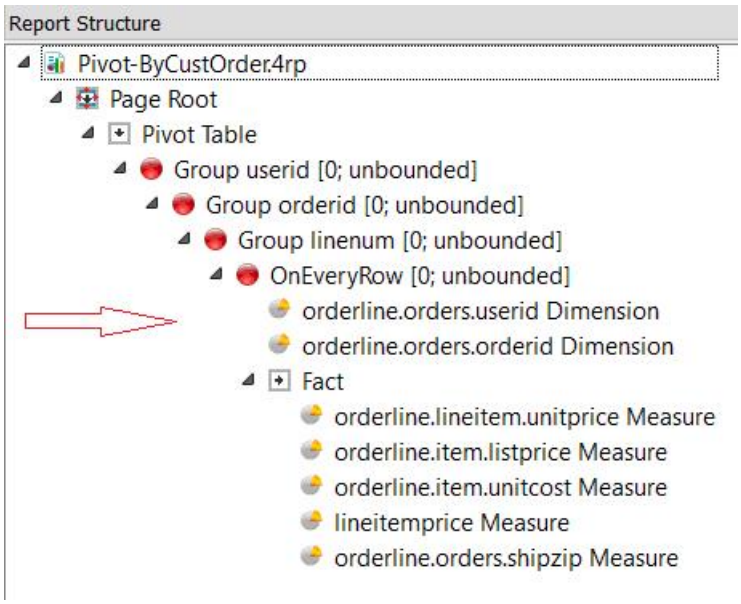


Figure 41: Flat shipping of dimension values

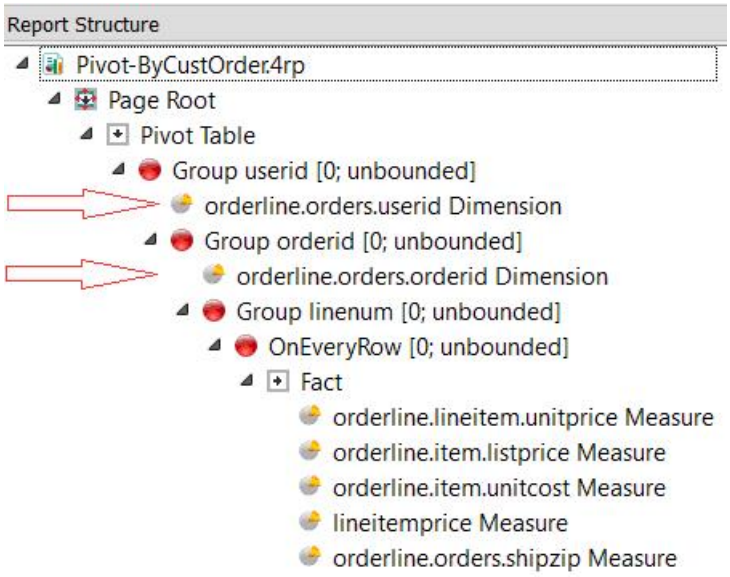


Figure 42: Optimized shipping of dimension values

Add a dimension

Dimensions are the values against which aggregation occurs.

- In the Value property, specify the column name for the dimension.
- Select the desired aggregation types. Compute Totals is selected by default.
- In the Title property, provide a title for the dimension.

Add a measure

Measures are placed under a Fact in the Structure View.

- In the Value property, specify the column name for the measure.
- If numeric, check the Numeric Column checkbox.
- In the Title property, provide a title for the dimension.

- If numeric, specify a format.

Pivot table elements and the Structure view

A static pivot table uses predefined dimensions and measures when creating the report.

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.

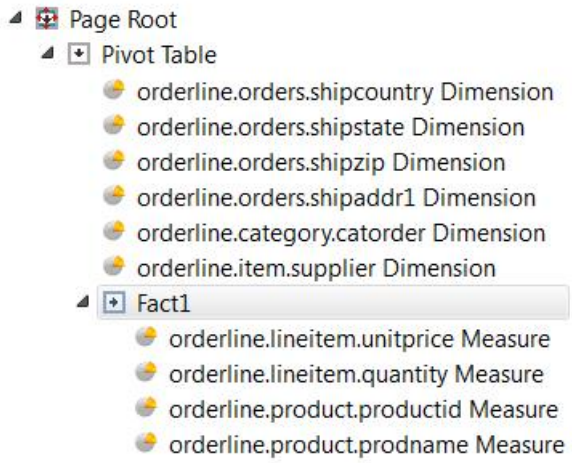


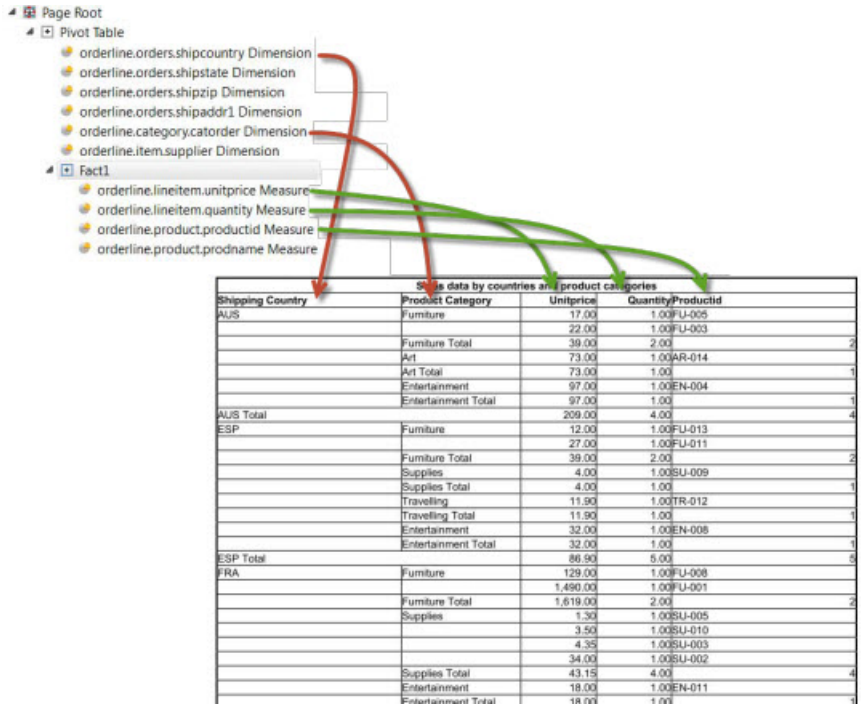
Figure 43: Pivot table elements in the Structure View

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.

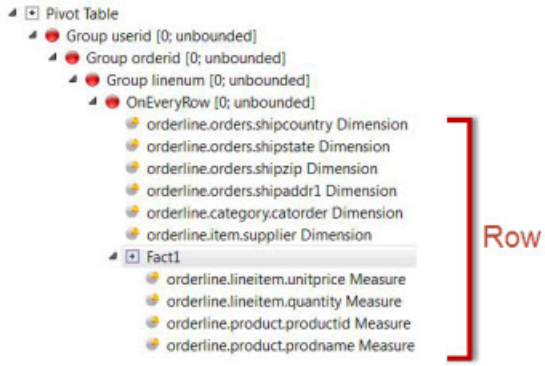


Shipping Country	Product Category	Unitprice	Quantity	Productid
AUS	Furniture	17.00	1.00	FU-005
		22.00	1.00	FU-003
	Furniture Total	39.00	2.00	2
	Art	73.00	1.00	AR-014
	Art Total	73.00	1.00	1
	Entertainment	97.00	1.00	EN-004
	Entertainment Total	97.00	1.00	1
AUS Total		209.00	4.00	4
ESP	Furniture	12.00	1.00	FU-013
		27.00	1.00	FU-011
	Furniture Total	39.00	2.00	2
	Supplies	4.00	1.00	SU-009
	Supplies Total	4.00	1.00	1
	Travelling	11.90	1.00	TR-012
	Travelling Total	11.90	1.00	1
	Entertainment	32.00	1.00	EN-008
	Entertainment Total	32.00	1.00	1
ESP Total		86.90	5.00	5
FRA	Furniture	129.00	1.00	FU-008
		1,490.00	1.00	FU-001
	Furniture Total	1,619.00	2.00	2
	Supplies	1.30	1.00	SU-005
		3.50	1.00	SU-010
		4.35	1.00	SU-003
		34.00	1.00	SU-002
	Supplies Total	43.15	4.00	4
	Entertainment	18.00	1.00	EN-011

Figure 44: 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.



Sales data by countries and product categories

Shipping Country	Product Category	Unitprice	Quantity	Productid
AUS	Furniture	17.00	1.00	FU-005
		22.00	1.00	FU-003
	Furniture Total	39.00	2.00	2
	Art	73.00	1.00	AR-014
	Art Total	73.00	1.00	1
	Entertainment	97.00	1.00	EN-004
	Entertainment Total	97.00	1.00	1
AUS Total		209.00	4.00	4
ESP		12.00		

Figure 45: Report row placed in a single trigger

It is allowed, but highly unusual, to specify the rows literally. All rows have to have exactly the same structure (number of dimensions and measures, types, and so on).

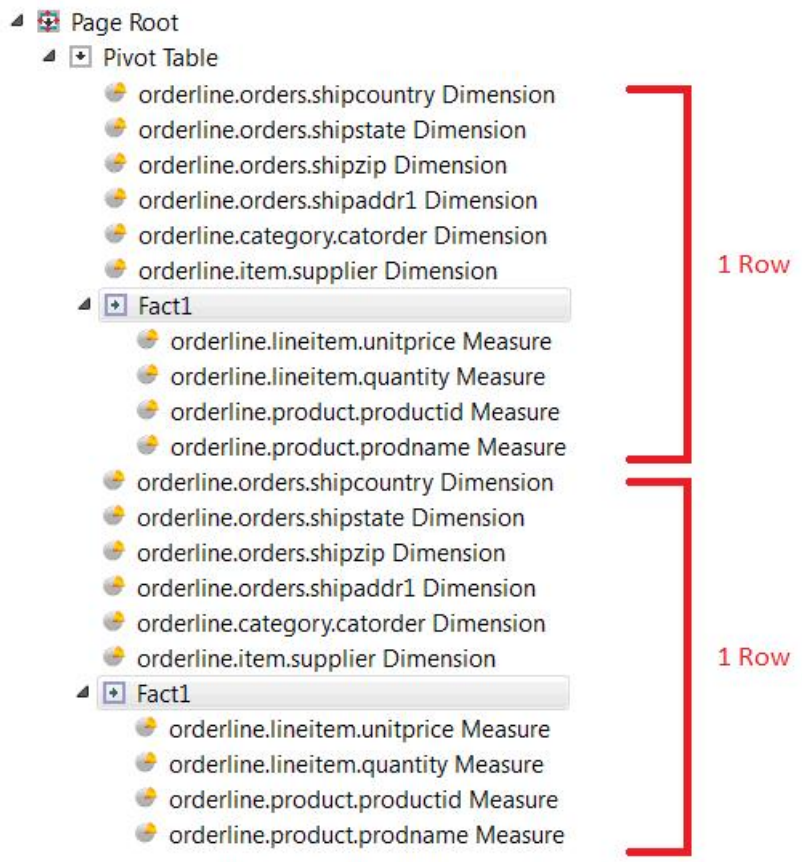


Figure 46: title

Expressions in properties

- Overview
- Using the RTL Expression Language
  - Operators
  - Conditional Expressions
  - Operands
  - 4GL Variables
  - Examples
- Using RTL Classes
  - Classes
  - Members
  - Examples
- Using the PXML Expression Language
  - Units of Measure
  - Variables
  - Functions
- Substituting 4GL Variables for Constants

See also: Dimension Resolver

Overview

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:

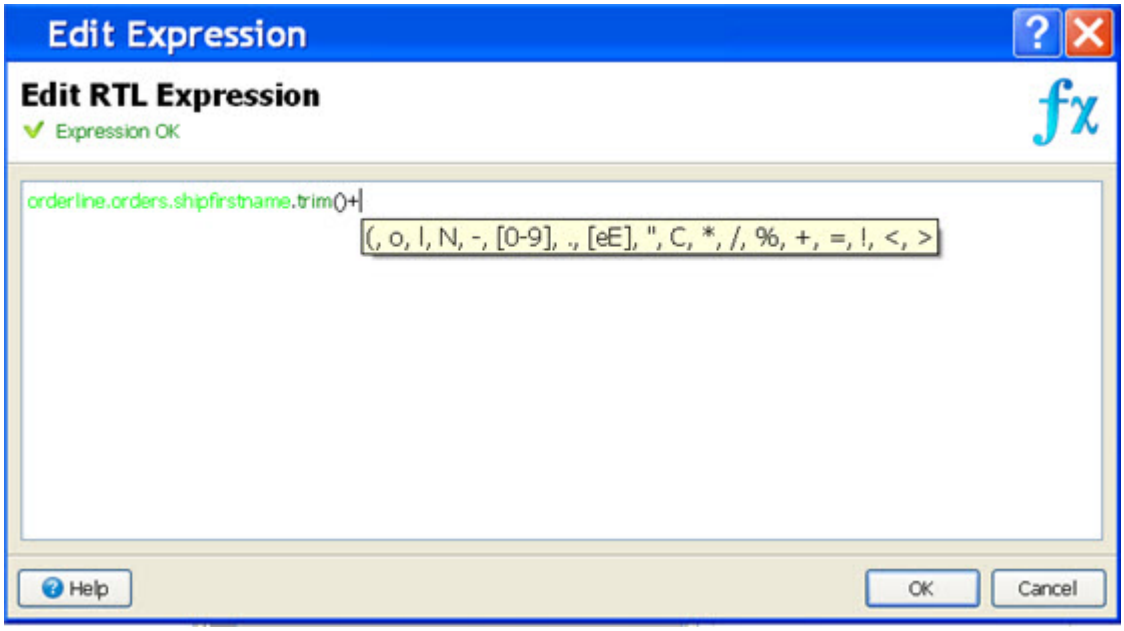


Figure 47: Edit Expression dialog

Press CTRL+SPACE and a Code Completion box appears, containing a list of the valid choices based on the context.

RTL Expressions

RTL Expressions:

- **are typed** - an expression is composed of items of different types and the expression returns a particular type. The properties in the Properties View are also typed. *Any expression entered as a value for a property must return the specified type*. For example, the text property of a WordBox has a type of String. So, any RTL expression for the text property is expected to yield a String. See Report Element Properties for a list of each report property and its type.
- **closely follow the Java™ syntax** for expressions. The main difference is that the "new" keyword is not supported; it is not possible to create and subclass objects. RTL Expressions loosely follow the Java™ evaluation semantics (operator precedence, evaluation order, and so on).
- **can use 4GL variablesvariables**. The variable is converted to a specific type within the expression (See Conversion table) .

RTL Classes provide member functions (methods) and member variables that you can use in your expressions. There is a class for each type of a report property.

> **Important:** Numeric data types are limited to 15 significant digits. To avoid problems with totals in reports, do not calculate the value of an item that is part of a total using an RTL expression. Perform any calculations in the BDL programreport application.

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 68
- Conditional Expressions on page 69
- Operands on page 69
- 4GL Variables on page 69
- XSD Variables on page 70

Operators

Table 8: RTL Operators

Operator	Description	Example	Precedence
`%`	Arithmetic: Modulus	`x % 2`	8
`*`	Multiplication	`x * y`	7
`/`	Division	`x / y`	7
`+`	Addition	`x + y`	7
`-`	Subtraction	`x - y`	6
`+`	Concatenation	`string + string`	5
`<`	Relational/Boolean: Less than	`numeric < 100`	4
`<=`	Less then or equal to	`numeric <= 100`	4
`>`	Greater than	`numeric > 100`	4
`>=`	Greater than or equal to	`numeric >= 100`	4
`==`	Equal to	`numeric == 100`	4
`!=`	Not equal to	`numeric <> 100`	4
`!`	Logical inverse (NOT)	`!(x = y )`	3
`&&`	Logical intersection (AND)	`expr1 && expr2`	2
`||`	Logical union (OR)	`expr1 || expr2`	1

The first column in the table describes the precedence order of the operators, listed highest to lowest. For example, the % modulus operator has a higher precedence than the * operator. Parentheses can be used to overwrite the precedence of operators.

Conditional Expressions

Conditional expressions allow you to express IF/ELSE statements.

Syntax:

```
Boolean-expression?expression-1:expression-2
```

The ? operator indicates that this expression is conditional; the return value is dependent on the result of the Boolean expression. If the Boolean expression is TRUE, the first expression is the return value; otherwise, the second expression is the return value.

You can use the `null` keyword in the ternary conditional operator. The “if then” and “if else” operands can be either expressions or the keyword `null`. A property whose RTL expression yields “null” is not set. This is useful in cases where a property should be set only when a certain condition is met. Consider the case where the background color of a WORDBOX should be set to red when a variable value *x* drops below a value of 10. The expression for this would be:

```
x<10?Color.RED:null
```

Operands

Operands include:

- Literal values
- Other expressions
- 4GL Variables
- RTL Class Members
  - Objects
  - Methods (returning a single value)

A literal value for a string in an expression should be delimited by double quotes: "Test".

4GL Variables

The data types of 4GL variables are taken into account when constructing expressions. For every 4GL variable an object is created that is either an instance of a FGLNumericVariable or an FGLString Variable. These objects hold the value of the 4GL variable, and at the same time they contain a member variable **value** which also contains the value. For this reason, it is legal to write "order_line.itemprice" in your expression as a shortcut for "order_line.itemprice.value". Both types of objects have these specific member variables defined:

- **value**- value of the 4GL variable
- **caption**- the title of the field
- **name**- the name of the variable
- **type** - the RTL type of the variable
- **isoValue**- the locale and formatting-independent representation of the value of the variable

The conversion table lists 4GL data types and the type into which they are converted within an RTL expression:

Table 9: 4GL data types and the type into which they are converted within an RTL expression

4GL type	Corresponding RTL type
CHAR, VARCHAR, STRING and TEXT	FGLStringVariable
DATE, DATETIME and INTERVAL	FGLNumericVariable

4GL type	Corresponding RTL type
INTEGER, SMALLINT, FLOAT, SMALLFLOAT, DECIMAL and MONEY	FGLNumericVariable, **limited to 15 significant digits**. The value of a number larger than 15 digits will be truncated, and the resulting number is rounded. For example, `12345678901234567` will be rounded to `123456789012346`.

Important: To avoid problems with inaccurate totals on a report due to rounding, do not perform RTL arithmetic on either the individual values or the total value; calculate the value of the item in the BDL program instead, before passing the value to the report.

XSD Variables

The data types of XSD variables are taken into account when constructing expressions. For every XSD variable an object is created that is either an instance of a Numeric or an String variable. These objects hold the value of the XSD variable, and at the same time they contain a member variable **value** which also contains the value. For this reason, it is legal to write "order_line.itemprice" in your expression as a shortcut for "order_line.itemprice.value". Both types of objects have only a single member variable defined:

- **value**- value of the XSD variable

The conversion table lists the XSD data types and the type into which they are converted within an RTL expression:

Table 10: XSD data types and the type into which they are converted within an RTL expression

XSD type	Corresponding RTL type
string, duration, dateTime, time, date, gYearMonth, gYear, gMonthDay, gDay, gMonth, hexBinary, base64Binary, anyURI, QName, and NOTATION	String
boolean, decimal, float, double, duration	Numeric

Important: To avoid problems with inaccurate totals on a report due to rounding, do not perform RTL arithmetic on either the individual values or the total value; calculate the value of the item in the report program instead, before passing the value to the report.

Examples

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

1. To add 10% to the itemprice: `order.itemprice*1.10`

   The data item order_line.itemprice is converted to a Numeric type, so we can use the Numeric operators. In order to display the result of a Numeric expression in a WordBox, we must convert the result to a String. See Example 1 in the Using RTL Classes section.
2. Let's add 10% to the item price conditionally, depending on the value: `order.itemprice<100? order.itemprice*1.10:order.itemprice`

   The condition in this Boolean expression tests whether the itemprice is greater than 100; if so, the value returned is 110% of the itemprice; otherwise, the value returned is simply the itemprice.
3. To set the font of a report item to italic when the 4GL variable **order_line.lineitemprice** exceeds $20, we must create an expression for the fontItalic property: `order.lineitemprice>20`

   The property fontItalic is of type boolean, so any RTL expression that we use for that property must return a boolean value (TRUE/FALSE). Any of the relational operators yields a boolean, so the type of the returned value of this expression is a boolean (The expression will return TRUE if the lineitemprice exceeds 20).

Note: A numeric value by itself is not a boolean value as it is in some programming languages.

Using RTL classes

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 Class - a class representing a Date value; contains methods for parsing and formatting strings.

Members

Instance Member Methods

Instance Member methods are called on an object instance. You can get an object instance by referencing a 4GL variablevariable or by calling a method on another object. You can also use a literal value as an object instance.

When you invoke the method, it is prefixed with the object instance name and the "." character.

Examples of instance methods are expressions like "`order_line.customer_name.trim()`". This is valid because the 4GL variablevariable `order_line.customer_name` has a CHAR data typestring 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:

```
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:

```
(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:

```
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:

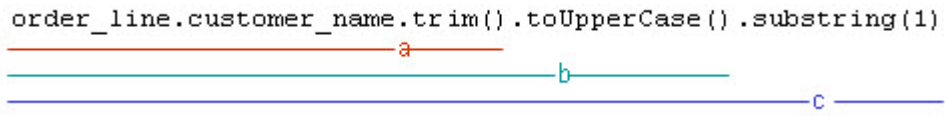


Figure 48: order_line.customer_name.trim().toUpperCas().substring(1)

   In this example, the object **order_line.customer_name** is a BDL CHAR variablestring variable; this variable is assigned to the String type. The String method **trim()** is called first, returning the String object a. The method **toUpperCase()** is called for the object a, returning object b which will be in upper case. Finally, the method **substring()** is called for the object b, returning object c. If the customer_name is "Springs", the resulting object c is the string "PRINGS".

   There are many additional examples of expressions in the properties of report elements defined in the `4rp` programs that are part of the GRWDemo projectdemo 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**

See Dimension Resolver for additional explanations and examples of the units that can be used.

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

For example, to center an element in its parent container you can use the max variable for these properties:

Table 11: Centering an element

Property	Value
x	max/2
y	max/2
anchorx	0.5
anchory	0.5

You can use the **rest** variable in the Y-Size property of a MiniPage container to force a page break by consuming the remainder of the container:

Table 12: Using rest

Property	Value
Y-Size	rest

Functions

The PXML Expression language has a 4GL-like syntax. The most commonly used functions are:

- **max(valueA, valueB)** - this is a function, not the variable listed in Variables on page 73!
- **min(valueA, valueB)**
- **length(value)**
- **width(value)**

For example, this expression uses the functions **max** and **width** :

```
max(10cm,width("HELLO"))
```

In this example, the report engine first calculates the width of the string "HELLO", taking the current font metrics into account. It then determines which is larger (10cm or the calculated width of "HELLO") and returns the larger value.

Substituting 4GL variables for constants

You may want your expression to depend on a data variable rather than on the 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 a 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({order_line.titlewidth}cm,width({order_line.title}))
```

For this expression to be legal, the variable **order_line.titlewidth** has to be of type Numeric, and the variable **order_line.title** has to be of type String.

> **Note:** You cannot construct a dynamic expression where the function names (such as `max` or `width`) or the unit names (such as `cm`) are dynamic.

Expression examples

Examples of common expressions used by report writers.

Check a field for a value

You want to check whether a field contains a value, is empty, or is null and act accordingly.

For this example, the data source includes the field `orderline.order.contact`, and the field is a `STRING`. The field either contains a value, is an empty string, or is null.

To test whether it contains a value, you write an expression:

```
orderline.order.contact.trim().length()>0
```

This expression will evaluate to either TRUE or FALSE.

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

```
orderline.order.contact.trim().length()==0
```

```
orderline.order.contact.isEmpty()==TRUE
```

To explicitly test for null:

```
orderline.order.contact.isNull()==TRUE
```

Use in the Visibility Condition property

Expressions that evaluate to TRUE or FALSE can be used in the **Visibility Condition** (`visibilityCondition`) property. If the expression evaluates as TRUE, then the instance of the element will appear in the report. If the expression evaluates as FALSE, the instance of the element (to include all its children, i.e. the entire element tree) is removed from the report. If you are using relative positioning, all sibling elements after this element in the report structure shift accordingly, reclaiming the space that the element would have occupied.

Use in the Text property

You can use these expressions when defining the **Text** (`text`) property. The `text` property specifies the text to be drawn.

```
orderline.orders.contact.trim().length()>0?orderline.orders.contact:""
```

In this expression, the expression evaluates to TRUE when the length of the trimmed field is greater than zero and the field value is printed (to include any leading or trailing spaces). If the length is not greater than zero, the field is identified as not having a value and an empty string is printed; the vertical allocated space for that field remains in the report.

An alternate expression could simply be:

```
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.
>
> ```
> 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
> ```

Report Designer Reference

- Report Design Elements (The Toolbox) on page 76
- Report Element Properties on page 90
- Bar Codes on page 119
- RTL Classes Overview on page 156
- Dimension Resolver on page 157

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 79
  - Decimal Format Box on page 79
  - Page Number (PageNoBox) on page 81
  - Image Box on page 81
  - Table on page 83
- Business Graphs
  - Map Chart on page 85
  - Map Chart Item on page 85
  - Category Chart on page 86
  - Category Chart Item on page 86
  - XY Chart on page 86
  - XY Chart Item on page 87
  - Pivot Table on page 87
  - Pivot Table Hierarchy Value on page 88
  - Pivot Table Fact on page 88
  - Pivot Table Measure on page 88
- References
  - Reference Box on page 89
  - Info Node on page 89
- Bar Codes
  - Bar Code Box on page 89

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:

marginWidth, marginRightWidth, marginBottomWidth, marginLeftWidth, marginTopWidth, borderWidth, borderRightWidth, borderBottomWidth, borderLeftWidth, borderTopWidth, paddingWidth, paddingRightWidth. paddingBottomWidth, paddingLeftWidth, paddingTopWidth, borderStyle, borderRightStyle, borderBottomStyle. borderLeftStyle, borderTopStyle, borderColor, borderRightColor, borderBottomColor, borderLeftColor, borderTopColor, roundedCorners

The x-Size and y-Size properties specify the INNER size of the box. Adding borders, for example, will increase 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 swapX (Swap X) on page 110property) 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

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 will resize 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 will use 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. You can change the object's default appearance by setting the values of its properties.

The text property specifies the string to be displayed in the WordBox or Word Wrap Box.

You can set the textAligment 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'll prevent that automatic enlargement.
- The text value of these boxes can be edited directly in the report design document; double-click the object and the input cursor will be placed in the text. The layout will be updated on each keystroke.
- To force a line break, use the newline character "\n".
- Beginning in version 2.4x, a Word Wrap Box can span pages; when the available vertical space for WORDWRAPBOX is not sufficient to display the entire text, the box now propagates the exceeding content to additional boxes, with the same behavior as a propagating MINIPAGE.

Decimal Format Box
Use a Decimal Format Box for 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 a variety of features designed to make it possible to parse and format numbers in any locale, including support for Western, Arabic, and Indic digits. It also 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. You can change the object's default appearance by setting the values of its properties.

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 will be placed in the text. The layout will be 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.

Use the Location property to specify the file name and path of the document whose image is to be displayed. Press the ... button to open a dialog and select the html file.

Note: The content of an HTML box cannot span pages.

Embedding HTML

To embed an image, we use a URL type that allows encoding the data in the body of the URL text. Such a URL starts with the protocol name “data” and a short description of the data, followed by a colon and the encoded data itself. The full syntax of data URLs is:

```
data:[<MIME-type>][;charset=<encoding>][;base64],<data>
```

See 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 utf8. Image data is always encoded in base64 while other data such as HTML content is typically “Percent encoded”.

For HTML content a URL would have the form “`data:text/html,<data>`. To embed HTML, we use the data protocol syntax in the Location property of the HTMLBOX element. The syntax is:

```
data:text/html, followed by the percent encoded data of the html document
```

To automatically construct this URL for HTML documents, press the **... button** for the Location property. When the Open dialog displays, select the HTML file located in your file structure, and check the **Embed in document** checkbox at the bottom of the dialog:

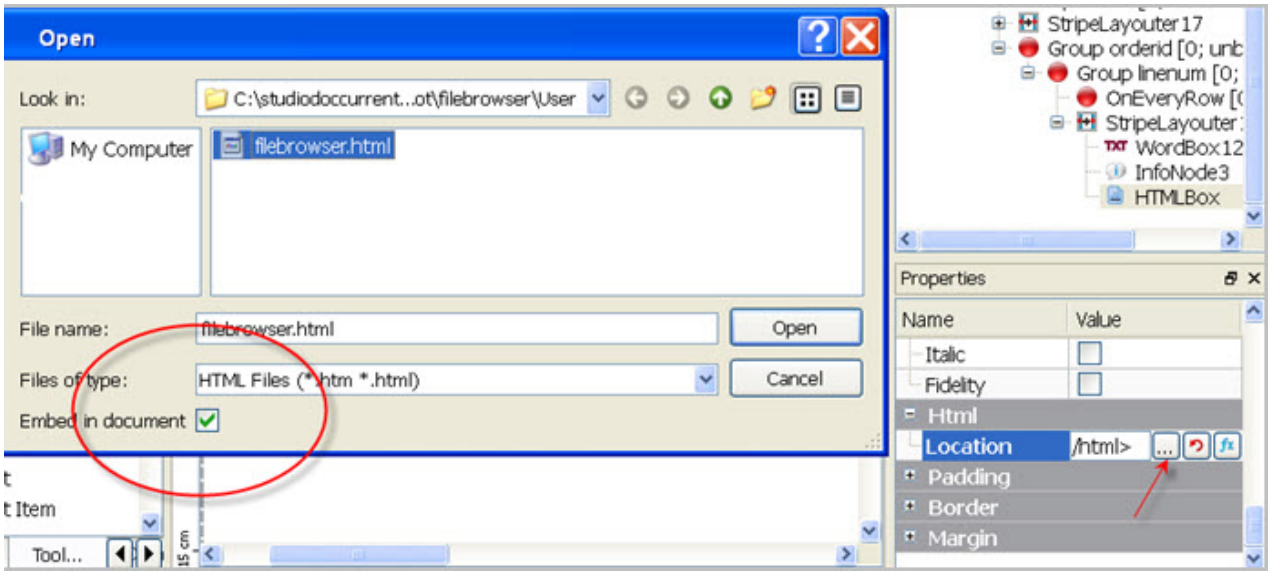


Figure 49: Embed in document checkbox

Populating HTML content from text variables

Data in text variables was typically input into a database via a TextEdit form field, with the textFormat style attribute set to **html**. See the Presentation Styles topic in the *Genero Business Development Language User Guide* for more details.

Press the **formula button** for the Location property of the HTML Box, and enter the expression, including the name of the text variable.

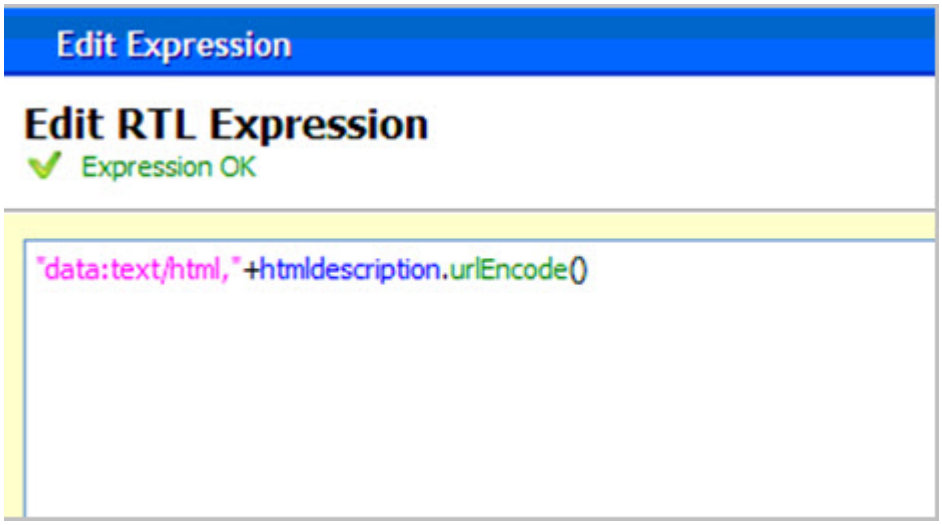


Figure 50: RTL Expression

The function String.urlEncode() is used to encode the data using percent encoding.

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.

The Location property is used to specify the html content.

Page Number (PageNoBox)
Use this object for 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 be specified to identify the logical page.

Specific functions are available to 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 Calculating the page number string.

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 PageNoBox:

pageName, pageNoOffset, pageNoFormat, textAlignment, textExpression

Additional properties are inherited from Word Box.

Image Box
Use this object for images.

Image Box (ImageBox type) is a layout container, found in the Drawable group in the Tool Box view.

Drawable: Use this object for images. On all platforms GIF, JPEG, PNG, BMP, WBMP, and SVG formats are supported. The images are cached on a per document basis. The default resolution is the current screen resolution.

Use the Location property to specify the location of the image to be rendered. Location values are URLs supporting the protocols "http", "file" and "data". The URL can be:

- absolute - Press the ... button to open a dialog and select the image located in your file structure; this populates the file name and path of the image to be rendered.
- variable - using an RTL expression to provide the image name as a variable. For example: `./images/database/"+orderline.product.prodpic.trim()`

If you want to preserve the aspect ratio of an image, set the value of either length (y-size) or width (x-size), allowing Report Writer to calculate the corresponding value. If you set both properties, the resulting image will appear distorted.

SVG images - Using SVG images instead of bitmap images can substantially reduce the document size, which is particularly useful when PDF documents are produced. When providing the SVG content from a 4GL variable, use the mime type "image/svg" so that the url looks something like "..." when read from a BLOB. The currently supported SVG version is "1.2 Tiny" (See http://www.w3.org/TR/SVGTiny12).

Embedding images

To embed an image, we use a URL type that allows encoding the data in the body of the URL text. Such a URL starts with the protocol name “data” and a short description of the data, followed by a colon and the encoded data itself. The full syntax of data URLs is:

`data:[<MIME-type>][;charset=<encoding>][;base64],<data>`

See 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 utf8. Image data is always encoded in base64 while other data such as HTML content is typically “Percent encoded”.

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:

```
 followed by the base 64 encoded bitmap data of the
 image
```

To automatically construct this URL for an image, press the **... button** for the Location property. When the Open dialog displays, select the image file located in your file structure, and check the **Embed in document** box at the bottom of the dialog:

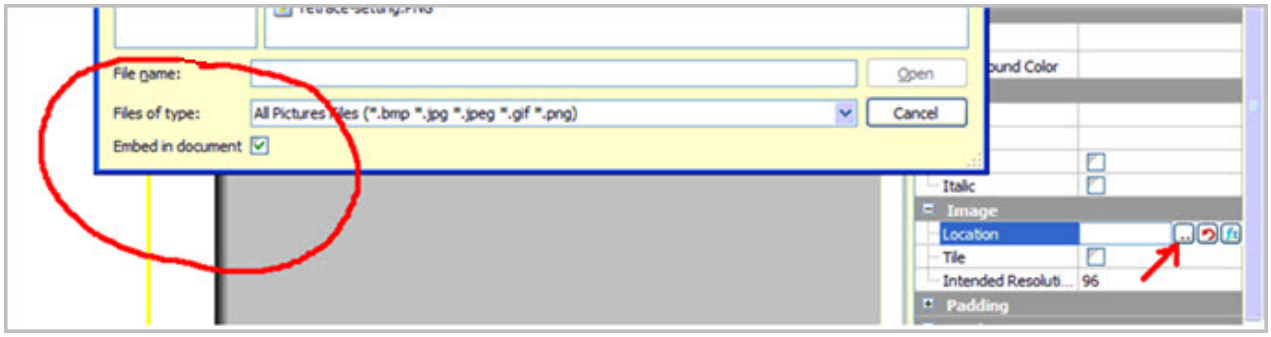


Figure 51: Embed in document checkbox

Populating images from blob variables

If you drag a BYTE variable from the Data View onto the Report Design, an Image Box object will be created. If the variable was named "imageblob" then the Location property would automatically be filled with this formula:

```
"+imageblob
```

where **imageblob** is the name of the blob

This has the same effect as pressing the **formula button** for the Location property, and entering the formula including the blob variable name.

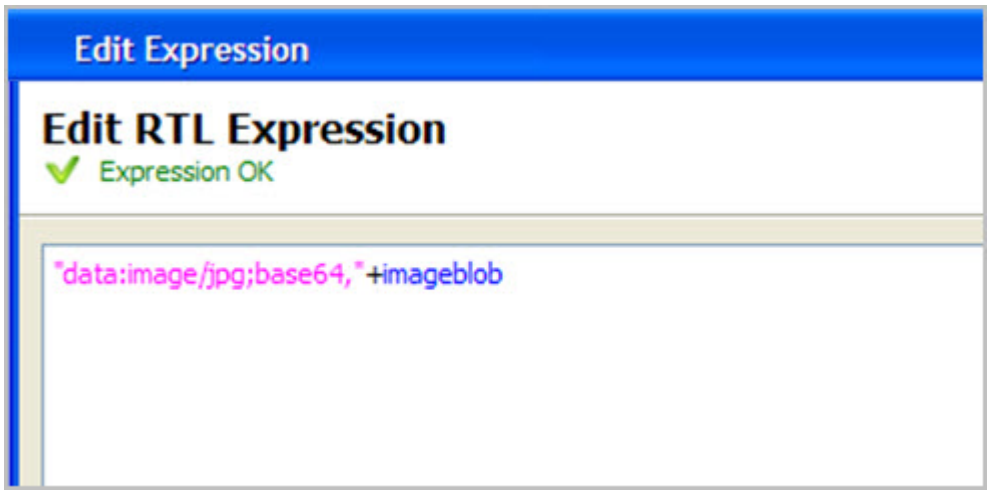


Figure 52: Entering blob variable name in RTL Expression editor

> **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 itself to determine the image type. This formula would work for blobs that were of **png** type also.

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.

The Location property is used to specify the image.

These properties are specific to Image Box:

Intended Resolution, Location, Fill

> **Note:** Beginning with version 2.4x, the **tile** property is replaced by a new property, fill. If you open a document containing the tile property, it will be automatically replaced by fill.

Fallback Image

A *fallback image* is the image to be 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; it resolves relative to the location of the form design (`4rp`) file.

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.

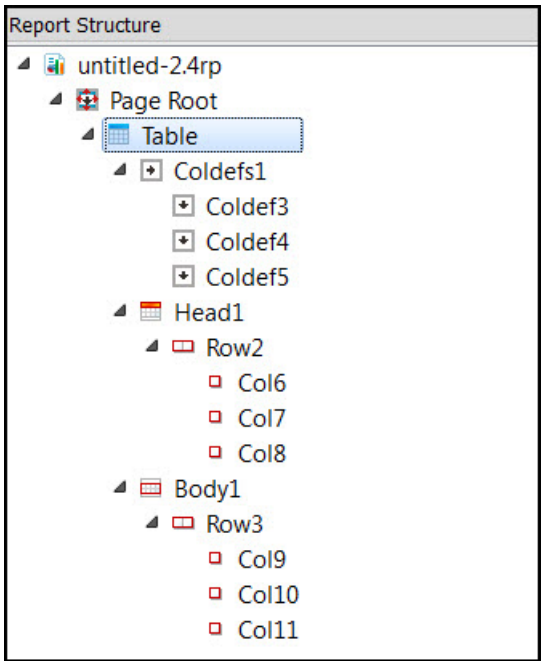


Figure 53: Table Object as viewed in the Report Structure

The top-level element is the Table element. It contains three child elements, which make up the parts of the table. These elements are the column definitions, the head, and the body.

The *column definitions* (or *coldefs*) define the basic properties for the columns in this table object. These properties include settings for padding, width, and alignment.

The head section contains the heading rows.

The body section contains the body rows.

Table Properties

Properties specific to the table involve rules, borders, and padding.

A *rule* refers to a line that separates two rows or two columns. Rule-related properties include Rule, Rule Color, Horizontal Rule, and Vertical Rule.

The *border* refers to the border around the table. Border-related properties include Border, Border Color, Top Border, Left Border, Bottom Border, and Right Border.

Padding refers to the space between a cell boundary and the value contained within. Padding-related properties include Padding, Horizontal Padding, and Vertical Padding.

Column Properties

Column properties are specific to the column selected. Any column property set overrides the same property set for the table.

You can set the padding for the cells of a column. Padding-related properties include Padding, Horizontal Padding, and Vertical Padding.

You can set the width of a column using Proportional Width or Fix Width.

You can set the alignment of a value within the cells of a column with the Horizontal Alignment and Vertical Alignment properties.

Cell Properties

Cell properties are specific to the cell selected. Any cell property set overrides the same property set for the table or the column.

You can set the padding for the cells of a column. Padding-related properties include Padding, Horizontal Padding, and Vertical Padding.

You can set the alignment of a value within the cells of a column with the Horizontal Alignment and Vertical Alignment properties.

You can merge cells by setting the Column Span property.

Demos

The **Reports** demo project includes two report design documents showing reports that include a table object.

- The `TableDemo.4rp` shows a report design document with a simple table containing five columns and two rows. One row is the **Any Page Header** row, while the other row is the **Body** row.
- The `GroupedTableDemo.4rp` shows a report design document where the table is more complex, with several header and body rows. For each header row, the section property specifies whether it is the First Page Header, Any Page Header, and so on. Each body row is created with a purpose - to show a row of data, to show the sum of a group of rows, and so on. Some of the cells in the summary rows span columns. Triggers are used to determine when each of the body rows is output to the table in the report.

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 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.

Map Chart (Mapchart type) is a layout container, found in the Business Graphs group in the Tool Box view.

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

The valid values of **drawAs** for this object are: Pie|Pie3D|Ring|Bar|Bar3D|Table|SortedTable| AggregatedTable. The default is a Pie.

The **fidelity** property applies only if the chart is drawn as a table (drawAs="Table").

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

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

The valid values of **drawAs** for this object are:

Bar|Bar3D|Area|StackedBar|StackedArea|Line|

Line3D|Waterfall|Table|SortedTable|AggregatedTable. The default is a Bar.

The **fidelity** property applies only if the chart is drawn as a table (drawAs="Table").

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.

category Key, key, value

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

Value values of **drawAs** for this object are: Polar|Scatter|Area|Line|Step|StepArea|TimeSeries|Table|SortedTable. The default is Line.

The **fidelity** property applies only if the chart is drawn as a table (drawAs="Table").

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 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

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 will expand 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 visualitzation types (such at Tables), the enclosing containers should support propagation.

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

In addition to the attributes available for LAYOUTNODE, the PIVOTTABLE element has the following properties:

- title
- drawAs

  The **drawAs** property specifies the type of output that is rendered from the data. Depending on the type selected and the number of available dimensions, the rendering is delegated to the map chart, category chart, XY chart or table element. In case that 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 so that if for example a pivot table with 4 dimensions and 3 measures is drawn as a category chart which has 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.

  The valid values of **drawAs** for this object are: Area | Bar | Bar3D | Line | Line3D | Pie | Pie3D | Polar | Ring | Scatter | 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

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.

- dataType (default String)
- title (default is the empty string)
- format (default ---,---,---,--&.&&)
- value
- enumValues
- computeTotal
- computeCount
- computeDistinctCount
- computeAverage
- computeMinimum
- computeMaximum

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

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.

- dataType (default String)
- title (default is the empty string)

- format (default ---,---,---,--&.&&)
- value

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 like "Total from previous pages: num".

Reference Box (ReferenceBox type) is a layout container, found in the References group in the Tool Box view.

This object allows you to create layout-dependent text output like "Total from previous pages: num".

Since the space to be allocated may not be known until the report is run, make sure that 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.

See DesignHowTo for an example, and the demo programs provided with the product.

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.

See DesignHowTo for an example of this use, and the demo programs provided with the product.

Bar Codes

The Bar Codes section of the toolbox contains barcode objects

Bar Code Box

The Bar Code Box displays bar codes.

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. Example:

Figure 54: Sample Barcode

The currently supported types are listed in the topic Bar Code type listing on page 120. 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 Barcode 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.

See Bar Codes for a description of all the possible bar code types.

Report Element Properties

Each element 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.

See Using Expressions in Properties.

- General Properties on page 90
- Properties related to margins and borders on page 115
- Properties for Report Document Metadata on page 119

General Properties

General properties of a report element.

- alignment (Alignment) on page 93
- anchorX (Anchor X) on page 93
- anchorY (Anchor Y) on page 93
- baselineType (Baseline Type) on page 93

- bBorder (Bottom Border) on page 94
- bgColor (Background Color) on page 94
- border (Border) on page 94
- categoryKey (Category Key) on page 94
- categoryTitle (Categories Title) on page 94
- check (Check) on page 94
- class (Class) on page 95
- clip (Clip) on page 95
- codeType (Code Type) on page 95
- codeValue (Code Value) on page 95
- codeValueExpression (Code Value Expression) on page 95
- color (Color) on page 96
- colspan (Column Span) on page 96
- computeAggregatesInnermostDimension (Compute aggregates on the innermost dimension) on page 96
- computeAverage (Compute Average) on page 96
- computeCount (Compute Count) on page 97
- computeDistinctCount (Compute Distinct Count) on page 97
- computeMaximum (Compute Maximum) on page 97
- computeMinimum (Compute Minimum) on page 97
- computeTotal (Compute Totals) on page 97
- controlCharacters (Control Characters) on page 97
- dataSymbolsPerLine (Data Symbols per Line) on page 98
- displayFactRows (Display Fact Rows) on page 98
- displayRecurringDimensions (Display Recurring Dimension Values) on page 98
- displaySelection (Display Selection) on page 98
- drawAs (Draw As) on page 99
- drawLabels (Draw Labels) on page 99
- drawLegend (Draw Legend) on page 99
- encoding (Encoding) on page 99
- enumValues (Enum Values) on page 100
- errorCorrectionDegree (Error Correction Degree) on page 100
- fidelity (Text Fidelity) on page 100
- fill (Fill) on page 100
- floatingBehavior (Floating Behavior) on page 101
- fontBold (Bold) on page 101
- fontItalic (Italic) on page 101
- fontName (Name) on page 101
- fontSize (Size) on page 101
- format (Format) on page 101
- fWidth (Fix Width) on page 102
- hAlign (Horizontal Alignment) on page 102
- hidePageHeaderOnLastPage (Hide Page Header On Last Page) on page 102
- hidePageFooterOnLastPage (Hide Page Footer On Last Page) on page 102
- hierarchiesInputOrder (Hierarchies input order) on page 102
- href (href) on page 103
- hPadding (Horizontal Padding) on page 103
- hRule (Horizontal Rule) on page 103
- id (id) on page 103
- indent (Indent) on page 103

- intendedResolution (Intended Resolution) on page 103
- key (Key) on page 104
- keysTitle (Keys Title) on page 104
- layoutDirection (Layout Direction) on page 104
- localizeText (Localize Text) on page 105
- location (Location) on page 105
- name (Name) on page 105
- noCheckDigits (Number Check Digits) on page 106
- noDigits (Number Digits) on page 106
- noText (Hide Text) on page 106
- outputOrder (Output Order) on page 106
- padding (Padding) on page 106
- pageName (Name) on page 106
- pageNoOffset (Offset) on page 107
- pageNoFormat (Format) on page 107
- preferRectangularSymbols (Prefer Rectangular Symbols) on page 107
- pWidth (Proportional Width) on page 107
- rawCodeValue (Raw Code Value) on page 107
- rBorder (Right Border) on page 107
- referenceDefault (Default) on page 108
- referenceName (InfoNode Name) on page 108
- rule (Rule) on page 108
- ruleColor (Rule Color) on page 108
- section (Section) on page 108
- scaleX (Scale X) on page 109
- scaleY (Scale Y) on page 109
- seriesTitle (Series Title) on page 109
- smartParse (Smart Parse) on page 109
- sortAscending (Sort Ascending) on page 110
- sortBy (Sort By) on page 110
- splitOversizedItem (Split Oversized Items) on page 110
- strikethrough (Strikethrough) on page 110
- swapX (Swap X) on page 110
- text (Text) on page 111
- textAlignment (Text Alignment) on page 111
- textExpression (Text Expression) on page 111
- thinToGapRelation (Thin To Gap Relation) on page 111
- thinToThickRelation (Thin To Thick Relation) on page 112
- tile - replaced by fill (Fill) on page 100
- title (Title) on page 112
- topN (Top N) on page 112
- trimText (Trim Text) on page 112
- objectType (Type) on page 113
- underline (Underline) on page 113
- URL (Location) on page 113
- vAlign (Vertical Alignment) on page 113
- value (Value) on page 113
- valuesTitle (Values Title) on page 113
- visibilityCondition (Visibility Condition) on page 114
- vPadding (Vertical Padding) on page 114

- vRule (Vertical Rule) on page 114
- x (X) on page 114
- xAxisTitle (xAxisTitle) on page 114
- X-Size (X-Size) on page 114
- X-Size Adjustment (X-Size Adjustment) on page 114
- y (Y) on page 115
- yAxisTitle (yAxisTitle) on page 115
- Y-Size (Y-Size) on page 115
- Y-Size Adjustment (Y-Size Adjustment) on page 115

alignment (Alignment)
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** - shortcust for x = max/2, anchorX = 0.5, centered in the parent container
- **baseline** - uses baseline alignment

The default value is **none**.

See: Placing Elements on the Report Page, Align Numbers, Center Elements

anchorX (Anchor X)
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)
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)
Specify which baseline of this report element should be linked to which baseline of a preceding element.

In the Properties view, this property is the **Baseline Type** property in the **Layout** category.

Provides additional information for **baseline** alignment, to specify which baseline of this report element should be linked to which baseline of a preceding element that also has the property alignment set to **baseline**. This property is relevant only if alignment for the report element is set to **baseline**.

Type Enum, choices are:

- leftleft - the report element and its preceding element will be aligned along their left baselines
- leftright - the left baseline of the report element will be aligned with the right baseline of the preceding element
- rightleft - the right baseline of the report element will be aligned with the left baseline of the preceding element
- rightright - the report element and its preceding element will be aligned along their right baselines

bBorder (Bottom Border)
The `bBorder` property sets the weight of the bottom border of a table object.

In the Properties view, this property is the **Bottom Border** property in the **Table** category.

The `bBorder` property overrides the more general border (Border) on page 94 property for a table object.

Type: PXML, point value.

Default value: None.

bgColor (Background Color)
Sets the background color for this 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

categoryKey (Category Key)
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)
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.

check (Check)
Check 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**.

class (Class)
Specifies one or more classes for a report element.

In the Properties view, this property is the **Class** property in the **Object** category.

Class names that are prefaced with the string "grw" are internally meaningful to the Genero Report Writer. The class property is available for all PXML nodes.

Example of class names with a "grw" prefix:

- grwTableHeader
- grwTableRow
- grwHeadlessTableRow
- grwTableNumericColumnValue
- grwStringValue

Type **String** (space separated list of identifiers).

clip (Clip)
Option to clip the report object box and its content along the sides.

In the Properties view, this property is the **Clip** property in the **Layout** category.

This applies to all layout nodes.

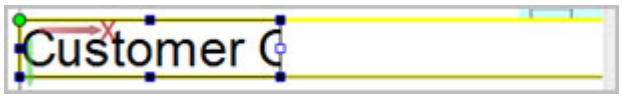


Figure 55: Clipped field

Type **Boolean**: The default value is **false**.

codeType (Code Type)
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.

The currently supported types are: U PC-A, UPC-E, UPC Supplementals 2 and 5, EAN-13, EAN-8, EAN Supplementals 2 and 5, Code 128 (EAN 128), 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, Code 93, Code 93 extended, Codabar 18 and Codabar 2.

See Bar Codes for additional information.

Type: Enum.

codeValue (Code Value)
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. See Bar Codes for complete information.

codeValueExpression (Code Value Expression)

In the Properties view, this property is the **Code Value Expression** property in the **Bar Code** category.

Property of the BARCODEBOX, 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 which is computed from the expression should be sufficiently exact.

Type: PXML.

If a value for this property, or for the X-Size property, is not set, a default length is used in calculating the page number string from these functions.

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.

See " Page N of M" examples.

color (Color)
Sets the paint color for this node and for all children of this node.

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 is set 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.

computeAggregatesInnermostDimension (Compute aggregates on the innermost dimension)
Specifies whether or not 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

computeAverage (Compute Average)
Specifies whether or not the average should be computed for a dimension.

In the Properties view, this property is the **Compute Average** property in the **Value** category.

Default is FALSE.

Type: Boolean

computeCount (Compute Count)
Specifies whether or not 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.

Default is FALSE.

Type: Boolean

computeDistinctCount (Compute Distinct Count)
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.

Default is FALSE.

Type: Boolean

computeMaximum (Compute Maximum)
Specifies whether or not the maximum value should be computed for a dimension.

In the Properties view, this property is the **Compute Maximum** property in the **Value** category.

Default is FALSE.

Type: Boolean

computeMinimum (Compute Minimum)
Specifies whether or not the minimum value should be computed for a dimension.

In the Properties view, this property is the **Compute Minimum** property in the **Value** category.

Default is FALSE.

Type: Boolean

computeTotal (Compute Totals)
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 then one aggregation option is selected, the processor will pick the aggregate option that is highest up in the priority list:

1. computeTotal
2. computeAverage
3. computeMaximum
4. computeMinimum
5. computeCount
6. computeDistinctCount

Type: Boolean

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

controlCharacters (Control Characters)
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)
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 150 bar code type.

Type: Integer value.

Specifies the number of data symbols per line. 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.

displayFactRows (Display Fact Rows)
Specifies whether or not fact rows (the individual unaggregated data items) are displayed.

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)
Specifies whether or not 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)
Selects which of the declared dimensions or measures should be displayed.

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

drawAs (Draw As)
Specifies the type of chart that is 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 Category Chart: Bar, Bar3D, Stacked Bar, Line, Line3D, Area, Stacked Area, or Waterfall, Table, Sorted Table, Aggregated Table

> **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 Map Chart: Pie, Pie3D, Bar, Bar3D or Ring, Table, Sorted Table, Aggregated Table

Valid values for XY Chart: Polar, Scatter, Area, Stacked Area, Line, Step, Step Area, or Time Series, Table, Sorted Table

Valid values for Pivot Table: Pie, Pie3D, Ring, Bar, Bar3D, Area, StackedBar, StackedArea, Line, Line3D, Waterfall, Polar, Scatter, XYArea, XYStackedArea, XYLine, Step, StepArea, TimeSeries, Table

drawLabels (Draw Labels)
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)
Controls whether the 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 more then several 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**.

encoding (Encoding)
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 unique to the pdf-417 on page 150 bar code type.

Type: Encoding

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)

enumValues (Enum Values)
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

errorCorrectionDegree (Error Correction Degree)
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 unique to the pdf-417 on page 150 bar code type.

Type: Integer value.

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

fidelity (Text Fidelity)
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.

Type: Boolean. The default value is **false**.

fill (Fill)
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)
Controls the sizing behavior of the parent box when this 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)
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)
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)
Sets the font face name for this node and for all children of this 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.

See the Genero Studio >> Report Writer documentation topic "Configuring Fonts and Printers" for additional information about fonts.

fontSize (Size)
Sets the font size in points for this node and for all children of this 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)
Specify (control) 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 Align and Format Numbers. See the BDL **format** attribute for additional information and format characters.

Type: String.

fWidth (Fix Width)
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 157.

Type: PXML, point value.

Default value: None.

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)
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**.

See also: Page Headers and Footers

hidePageFooterOnLastPage (Hide Page Footer On Last Page)
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**.

See also: Page Headers and Footers

hierarchiesInputOrder (Hierarchies input order)
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

href (href)
This property can be used to define 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. See Using Hyperlinks in a Report.

Type: String.

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 106 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 108 property.

Type: PXML, point value.

Default value: none.

id (id)
This property can be used to create 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. See Using Hyperlinks in a Report.

Type: String.

indent (Indent)
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)
Controls the mapping of pixels to device pixels.

In the Properties view, this property is the **Intended Resolution** property in the **Image** category.

This property is relevant only if neither X-Size nor Y-Size are set, or if **tile** is set.

The default is the current screen resolution.

key (Key)
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)
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.

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 depends 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 56: A top-to-bottom layout direction

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

lBorder (Left Border)
The `lBorder` property sets the weight of the left border of a table object.

In the Properties view, this property is the **Left Border** property in the **Table** category.

The `lBorder` property overrides the more general border (Border) on page 94 property for a table object.

Type: PXML, point value.

Default value: None.

localizeText (Localize Text)
Check the box to indicate that there is a localized string 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 WordWrap Boxes.

Type: Boolean. Valid choices are True, False. The default value is False.

location (Location)
The location of an image file.

In the Properties view, this property is the **Location** property in the **Image** category.

Location values are URLs supporting the protocols "http", "file" and "data".

The location can be set to an absolute path and filename. In this instance, the image file remains the same for the duration of the report.

Variables (RTL expressions) can be used if the image 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. You specify a URL based on one or more variables using an RTL expression. This is demonstrated in the "`OrderReport.4rp`" where the "`ImageBox2`" element has the expression "`./images/database/"+orderline.product.prodpic.trim()`"

Type: String.

name (Name)
Assigns the specified 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 **""**.

noCheckDigits (Number Check Digits)
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)
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" and "codabar-2".

Type: Numeric.

noText (Hide Text)
Suppresses text output if set.

In the Properties view, this property is the **Hide Text** property in the **Bar Code** category.

Type: Boolean. The default value is **false**.

outputOrder (Output Order)
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

padding (Padding)
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)
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)
An offset that is 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)
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**.

preferRectangularSymbols (Prefer Rectangular Symbols)
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. Check the box to enable rectangular symbols.

Type: Boolean. Valid choices are True, False. The default value is False.

pWidth (Proportional Width)
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. Consider the following example: A table has three 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.

rawCodeValue (Raw Code Value)
Specify 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 150 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

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 94 property for a table object.

Type: PXML, point value.

Default value: None.

referenceDefault (Default)
The text value to be displayed, if 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 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 103 and vRule (Vertical Rule) on page 114 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: The Color Class on page 164

Default value: none.

section (Section)
The layout node attribute that controls printing 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.

evenPageHeader	Specific Page Header to be printed on even pages.
firstPageFooter	A 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.
evenPageFooter	Specific Page Footer to be printed on even pages.
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 will print 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 knows as a *primary port*.

See Print Headers and Footers.

scaleX (Scale X)
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)
Applies the specified scale in this 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**.

seriesTitle (Series Title)
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 .

smartParse (Smart Parse)
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".

Type: Boolean. The default value is **false.**

sortAscending (Sort Ascending)
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.

sortBy (Sort By)
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 categoriy/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.

splitOversizedItem (Split Oversized Items)
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.

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.

Type: Boolean.

strikethrough (Strikethrough)
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)
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, WordWrap 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 PageNo Boxes, BarCode Boxes, and Reference Boxes, the value of the text property specifies the maximum width.

textAlignment (Text Alignment)
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)
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.

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 width.

Type: PXML.

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 from these functions:

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.

See " Page N of M" examples.

thinToGapRelation (Thin To Gap Relation)
Controls the ratio of thin bars to the gaps between individual digits.

In the Properties view, this property is the **Thin To Gap Relation** property in the **Bar Code** category.

The value of a gap is calculated by the formula GAPWIDTH=THINBARWIDTH/thinToGapRelation. This parameter applies only to the 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" and "codabar-2".

Type: Numeric. Default values: 0.5 for the types "code-2-5-industrial", "code-2-5-inverted" and "code-2-5-IATA"; 1 for the types "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", "code-93" and "code-93-extended".

thinToThickRelation (Thin To Thick Relation)
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 formulaTHICKBARWIDTH=THINBARWIDTH/ thinToThickRelation. This parameter applies only to 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","code-32". and "codabar 2".

Type: Numeric. Default values: 1/3

title (Title)
Specifies the title of the report, output, or column.

In the Properties view, this property is the **Title** property in the **Metadata** category.

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.

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).

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).

Type: String.

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 94 property for a table object.

Type: PXML, point value.

Default value: None.

topN (Top N)
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

trimText (Trim Text)
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.

transformTransparently (Transform transparently)
The transformTransparently property changes the effect of the properties layoutDirection and swapX. When set, the transformation extends to the entire fragment so that entire documents can be rotated.

In the Properties view, this property is the **Transform transparently** property in the **Orientation** category.

Type: Boolean.

objectType (Type)
The type of the report element.

In the Properties view, this property is the **Type** property in the **Object** category.

This is automatically set when you drop a specific element on the page.

underline (Underline)
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)
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)
Specifies the value of the item.

In the Properties view, this property is the **Value** property in the **Miscellaneous** category.

Type (non-pivot table): Numeric.

Type (pivot table): Can be a String or Float depending on the declared dataType. 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.

valuesTitle (Values Title)
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.

visibilityCondition (Visibility Condition)
Boolean value (TRUE/FALSE) indicating whether the object is visible (not hidden).

In the Properties view, this property is the **Visibility Condition** property in the **Object** category.

Type: Boolean. The default is TRUE.

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 106 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 108 property.

Type: PXML, point value.

Default value: none.

x (X)
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.

xAxisTitle (xAxisTitle)
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.

X-Size (X-Size)
Gives the box a fixed dimension.

In the Properties view, this property is the **X-Size** property in the **Geometry** category.

> **Note:** If you want 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 will appear 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.

X-Size Adjustment (X-Size Adjustment)
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.

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** will not shrink the object below this size even if all children are smaller than this size or there are no children at all. Note that self-placing children are not considered.

Type: Enum. Choices are **shrinkToChildren**, **expandToParent**.

y (Y)
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.

yAxisTitle (yAxisTitle)
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.

Y-Size (Y-Size)
Gives the box a fixed dimension.

In the Properties view, this property is the **Y-Size** property in the **Geometry** category.

> **Note:** If you want 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 will appear distorted.

Do not set a value for this property in WordWrapBoxes, because the element should typically grow based on its content.

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.

Y-Size Adjustment (Y-Size Adjustment)
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 as much as possible without clipping any of the children. A value of **expandToParent** causes it to stretch as much as possible.

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** will not shrink the object below this size even if all children are smaller than this size or there are no children at all. Note that self placing-children are not considered.

Type: Enum. Choices are **shrinkToChildren**, **expandToParent**.

Properties related to margins and borders

Margin and Border-related properties for report elements.

- marginWidth
- marginRightWidth
- marginBottomWidth
- marginLeftWidth
- marginTopWidth
- borderWidth

- borderRightWidth
- borderBottomWidth
- borderLeftWidth
- borderTopWidth
- borderStyle
- borderRightStyle
- borderBottomStyle
- borderLeftStyle
- borderTopStyle
- borderColor
- borderRightColor
- borderBottomColor
- borderLeftColor
- borderTopColor
- roundedCorners
- paddingWidth
- paddingRightWidth
- paddingBottomWidth
- paddingLeftWidth
- paddingTopWidth

marginWidth

Sets the thickness of all an object's margins; can be overridden by specific Margin properties. See Design HowTo for illustrations.

Type: PXML, point value.

marginRightWidth

Sets the thickness of an object's right margin; overrides the property marginWidth. See Design HowTo for illustrations.

Type: PXML, point value.

marginBottomWidth

Sets the thickness of an object's bottom margin; overrides the property marginWidth. See Design HowTo for illustrations.

Type: PXML, point value.

marginLeftWidth

Sets the thickness of an object's left margin; overrides the property marginWidth. See Design HowTo for illustrations.

Type: PXML, point value.

marginTopWidth

Sets the thickness of an object's top margin; overrides the property marginWidth. See Design HowTo for illustrations.

Type: PXML, point value.

borderWidth

Sets the thickness of an object's borders; can be overridden by specific border properties. See Design HowTo for illustrations.

Type: PXML, point value. Default value is **2**.

borderRightWidth

Sets the thickness of an object's right border; overrides the borderWidth property. See Design HowTo for illustrations.

Type: PXML, point value.

borderBottomWidth

Sets the thickness of an object's bottom border; overrides the borderWidth property. See Design HowTo for illustrations.

Type: PXML, point value.

borderLeftWidth

Sets the thickness of an object's left border; overrides the borderWidth property. See Design HowTo for illustrations.

Type: PXML, point value.

borderTopWidth

Sets the thickness of an object's top border; overrides the borderWidth property. See Design HowTo for illustrations.

Type: PXML, point value.

borderStyle

Sets the style for all an object's borders. Can be overridden by specific borderSyle properties. See Design HowTo for illustrations.

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. See Design HowTo for illustrations.

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. See Design HowTo for illustrations.

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. See Design HowTo for illustrations.

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. See Design HowTo for illustrations.

Type: Enum, choices are: solid, dotted, dashed, groove, ridge, inset, outset, double.

borderColor
The `borderColor` property sets the color of all an object's borders.

In the properties window, this property is identified by the label **Border Color**.

Can be overridden by specific `borderColor` properties. See Design HowTo for illustrations.

Type: **Color**, valid colors are selected from the Edit Expression color palette.

borderRightColor

Sets the color of an object's right border. Can override the borderColor property. See Design HowTo for illustrations.

Type: **Color**, valid colors are selected from the Edit Expression color palette.

borderBottomColor

Sets the color of an object's bottom border. Can override the borderColor property. See Design HowTo for illustrations.

Type: **Color**, valid colors are selected from the Edit Expression color palette.

borderLeftColor

Sets the color of an object's left border. Can override the borderColor property. See Design HowTo for illustrations.

Type: **Color**, valid colors are selected from the Edit Expression color palette.

borderTopColor

Sets the color of an object's top border. Can override the borderColor property. See Design HowTo for illustrations.

Type: **Color**, valid colors are selected from the Edit Expression color palette.

roundedCorners

Specifies that the object's border corners will be round. This applies to the border styles **solid**, **dashed**, and **double** only. See Design HowTo for illustrations.

Type: Boolean. Valid choices are True, False. The default value is False.

paddingWidth

Sets the width of all of an object's padding. Can be overridden by specific padding properties. See Design HowTo for illustrations.

Type: PXML, point value.

paddingRightWidth

Sets the width of an object's right padding. Can override the paddingWidth property. See Design HowTo for illustrations.

Type: PXML, point value.

paddingBottomWidth

Sets the width of an object's bottom padding. Can override the paddingWidth property. See Design HowTo for illustrations.

Type: PXML, point value.

paddingLeftWidth

Sets the width of an object's left padding. Can override the paddingWidth property. See Design HowTo for illustrations.

Type: PXML, point value.

paddingTopWidth

Sets the width of an object's top padding. Can override the paddingWidth property. See Design HowTo for illustrations.

Type: PXML, point value.

Properties for Report Document Metadata

Report Metadata properties can be set in the report design document (`.4rp`).

Report Metadata properties can be set in the `4rp` report design document. For compatibility reports, which have no `4rp` document, API functions are provided. See Adding Report Document Metadata.

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

- title
- author
- creator
- subject
- keywords

title
Specifies the title of the object.

Type: String.

author
Specifies the metadata for the author of the report.

Type: String.

creator
Specifies the metadata for the creator of the report.

Type: String.

subject
Specifies the metadata for the subject of the report.

Type: String.

keywords
Specifies the metadata for the keyword of the report.

Type: String.

Bar Codes

The report element container for a Bar Code is a Barcode Box. This flow object draws bar codes.

These properties are specific to the Bar Code Box:

- codeType
- fidelity
- noText

- codeValue
- check
- noDigits
- noCheckDigits
- thinToThickRelation
- thinToGapRelation
- controlCharacters
- CodeValueExpression

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

Table 13: Bar Code Types

Bar Code Type	Number of Digits Supported	Normal size
codabar-18 on page 122	varies	(calculated width x 20mm h)
codabar-2 on page 121	varies	(calculated width x 20mm h)
code-11-matrix on page 123	varies	(calculated width x 1in h)
code-128 on page 124	varies	(calculated width x 6.5mm h)
code-2-5-datalogic on page 129	varies	(calculated width x 1in h)
code-2-5-IATA on page 129	varies	(calculated width x 1in h)
code-2-5-industrial on page 129	varies	(calculated width x 1in h)
code-2-5-interleaved on page 130	varies	(calculated width x 1in h)
code-2-5-inverted on page 130	varies	(calculated width x 1in h)
code-2-5-matrix on page 130	varies	(calculated width x 1in h)
code-BCD-matrix on page 130	varies	(calculated width x 1in h)
code-32 on page 130	9	(calculated w x h)
code-39 on page 132	varies	(calculated width x 1in h)
code-39-extended on page 138	varies	(calculated width x 1in h)
code-93 on page 142	varies	(calculated w x h)
code-93-extended on page 143	varies	(calculated w X h)
data-matrix on page 147	varies	
ean-8 on page 148	8	26.73mm x 21.64 mm (w x h)
ean-13 on page 148	13	37.29mm x 26.26mm (w x h)
ean-code-128 on page 149	varies	(calculated width x 6.5mm h)
ean-data-matrix on page 149	varies	
ean-supplemental-2 on page 149	2	6.6mm x 26.26mm (w x h)

Bar Code Type	Number of Digits Supported	Normal size
ean-supplemental-5 on page 149	5	15.5mm x 26.26mm (w x h)
gs1-8 on page 149	8	26.73mm x 21.64 mm (w x h)
gs1-13 on page 149	13	37.29mm x 26.26mm (w x h)
gs1-code-128 on page 150	varies	(calculated width x 6.5mm h)
gs1-data-matrix on page 150	varies	
gs1-supplemental-2 on page 150	2	6.6mm x 26.26mm (w x h)
gs1-supplemental-5 on page 150	5	15.5mm x 26.26mm (w x h)
intelligent-mail on page 150	varies, up to 31 digits.	
pdf-417 on page 150	varies	(calculated w x h)
qr-code on page 152	varies	
upc-a on page 155	12	1.469in x 1.020in (w x h)
upc-e on page 155	8	0.897in x 1.020in (w x h)
upc-supplemental-2 on page 156	2	0.26in x 1.02in (w x h)
upc-supplemental-5 on page 156	5	(0.611in x 1.02in w x h)

Bar Code type details

A description of each code type, the expected value type, semantic constraints and size information.

The character set and semantic constraints depend on the code type selected.

codabar-2

Details on the codabar-2 bar code type.

Codabar 2 can be used to encode text of variable length by using characters from this character set:

Table 14: Character set for Codabar 2

Reference Number	Character
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8

Reference Number	Character
9	9
10	-
11	$
12	:
13	/
14	.
15	+
16	a
17	b
18	c
19	d

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-(Sum(i=1 to n 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="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/ thinToThickRelation where thinToThickRelation should take values between1/3 and1/2. Between digits a gap having the width GAPWIDTH=THINBARWIDTH/ thinToGapRelationis 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' 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

Reference Number	Character
0	0
1	1

Reference Number	Character
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	-
11	$
12	:
13	/
14	.
15	+
16	a
17	b
18	c
19	d
16	t
17	n
18	*
19	e

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-(Sum(i=1 to n of Ref(i))) mod 16) Where Ref(i) is the reference number of the character i, and n is the total number of characters. Example: codeValue="a37859n"CS=16-((16+3+7+8+5+9+17) mod 16),CS=16-(65 mod 16),CS=15 Looking up reference number 15 yields the character '+'. The full code value including checksum is therefore: codeValue="a37859+n"

When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated.

The nominal height is 20mm. The nominal width of a thin bar is THINBARWIDTH=0.165mm. The width of a character is 2.095mm. Between digits a gap with the width GAPWIDTH=THINBARWIDTH/ thinToGapRelation is drawn. The padding on both sides measures 10*THINBARWIDTH.

code-11-matrix
Details on the code-11-matrix bar code type.

The code represents a character string with a variable number of characters. The string can contain the digits 0-9 and the '-' character. The number of digits can be specified by setting the noDigits

attribute. The code can contain up to two check characters. The attribute noCheckDigits specifies how many check characters are used. If two check characters are used, the rightmost character is the 'K' checksum character and the last but one character is the 'C' checksum character. if only one checksum character is specified then the rightmost character is a 'C' checksum character. The 'C' checksum is calculated asC=(Sum(i=1 to n of ((i-1 mod 10)+1)*Ref(n-i+1))) mod 11 and the 'K' checksum is calculated usingK=(Sum(i=1 to n of ((i-1 mod 9)+1)*Ref(n-i+1))) mod 11 where n specifies the number of characters to the left of the particular check digit and Ref(i) specifies the value of the character at position i, starting with the leftmost character having the value 1. For digits Ref() yields the digit value itself and for the '-' character Ref() yields the value 10. Example calculating the 'C' checksum: codeValue="12-12345-67890", noDigits="16", noCheckDigits="2", n=14C=(1*0+2*9+3*8+4*7+....10*2+1*1+2*10+3*2+4*1) mod 11=305 mod 11=8 The K checksum can then be calculated as: n=15, Ref(15)=C=8K=(1*8+2*0+3*9+4*8+....9*4+1*3+2*2+3*1+4*10+5*2+6*1) 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 124 lists the available characters in the character sets A, B and C.

Table 16: Available characters in the character sets A, B, and C

Reference Number	Character Set A	Character Set B	Character Set C
0	SP	SP	00
1	!	!	01
2	"	"	02
3	#	#	03
4	$	$	04
5	%	%	05
6	&	&	06
7	'	'	07
8	(	(	08
9	)	)	09
10	*	*	10
11	+	+	11
12	,	,	12
13	-	-	13
14	.	.	14

Reference Number	Character Set A	Character Set B	Character Set C
15	/	/	15
16	0	0	16
17	1	1	17
18	2	2	18
19	3	3	19
20	4	4	20
21	5	5	21
22	6	6	22
23	7	7	23
24	8	8	24
25	9	9	25
26	:	:	26
27	;	;	27
28	<	<	28
29	=	=	29
30	>	>	30
31	?	?	31
32	@	@	32
33	A	A	33
34	B	B	34
35	C	C	35
36	D	D	36
37	E	E	37
38	F	F	38
39	G	G	39
40	H	H	40
41	I	I	41
42	J	J	42
43	K	K	43
44	L	L	44
45	M	M	45
46	N	N	46
47	O	O	47
48	P	P	48

Reference Number	Character Set A	Character Set B	Character Set C
49	Q	Q	49
50	R	R	50
51	S	S	51
52	T	T	52
53	U	U	53
54	V	V	54
55	W	W	55
56	X	X	56
57	Y	Y	57
58	Z	Z	58
59	[	[	59
60	\	\	60
61	]	]	61
62	^	^	62
63	_	_	63
64	NUL	`	64
65	SOH	a	65
66	STX	b	66
67	ETX	c	67
68	EOT	d	68
69	ENQ	e	69
70	ACK	f	70
71	BEL	g	71
72	BS	h	72
73	HT	i	73
74	LF	j	74
75	VT	k	75
76	FF	l	76
77	CR	m	77
78	SO	n	78
79	SI	o	79
80	DLE	p	80
81	DC1	q	81
82	DC2	r	82

Reference Number	Character Set A	Character Set B	Character Set C
83	DC3	s	83
84	DC4	t	84
85	NAK	u	85
86	SYN	v	86
87	ETB	w	87
88	CAN	x	88
89	EM	y	89
90	SUB	z	90
91	ESC	{	91
92	FS	|	92
93	GS	}	93
94	RS	~	94
95	US	DEL	95
96	FNC3	FNC3	96
97	FNC2	FNC2	97
98	SHIFT	SHIFT	98
99	CODEC	CODEC	99
100	CODEB	FNC4	CODEB
101	FNC4	CODEA	CODEA
102	FNC1	FNC1	FNC1
103	STARTA	STARTA	STARTA
104	STARTB	STARTB	STARTB
105	STARTC	STARTC	STARTC
-	STOP	STOP	STOP

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 149 and gs1-code-128 on page 150 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

AI	No of Data Characters	Description
00	18	Identification of a delivery unit
01	14	An EAN 13 Number including check digit
10	up to 20 alphanumeric characters	Shipping batch identifier
11	3	Production date in the format YYMMDD

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

Textual representation	code value	Remark
(25)03x57	STARTC, FNC1, 25, 03, CODEB, x, 5, 7	Note that although, it is a variable length AI, no FNC1 is added at the end in the case when the AI is the last one.
(30)19(21)3456789012	STARTC, FNC1, 30, 19, CODEA, FNC1, CODEC, 21, 12, 34, 56, 78, 90, 12	Note that switching to code a is not necessary, as FNC1 exists in all three character sets.

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=(Ref(1)+Sum(i=2 to n of (i-1)*Ref(i))) mod 103 where Ref(i) is the reference number of the character i, and n is the

total number of characters. Example: codeValue="STARTB,A,B,C"CS=(104+(1*33)+(2*34)+(3*35)) mod 103,CS=310 mod 103,CS=1 Looking up reference number 1 in character set B yields the exclamation mark '!' character. The full code value including checksum and stop character is therefore: codeValue="STARTB,A,B,C,!,STOP"

Height and width

The nominal height is 6.5mm. The width of the bar can be calculated using this formula: L/mm=(5.5Nc +11Nab+35)*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 THINBARWIDTH=0.0236in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/ thinToThickRelation where thinToThickRelation thinToGapRelation is drawn. The padding on both sides measures 10*THINBARWIDTH.

code-2-5-IATA
Details on the code-2-5-IATA bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the noDigits attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars yielding a total of 5 bars per digit. The nominal width of a thin bar is 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 padding on both sides measures 10*THINBARWIDTH.

code-2-5-industrial
Details on the code-2-5-industrial bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the noDigits attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars yielding a total of 5 bars per digit. The nominal width of a thin bar is THINBARWIDTH=0.0236in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/ thinToThickRelation where thinToThickRelation should take values between1/3 and1/2. Between digits a gap of the width GAPWIDTH=THINBARWIDTH/ thinToGapRelation (Thin To Gap Relation) on page 111is drawn. The padding on both sides measures 10*THINBARWIDTH.

code-2-5-interleaved
Details on the code-2-5-interleaved bar code type.

The code represents a decimal number with a variable number of digits. The number of digits must be a multiple of 2. The number of digits can be specified by setting the noDigits attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars resulting in a total of 5 bars per digit. The nominal width of a thin bar is THINBARWIDTH=0.0236in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/ thinToThickRelation where thinToThickRelation should take values between1/3 and1/2. Between digits a gap of width GAPWIDTH=THINBARWIDTH/ thinToGapRelation is drawn. The padding on both sides measures 10*THINBARWIDTH.

code-2-5-inverted
Details on the code-2-5-inverted bar code type.

The code is the same as code 2/5 industrial, the only difference being that the gaps are drawn instead of the bars.

code-2-5-matrix
Details on the code-2-5-matrix bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the noDigits attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by noDigits then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars yielding a total of 5 bars per digit. The nominal width of a thin bar is THINBARWIDTH=0.0236in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/ thinToThickRelation where thinToThickRelation should take values between1/3 and1/2. Between digits a gap of width GAPWIDTH=THINBARWIDTH/ thinToGapRelationis 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 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-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

Reference Number	Character
0	0
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	B
10	C
11	C
12	D
13	F
14	G
15	H
16	J
17	K
18	L
19	M
20	N
21	P
22	Q
23	R
24	S
25	T
26	U
27	V
28	W

Reference Number	Character
29	X
30	Y
31	Z

Based on this table the following bar codes will produce identical bar pattern: <BARCODEBOX codeType="code 32" check="true" codeValue="026089019" noText="true" orientation="vertical" mirrored="true"/><BARCODEBOX codeType="code 39" check="false" codeValue="0SW5KV" noText="true" orientation="vertical" mirrored="true"/> Note that the bars are drawn without text by setting noText="true". This is necessary to produce identical output since otherwise the code 32 box draws the decimal number "026089019" while the code 39 box would draw the string "0SW5KV".

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars making a total of 5 bars per digit. The nominal width of a thin bar is THINBARWIDTH=0.0197in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/ thinToThickRelation where thinToThickRelation should take values between1/3 and1/2. 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

Reference Number	Character
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	G

Reference Number	Character
17	H
18	I
19	J
20	K
21	L
22	M
23	N
24	O
25	P
26	Q
27	R
28	S
29	T
30	U
31	V
32	W
33	X
34	Y
35	Z
36	-
37	.
38	
39	$
40	/
41	+
42	%

The last but one character is the checksum character that is calculated as follows:CS=Sum(i=1 to n of Ref(i))) mod 43 where Ref(i) is the reference number of the character i, and n is the total number of characters. Example: codeValue="DATALOGIC"CS=(13+10+29+10+21+24+16+18+12) mod 43,CS=153 mod 43,CS=24 Looking up reference number 24 yields the character 'O'. The full code value including checksum and stop character is therefore: codeValue="DATALOGICO"

Some scanners support an "extended mode" in which the scanner recognizes special two-character sequences of the code 39 character set and decodes these as ASCII characters. With this method the full 128 character ASCII character set can be encoded using the 43 basic characters of code 39. Scanners are switched into "extended mode" by a bar containing the sequence "+$". The sequence "-$" returns the scanner into regular mode. This table lists the character sequences needed to encode the ASCII character set in "extended mode".

Table 21: Character sequences needed to encode the ASCII character set in "extended mode"

ASCII code	Code 39 sequence
NUL	%U
SOH	$A
STX	$B
ETX	$C
EOT	$D
ENQ	$E
ACK	$F
BEL	$G
BS	$H
HT	$I
LF	$J
VT	$K
FF	$L
CR	$M
SO	$N
SI	$O
DLE	$P
DC1	$Q
DC2	$R
DC3	$S
DC4	$T
NAK	$U
SYN	$V
ETB	$W
CAN	$X
EM	$Y
SUB	$Z
ESC	%A
FS	%B
GS	%C
RS	%D
US	%E
SP	

ASCII code	Code 39 sequence
!	/A
"	/B
#	/C
$	/D
%	/E
&	/F
'	/G
(	/H
)	/I
*	/J
+	/K
,	/L
-	-
.	.
/	/O
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
:	/Z
;	%F
>TD/TD	
=	%H
>	%I
?	%J
@	%V
A	A
B	B

ASCII code	Code 39 sequence
C	C
D	D
E	E
F	F
G	G
H	H
I	I
J	J
K	K
L	L
M	M
N	N
O	O
P	P
Q	Q
R	R
S	S
T	T
U	U
V	V
W	W
X	X
Y	Y
Z	Z
[	%K
\	%L
]	%M
^	%N
_	%O
`	%W
a	+A
b	+B
c	+C
d	+D

ASCII code	Code 39 sequence
e	+E
f	+F
g	+G
h	+H
i	+I
j	+J
k	+K
l	+L
m	+M
n	+N
o	+O
p	+P
q	+Q
r	+R
s	+S
t	+T
u	+U
v	+V
w	+W
x	+X
y	+Y
z	+Z
{	%P
|	%Q
}	%R
~	%S
DEL	%T

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 between1/3 and1/2. 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 138 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

ASCII code	Textual representation
NUL	<NUL>
SOH	<SOH>
STX	<STX>
ETX	<ETX>
EOT	<EOT>
ENQ	<ENQ>
ACK	<ACK>
BEL	<BEL>
BS	<BS>
HT	<HT>
LF	<LF>
VT	<VT>
FF	<FF>
CR	<CR>
SO	<SO>
SI	<SI>
DLE	<DLE>
DC1	<DC1>
DC2	<DC2>
DC3	<DC3>
DC4	<DC4>
NAK	<NAK>
SYN	<SYN>
ETB	<ETB>
CAN	<CAN>
EM	<EM>
SUB	<SUB>
ESC	<ESC>

ASCII code	Textual representation
FS	<FS>
GS	<GS>
RS	<RS>
US	<US>
SP	
!	!
"	"
#	#
$	$
%	%
&	&
'	'
(	(
)	)
*	*
+	+
COMMA	,
-	-
.	.
/	/
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
:	:
;	;
<	<
=	=

ASCII code	Textual representation
>	>
?	?
@	@
A	A
B	B
C	C
D	D
E	E
F	F
G	G
H	H
I	I
J	J
K	K
L	L
M	M
N	N
O	O
P	P
Q	Q
R	R
S	S
T	T
U	U
V	V
W	W
X	X
Y	Y
Z	Z
[	[
\	\
]	]
^	^
_	_

ASCII code	Textual representation
`	`
a	a
b	b
c	c
d	d
e	e
f	f
g	g
h	h
i	i
j	j
k	k
l	l
m	m
n	n
o	o
p	p
q	q
r	r
s	s
t	t
u	u
v	v
w	w
x	x
y	y
z	z
{	{
|	|
}	}
~	~
DEL	<DEL>

code-93
Details on the code-93 bar code type.

Code 93 can be used to encode ASCII text of variable length. Characters can be selected from this set of characters. You can use the code-93-extended type to encode the full 128 character ASCII character set using the 47 basic characters of code 93.

Table 23: code-93

Reference Number	Character
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	G
17	H
18	I
19	J
20	K
21	L
22	M
23	N
24	O
25	P
26	Q
27	R
28	S

Reference Number	Character
29	T
30	U
31	V
32	W
33	X
34	Y
35	Z
36	-
37	.
38	
39	$
40	/
41	+
42	%
43	!
44	?
45	\
46	|

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=(Sum(i=1 to n of ((i-1 mod 20)+1)*Ref(n-i+1))) mod 47 and the 'K' checksum is calculated as followsK=(Sum(i=1 to n of ((i-1 mod 15)+1)*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. T

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

ASCII code	Code 93 sequence
NUL	?U
SOH	!A
STX	!B
ETX	!C
EOT	!D
ENQ	!E
ACK	!F
BEL	!G
BS	!H
HT	!I
LF	!J
VT	!K
FF	!L
CR	!M
SO	!N
SI	!O
DLE	!P
DC1	!Q
DC2	!R
DC3	!S
DC4	!T
NAK	!U
SYN	!V
ETB	!W
CAN	!X
EM	!Y
SUB	!Z
ESC	?A
FS	?B
GS	?C

ASCII code	Code 93 sequence
RS	?D
US	?E
SP	
!	\A
"	\B
#	\C
$	$
%	%
&	\F
'	\G
(	\H
)	\I
*	\J
+	+
COMMA	\L
-	-
.	.
/	/
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
:	\Z
;	?F
<	?G
=	?H
>	?I
?	?J

ASCII code	Code 93 sequence
@	?V
A	A
B	B
C	C
D	D
E	E
F	F
G	G
H	H
I	I
J	J
K	K
L	L
M	M
N	N
O	O
P	P
Q	Q
R	R
S	S
T	T
U	U
V	V
W	W
X	X
Y	Y
Z	Z
[	?K
\	?L
]	?M
^	?N
_	?O
`	?W
a	|A

ASCII code	Code 93 sequence
b	|B
c	|C
d	|D
e	|E
f	|F
g	|G
h	|H
i	|I
j	|J
k	|K
l	|L
m	|M
n	|N
o	|O
p	|P
q	|Q
r	|R
s	|S
t	|T
u	|U
v	|V
w	|W
x	|X
y	|Y
z	|Z
{	?P
|	?Q
}	?R
~	?S
DEL	?T

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 this attribute (codeValue) 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.

This attribute is unique to this bar code type:

- preferRectangularSymbols
  - Type: Boolean value
  - Description: 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.
  - Fails if: Value cannot be parsed as a boolean value. Valid values are True, False.
  - Default value: False

ean-8
Details on the ean-8 bar code type.

> **Note:** The ean-8 and gs1-8 on page 149 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 149 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 124 bar code.

> **Note:** The ean-code-128 and gs1-code-128 on page 150 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 150 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 150 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 150 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 148.

gs1-13
A synonym of the ean-13 bar code type.

See ean-13 on page 148.

gs1-code-128
A synonym for the ean-code-128 bar code type.

See ean-code-128 on page 149.

gs1-data-matrix
A synonym for the ean-data-matrix bar code type.

See ean-data-matrix on page 149.

gs1-supplemental-2
A synonym of the ean-supplemental-2 bar code type.

See ean-supplemental-2 on page 149.

gs1-supplemental-5
A synonym of the ean-supplemental-5 bar code type.

See ean-supplemental-5 on page 149.

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

Type	Field	Digits
Tracking code field	Barcode Identifier	2 digits; 2nd digit must be 0-4.
Tracking code field	Service Type Identifier	3 digits.
Tracking code field	Mailer Identifier	6 or 9 digits.
Tracking code field	Serial Number	9 digits, when used with a 6-digit Mailer Identifier. 6 digits, when used with a 9-digit Mailer Identifier.
Routing code field	Delivery Point ZIP Code	0, 5, 9, or 11 digits.

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 this attribute (codeValue) 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 unique to this bar code type:

- dataSymbolsPerLine (Data Symbols per Line) on page 98:
  - Type: Integer value.
  - Specifies the number of data symbols per line. 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 107:
  - 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 100:
  - Type: Integer value.
  - 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 99:
  - Type: Encoding
  - 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)

Attributes unique to this bar code type:

- errorCorrectionDegree
  - Type: Integer value
  - Description: 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
- Description: Sets the encoding for non-ASCII characters in the code value. The following encodings can be set:

Encoding Type	Description
ISO-8859-1	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 "UTF-8" 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.)
Bytes	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.
UTF-8	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.
Any encoding listed by CharsetInfo (e.g. "Shift_JIS", "Big5" or "ISO-8859-8")	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.
Any encoding listed by CharsetInfo prefixes with the string "RAW-" (e.g. "RAW-Shift_JIS", "RAW-Big5" or "RAW-UTF-8"	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.

- Fails if: Value is not a valid encoding name.
- Default value: UTF-8

QR code examples

Figure 57: Hello World: size not specified (default error correction (3))

Figure 58: Hello World: size not specified (error correction degree (2))

Figure 59: Hello World: size not specified (error correction degree (1))

Figure 60: Hello World: width="3cm"

Figure 61: http://www.4js.com

upc-a
Details on the upc-a bar code type.

The code represents a 12 digit decimal number. First digit is the 'number system character' code, the next five identify the manufacturer and the following five identify an article. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with such a 11 digit value then the 12th digit is automatically calculated by the system.

The nominal size is 1.469in x 1.020in (w x h). The bar code area has the measurements 1.235in x 0.965in. The left padding measures 0.117in.

upc-e
Details on the upc-e bar code type.

The code represents a 8 digit decimal number. Upc-e is a compressed version of upc-a. By the method of zero suppression some upc a codes are available as upc e codes. This translation table shows which upc-a numbers can be transformed to short versions:

Table 26: upc-a numbers that can be transformed to short versions

UPC-E Number	Digits to insert	Position of Insertion	Resulting UPC-A Number
sNNNNN0c	00000	3	sNN00000NNNc
sNNNNN1c	10000	3	sNN10000NNNc
sNNNNN2c	20000	3	sNN20000NNNc
sNNNNN3c	00000	4	sNNN00000NNc
sNNNNN4c	00000	5	sNNNN00000Nc
sNNNNN5c	00005	6	sNNNNN00005c
sNNNNN6c	00006	6	sNNNNN00006c
sNNNNN7c	00007	6	sNNNNN00007c
sNNNNN8c	00008	6	sNNNNN00008c
sNNNNN9c	00009	6	sNNNNN00009c

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:

```
<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 Classes Overview

Object Classes

There are object classes for each type of the report item properties. 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

Class	Description
`Boolean`	Contains methods used for all logical operations.
`Color`	Contains methods and static member variables related to color.
`Enum`	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.
`Date`	Provides methods for date formatting and parsing.
`FGLNumericVariable`	For every 4GL numeric variable of report data, an object is created that is an instance of an FGLNumericVariable. These objects hold the value of the 4GL variable.
`FGLStringVariable`	For every `CHAR`, `VARCHAR`, `STRING`, `TEXT`, `DATE`, `DATETIME` and `INTERVAL` 4GL variable of report data, an object is created that is an instance of an `FGLStringVariable`. These objects hold the value of the 4GL variable.

Class	Description
`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.
`Runtime`	Provides functions that simplify the creation of dynamic designs that change behavior based on the runtime setup.
`String`	Contains methods used for all string operations.

Dimension Resolver

Unit Names

Table 28: Unit Names

Unit abbreviations	Unit description	Point value	Example
point|pt	Point value (72.27point = 1in)	1	10point
scrpixels[xy]	Screen pixel value (e.g. for 96DPI: 96pixel = 1in = 72.27point)	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.	640scrpixelsx
prnpixels[xy]	Printer pixel value (e.g. for 300DPI 300pixel = 1in = 72.27point)	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).	150prnpixelsy
pica|pc	Pica value (12point = 1pica)	12	3pica
inch|in	Inch value (72.27point = 1in)	72.27	10in
bigpoint|bp	Big point (72bigpoint = 1in)	72.27/72	10bp
cm	Centimeter value (2.54cm = 1in)	72.27/2.54	10cm
mm	Centimeter value (10mm = 1cm)	72.27/2.54/10	10mm
didot|dd	Didot point (1157dd = 1238pt)	1238/1157	10dd
cicero|cc	Cicero value(1cc = 12dd)	12*1238/1157	0.5cc

Paper Format Abbreviations

Table 29: Paper Format Abbreviations

Unit abbreviations	Unit Description	Width value	Length value
iso4a0(width | length)	ISO 4A0	1682mm	2378mm
iso2a0(width | length)	ISO 2A0	1189mm	1682mm
(isodesignatedlong | dl | dinlang)(width | length)	ISO designated long	110mm	220mm
executive(width | length)	Executive	7.25in	10.5in
(folio | germanlegalfanfold)(width | length)	Folio/German legal fanfold	8.5in	13in
(invoice | statement)(width | length)	Invoice/Statement	5.5in	8.5in
(ledger | tabloid | 11x7)(width | length)	Ledger/Tabloid	11in	7in
(naletter | letter | note)(width | length)	Letter	8.5in	11in
(nalegal | legal)(width | length)	Legal	8.5in	14in
quarto(width | length)	Quarto	215mm	275mm
a(width | length)	Engineering A	8.5in	11in
b(width | length)	Engineering B	11in	17in
c(width | length)	Engineering C	17in	22in
d(width | length)	Engineering D	22in	34in
e(width | length)	Engineering E	34in	44in
(na10x15envelope | 10x15envelope)(width | length)	Envelope 10x15	10in	15in
(na10x14envelope | 10x14envelope)(width | length)	Envelope 10x14	10in	14in
(na10x13envelope | 10x13envelope)(width | length)	Envelope 10x13	10in	13in
(na9x12envelope | 9x12envelope)(width | length)	Envelope 9x12	9in	12in
(na9x11envelope | 9x11envelope)(width | length)	Envelope 9x11	9in	11in

Unit abbreviations	Unit Description	Width value	Length value
(na7x9envelope | 7x9envelope)(width | length)	Envelope 7x9	7in	9in
(na6x9envelope | 6x9envelope)(width | length)	Envelope 6x9	6in	9in
(nanumber9envelope | number9envelope | env9)(width | length)	Envelope number 9	3+7/8in	8+7/8in
(nanumber10envelope | number10envelope | env10)(width | length)	Envelope number 10	1+1/8in	9+1/2in
(nanumber11envelope | number11envelope | env11)(width | length)	Envelope number 11	4+1/2in	10+3/8in
(nanumber12envelope | number12envelope | env12)(width | length)	Envelope number 12	4+3/4in	11in
(nanumber14envelope | number14envelope | env14)(width | length)	Envelope number 14	5in	11+1/2in
(envinvite | inviteenvelope | invite)(width | length)	Invite envelope	220mm	220mm
(envitaly | italyenvelope | italy)(width | length)	Italy envelope	110mm	230mm
(envmonarch | monarchenvelope | monarch)(width | length)	Monarch envelope	3+7/8in	7+1/2in
(envpersonal | personalenvelope | personal)(width | length)	Personal Envelope	3+5/8in	6+1/2in
(usstandardfanfold | usstdfanfold)(width | length)	US standard fanfold	14.875in	11in
(germanstandardfanfold | germanstdfanfold)(width | length)	German standard fanfold	8.5in	12in
(isoa0 | a0 | dina0)(width | length)	ISO/DIN & JIS A0	841mm	1189mm
(isoa1 | a1 | dina1)(width | length)	ISO/DIN & JIS A1	594mm	841mm
(isoa2 | a2 | dina2)(width | length)	ISO/DIN & JIS A2	420mm	594mm
(isoa3 | a3 | dina3)(width | length)	ISO/DIN & JIS A3	297mm	420mm

Unit abbreviations	Unit Description	Width value	Length value
(isoa4 | a4 | dina4)(width | length)	ISO/DIN & JIS A4	210mm	297mm
(isoa5 | a5 | dina5)(width | length)	ISO/DIN & JIS A5	148mm	210mm
(isoa6 | a6 | dina6)(width | length)	ISO/DIN & JIS A6	105mm	148mm
(isoa7 | a7 | dina7)(width | length)	ISO/DIN & JIS A7	74mm	105mm
(isoa8 | a8 | dina8)(width | length)	ISO/DIN & JIS A8	52mm	74mm
(isoa9 | a9 | dina9)(width | length)	ISO/DIN & JIS A9	37mm	52mm
(isoa10 | a10 | dina10)(width | length)	ISO/DIN & JIS A10	26mm	37mm
(isob0 | b0 | dinb0)(width | length)	ISO/DIN B0	1000mm	1414mm
(isob1 | b1 | dinb1)(width | length)	ISO/DIN B1	707mm	1000mm
(isob2 | b2 | dinb2)(width | length)	ISO/DIN B2	500mm	707mm
(isob3 | b3 | dinb3)(width | length)	ISO/DIN B3	353mm	500mm
(isob4 | b4 | dinb4)(width | length)	ISO/DIN B4	250mm	353mm
(isob5 | b5 | dinb5)(width | length)	ISO/DIN B5	176mm	250mm
(isob6 | b6 | dinb6)(width | length)	ISO/DIN B6	125mm	176mm
(isob7 | b7 | dinb7)(width | length)	ISO/DIN B7	88mm	125mm
(isob8 | b8 | dinb8)(width | length)	ISO/DIN B8	62mm	88mm
(isob9 | b9 | dinb9)(width | length)	ISO/DIN B9	44mm	62mm
(isob10 | b10 | dinb10)(width | length)	ISO/DIN B10	31mm	44mm
(isoc0 | c0 | dinc0)(width | length)	ISO/DIN C0	917mm	1297mm
(isoc1 | c1 | dinc1)(width | length)	ISO/DIN C1	648mm	917mm
(isoc2 | c2 | dinc2)(width | length)	ISO/DIN C2	458mm	648mm

Unit abbreviations	Unit Description	Width value	Length value
(isoc3 | c3 | dinc3)(width | length)	ISO/DIN C3	324mm	458mm
(isoc4 | c4 | dinc4)(width | length)	ISO/DIN C4	229mm	324mm
(isoc5 | c5 | dinc5)(width | length)	ISO/DIN C5	162mm	229mm
(isoc6 | c6 | dinc6)(width | length)	ISO/DIN C6	114mm	162mm
(isoc7 | c7 | dinc7)(width | length)	ISO/DIN C7	81mm	114mm
(isoc8 | c8 | dinc8)(width | length)	ISO/DIN C8	57mm	81mm
(isoc9 | c9 | dinc9)(width | length)	ISO/DIN C9	40mm	57mm
(isoc10 | c10 | dinc10)(width | length)	ISO/DIN C10	28mm	40mm
jisb0(width | length)	JIS B0	1030mm	1456mm
jisb1(width | length)	JIS B1	728mm	1030mm
jisb2(width | length)	JIS B2	515mm	728mm
jisb3(width | length)	JIS B3	364mm	515mm
jisb4(width | length)	JIS B4	257mm	364mm
jisb5(width | length)	JIS B5	182mm	257mm
jisb6(width | length)	JIS B6	128mm	182mm
jisb7(width | length)	JIS B7	91mm	128mm
jisb8(width | length)	JIS B8	64mm	91mm
jisb9(width | length)	JIS B9	45mm	64mm
jisb10(width | length)	JIS B10	32mm	45mm

Customize Report Designer: 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

RTL Class Reference

- The Boolean Class on page 163
- The Color Class on page 164
- The Date Class on page 167
- The Enum Classes on page 169
- The FGLNumericVariable Class on page 176
- The FGLStringVariable Class on page 177
- The Numeric Class on page 178
- The Runtime Class on page 188
- The String Class on page 192

The Boolean Class

Details about the Boolean class and its public members.

- Syntax
- Member Objects
- Usage

Syntax

```
Boolean
```

Member Objects

Table 30: Member Objects (Static Member Variables)

Name	Description
TRUE	Contains the value TRUE.
FALSE	Contains the value FALSE.

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
```

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 31: Member Objects (Static Member Variables)

Name	Description
BLACK	Specifies a Color object having the color BLACK
BLUE	Specifies a Color object having the color BLUE
CYAN	Specifies a Color object having the color CYAN
DARK_GRAY	Specifies a Color object having the color DARK_GRAY
GRAY	Specifies a Color object having the color GRAY
GREEN	Specifies a Color object having the color GREEN
LIGHT_GRAY	Specifies a Color object having the color LIGHT_GRAY
MAGENTA	Specifies a Color object having the color MAGENTA
ORANGE	Specifies a Color object having the color ORANGE
PINK	Specifies a Color object having the color PINK
RED	Specifies a Color object having the color RED
WHITE	Specifies a Color object having the color WHITE
YELLOW	Specifies a Color object having the color YELLOW

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 32: Class Methods (Static Member Methods)

Name	Description
`fromHSBA( h Numeric, s Numeric, b Numeric )`	Returns a Color object based on the specified values of hue, saturation, and brightness for the HSB color model. `Color.fromHSBA(h,s,b)` • *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).
`fromHSBA( h Numeric, s Numeric, b Numeric, a Numeric)`	Returns a Color object based on the specified values of hue, saturation, brightness, and alpha for the HSB color model. `Color.fromHSBA(h,s,b,a)` • *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.
`fromRGBA( r Numeric, g Numeric, b Numeric)`	Returns an opaque sRGB Color object with the specified red, green, and blue values. `Color.fromRGBA(r,g,b)` • *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)

Name	Description
	Alpha defaults to 1.0. The color used depends on the best match from the colors available for the output device.
`fromRGBA( r Numeric, g Numeric, b Numeric, a Numeric)`	Returns an sRGB Color object with the specified red, green, blue, and alpha values. `fromRGBA(r,g,b,a)` • *r* (red) is within the range (0.0 - 255) • *g* green) is within the range (0.0 - 255) • *b* (blue) is within the range (0.0 - 255) • *a* (alpha) is within the range (0.0 - 255) The color used depends on the best match from the colors available for the output device.

Example - Color Class Method

```
Color.fromRGBA(0.5,0.5,0.5,0.5)
```

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 33: Object Methods

Name	Description
`brighter()`	Brightens the Color object. 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 darker and brighter methods might give an inconsistent result because of rounding errors.
`darker()`	Darkens the Color object. 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

Name	Description
	of the darker and brighter methods might give an inconsistent result because of rounding errors.
`getAlpha()`	Returns the Numeric alpha component in the range 0-255.
`getBrightness()`	Returns the Numeric brightness value of the value in the HSB color model
`getBlue()`	Returns the Numeric blue component in the range 0-255 in the default sRGB space.
`getGreen()`	Returns the Numeric green component in the range 0-255 in the default sRGB space.
`getRGBA()`	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.
`getHue()`	Returns the hue value of the value in the HSB color model.
`getRed()`	Returns the Numeric red component in the range 0-255 in the default sRGB space.
`getSaturation()`	Returns the saturation value of the value in the HSB color model.
`toString()`	Return a string representation in the form "#argb".

Example - Color Object Method

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

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 34: Class Methods (Static Member Methods)

Name	Description
`fromIsoValue(String value)`	Constructs a Date from the date value; returns a Date instance representing the value. The value is expected in iso format ("YYYY-MM-DD"). Using the function guarantees that the Date value is constructed correctly without the danger of runtime parse errors due to changing 4GL date formatting or locale settings.
`parseString(String value, String format)`	Constructs a Date from *value*. The parsing is based on the passed *format* pattern. See Formatting Symbols for Dates.
`format(java.lang.String.format)`	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.
`today()`	Constructs a Date from the current date value. Returns a Date instance representing the current date.

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 '.' For example, this expression uses the isoValue of the variable **orderline.orders.orderdate** to create a valid date for this value in the specified format: `Date.fromIsoValue(orderline.orders.orderdate.isoValue).format("DDD DD MMM YYYY")`

Formatting symbols

The formatting symbol can use either uppercase or lowercase letters.

Table 35: Formatting Symbols for Dates

Character	Description
`dd`	Day of the month as a two-digit integer.
`ddd`	Three-letter English-language abbreviation of the day of the week, for example, Mon, Tue.
`mm`	Month as a two-digit integer.
`mmm`	Three-letter English-language abbreviation of the month, for example, Jan, Feb.
`yy`	Year, as a two-digit integer representing the two trailing digits.

Character	Description
yyyy	Year as a four-digit number.

The Enum Classes

These classes provide the list of valid values for form item properties that are of type Enum.

- The Alignment Class on page 169
- The TextAlignment Class on page 169
- The BaselineType Class on page 170
- The LayoutDirection Class on page 170
- The Y-SizeAdjustment Class on page 171
- The PageNoFormat Class on page 171
- The TrimText Class on page 171
- The X-SizeAdjustment Class on page 172
- FloatingBehavior Class on page 172
- Section Class on page 172
- XYChartDrawAs Class on page 173
- MapChartDrawAs Class on page 173
- CategoryChartDrawAs Class on page 174
- CodeType Class on page 174
- BorderStyles Classes on page 176

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 36: Member Objects (Static Member Variables)

Name
Baseline
Center
Far
Near
None

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 37: Member Objects (Static Member Variables)

Name
Center
Left
Right

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 38: Member Objects (Static Member Variables)

Name
Leftleft
Leftright
Rightleft
Rightright

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 39: Member Objects (Static Member Variables)

Name
BottomToTop
HorizontalNatural
LeftToRight
RightToLeft
Swapped
TopToBottom

Name
TurnLeft
TurnRight
Unturned
UpsideDown
VerticalNatural
VerticalUnnatural

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 the valid values for the Y-SizeAdjustment property.

Table 40: Member Objects (Static Member Variables)

Name
ExpandtoParent
ShrinktoChildren

The PageNoFormat Class

Public static member variables that represent the valid values for the pageNoFormat property.

Syntax

```
PageNoFormat
```

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

Table 41: Member Objects (Static Member Variables)

Name
Arabic
Lowerroman
Upperroman

The TrimText Class

Public static member variables that represent the valid values for the trimText property.

Syntax

```
TrimText
```

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

Table 42: Member Objects (Static Member Variables)

Name
Both
Compress
Left
Right

The X-SizeAdjustment Class

Public static member variables that represent the valid values for the X-SizeAdjustment property.

Syntax

```
X-SizeAdjustment
```

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

Table 43: Member Objects (Static Member Variables)

Name
ExpandtoParent
ShrinktoChildren

FloatingBehavior Class

Public static member variables that represent the valid values for the floatingBehavior property.

Syntax

```
FloatingBehavior
```

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

Table 44: Member Objects (Static Member Variables)

Name
Encloses
Free

Section Class

Public static member variables that represent the valid values for the Section property.

Syntax

```
Section
```

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

Table 45: Member Objects (Static Member Variables)

Name
AnyPageFooter
AnyPageHeader
EvenPageHeader
EvenPageFooter
FirstPageHeader
FirstPageFooter
OddPageFooter
OddPageHeader
ItemSeparator

XYChartDrawAs Class

Public static member variables that represent the valid values for the XY chart DrawAs property.

Syntax

```
XYChartDrawAs
```

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

Table 46: Member Objects (Static Member Variables)

Name
Polar
Scatter
Area
StackedArea
Line
Step
StepArea
TimeSeries

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 47: Member Objects (Static Member Variables)

Name
Bar
Bar3D
Pie
Pie3D
Ring

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 48: Member Objects (Static Member Variables)

Name
Area
Bar
Bar3D
Line
Line3D
StackedArea
StackedBar
Waterfall

CodeType Class

Public static member variables that represent the valid values for the codeType property.

Syntax

```
CodeType
```

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

Table 49: Member Objects (Static Member Variables)

Name
Upc_a
Upc_e
Upc_supplemental_2
Upc_supplemental_5
Gs1_13
Ean_13
Gs1-8
Ean_8
Gs1_supplemental_2
Ean_supplemental_2
Gs1_supplemental_5
Ean_supplemental_5
Code_128
Gs1_code_128
Ean_Code_128
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
Code_93
Code_93_extended
Codabar_18
Codabar_2
Pdf_417
Data_matrix
Gs1_data_matrix

Name
Ean_data_matrix
Qr_code
Intelligent_mail

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 50: Member Objects (Static Member Variables)

Name
Dashed
Dotted
Double
Groove
Inset
Outset
Ridge
solid

The FGLNumericVariable Class

For every Genero numeric variable (`INTEGER`, `SMALLINT`, `FLOAT`, `SMALLFLOAT`, `DECIMAL` and MONEY) of report data, an object is created that is an instance of an FGLNumericVariable. These objects hold the value of the 4GL variable.

- Syntax on page 176
- Member Objects on page 176
- Class Methods on page 177
- Class Usage on page 177

Syntax

```
FGLStringVariable
```

Member Objects

Defined fields for the `FGLNumericVariable` class.

Table 51: Member Objects (Member Variables)

Name	Description
`value`	A Numeric containing the value of the field; this is the same as the value of the 4GL variable. See Usage.
`fglValue`	A String containing the value of the field as formatted by the DVM. See Usage.
`name`	A String specifying the name of the field.
`caption`	A String specifying the title of the field.
`type`	A String specifying the type of the field.

Class Methods

This subclass inherits the public methods of the Numeric class.

Class Usage

In addition to the value of the 4GL variable, these objects contain member variables, among which is the variable `value` also containing the numeric value. For this reason, it is legal to write "`order_line.itemprice`" in your expression as a shortcut for "`order_line.itemprice.value`".

The member variable `fglValue` contains a String representing the numeric value as formatted by the DVM, taking into consideration such parameters as USING, DBMONEY, etc. In contrast, the member variable `value` is a numeric value without formatting.

The FGLStringVariable Class

For every `CHAR`, `VARCHAR`, `STRING`, `TEXT`, `DATE`, `DATETIME` and `INTERVAL` Genero variable of report data, an object is created that is an instance of an `FGLStringVariable`. These objects hold the value of the Genero variable.

- Syntax
- Member Objects on page 177
- Class Methods on page 178
- Class Usage on page 178

Syntax

```
FGLStringVariable
```

Member Objects

Defined fields for the FGLStringVariable class.

Table 52: Member Objects (Member Variables)

Name	Description
`value`	A String containing the value of the field; this is the same as the value of the Genero variable. See Usage.
`name`	A String specifying the name of the field.
`caption`	A String specifying the title of the field.

Name	Description
`type`	A String specifying the type of the field.

Class Methods

This subclass inherits the public methods of the String class.

Class Usage

In addition to the value of the Genero variable, these objects contain a member variable "`value`" which also contains the value. For this reason, it is legal to write "`order_line.billState`" in your expression as a shortcut for "`order_line.billState.value`".

The Numeric Class

Details about the Numeric class and its public members.

Values for this data type are limited to 15 significant digits.

- Syntax
- Methods
- Usage
- Examples

Syntax

```
Numeric
```

Methods

Table 53: Object Methods

Name	Description
`abs()`	Returns the absolute value.
`atan2(Numeric x)`	Returns the angle *theta* from the conversion of rectangular coordinates (x, y) to polar coordinates (r, *theta*)
`byteValue()`	Returns the value converted to a byte.
`cbrt()`	Returns the cube root of a numeric value.
`ceil()`	Returns the smallest numeric value that is greater than or equal to the value and is equal to a mathematical integer.
`cos()`	Returns the trigonometric cosine of an angle.
`cosh()`	Returns the hyperbolic cosine of a numeric value.
`exp()`	Returns the base -e exponential.
`floor()`	Returns the largest numeric value that is less than or equal to the value and is equal to a mathematical integer.
`format("format-string")`	Converts the value to a string representation defined by a format string. For example, for DECIMAL and FLOAT data types, *format-string* consists of pound signs (#) that represent digits and a decimal point; that is, "###.##" produces three places to the left of the decimal point

Name	Description
	and exactly two to the right. Additional options for the string are listed in the Usage section, format.
`getExponent()`	Returns the unbiased exponent used in the representation of a numeric value.
`intValue()`	Returns the value converted to an integer.
`isInfinite()`	Returns TRUE if the value has infinite value
`isNaN()`	Returns TRUE if the value is not a number
`isNull()`	Returns true if `toDouble()` is 0 and 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

Name	Description
	operators. Specifically a numeric input variable that is null behaves in arithmetic like the 0 value.
`log()`	Returns the natural logarithm (base e) of a numeric value.
`log10()`	Returns the base 10 logarithm of a numeric value.
`max(Numeric b)`	Returns the greater of two values.
`min(Numeric b)`	Returns the smaller of two values.
`rint()`	Returns the Numeric value that is closest in value to the value and is equal to a mathematical integer.
`round()`	Returns the closes integer to the value.
`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.
`sin()`	Returns the trigonometric sine, measured in radians.
`sinh()`	Returns the hyperbolic sine of a numeric value.
`sqrt()`	Returns the tangent of an angle measured in radians.
`tan()`	Returns the tangent of an angle measured in radians.
`tanh()`	Returns the hyperbolic tangent of a numeric value.
`toBoolean()`	Returns the Boolean false when the value is 0. Returns true for any other value.
`toChar()`	Returns the unicode character representation of a numeric value.
`toColor()`	Returns a color object. The value is interpreted as a RGB integer.
`toDegrees()`	Converts the value from radians to degrees.
`toRadians()`	Converts the value from degrees to radians.
`toString()`	Converts the value to a string representation.
`getPhysicalPageNumber()`	Gets the current page number of the physical page.

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.

abs()

```
abs()
```

Returns the absolute value of an int value:

- 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.

cos()

```
cos()
```

Returns the trigonometric cosine of an angle. However, 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.

sin()

```
sin()
```

Returns the trigonometric sine of an angle. However,

- 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.

tan()

```
tan()
```

Returns the trigonometric tangent of an angle. 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.

byteValue()

```
byteValue()
```

Returns the value converted to a byte .

cbrt()

```
cbrt()
```

Returns the cube root of a double value. 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.

ceil()

```
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.

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

cosh()

```
cosh()
```

Returns the hyperbolic cosine of a double value. The hyperbolic cosine 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 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.

exp()

```
exp()
```

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

```
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.

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.

getExponent()

```
getExponent()
```

Returns the unbiased exponent used in the representation of a float.

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.

intValue()

```
intValue()
```

returns the value converted to an integer (signed 32 bit)

isInfinite()

```
isInfinite()
```

returns the Boolean value true in case the value has the infinite value

isNaN()

```
isNaN()
```

returns the Boolean value true in case that the value is not a number

log()

```
log()
```

Returns the natural logarithm (base e) of a double value.

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.

log10()

```
log10()
```

Returns the base 10 logarithm of a double value.

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 10n 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.

rint()

```
rint()
```

Returns the double (floating-point) value that is closest in value to the value and is equal to a mathematical integer. 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.

Returns:

the closest floating-point value to a that is equal to a mathematical integer.

round()

```
round()
```

Returns the closest int to the value. 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:

`(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.

See Also:

Integer.MAX_VALUE, Integer.MIN_VALUE

signum ()

```
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.

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.

Returns:

the signum function of the value

sinh()

```
sinh()
```

Returns the hyperbolic sine of a double value. 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.

sqrt()

```
sqrt()
```

Returns the correctly rounded positive square root of a double value.

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

tanh()

```
tanh()
```

Returns the hyperbolic tangent of a double value. The hyperbolic tangent of x is defined to be (ex - e-x)/(ex + e-x), in other words, sinh(x)/cosh(x). Note that 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.

toChar()

```
toChar()
```

Converts the value to a unicode character representation.

For example, `65.toChar()` yields "A".

atan2(Numeric *x*))

```
atan2(Numeric x))
```

Returns the angle theta from the conversion of rectangular coordinates (x, y) to polar coordinates (r, theta). 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.

Parameters:

x - the abscissa coordinate

max(Numeric *b*)

```
max(b)
```

Returns the greater of two int values. That is, 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.

Parameters:

- *b* - an argument.

See Also: Long.MAX_VALUE

min(Numeric *b*)

```
min( b)
```

Returns the smaller of two int values. That is, 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.

Parameters:

- *b* - an argument.

See Also: Long.MIN_VALUE

format("*format-string*")

```
format("format-string")
```

Converts the value to a string representation defined by a format string. 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 54: Formatting symbols

Character	Description
*****	Fills with asterisks any position that would otherwise be blank.
&	Fills with zeros any position that would otherwise be blank.
#	This does not change any blank positions in the display.
<	Causes left alignment.
, (comma)	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.
. (period)	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.
-	Displays a minus sign for negative numbers.
$	This is the placeholder for the front specification of DBMONEY or DBFORMAT.
(	Displayed as left parentheses for negative numbers (accounting parentheses).
)	Displayed as right parentheses for negative numbers (accounting parentheses).

Examples

1. This example converts the number 65 to "A", its unicode character representation: `65.toChar()`
2. This example formats the numeric value of overall total as a string: `overalltotal.format("###.##")`

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 55: Class Methods (Static Member Methods)

Name	Description
`getDebugLevel()`	Numeric, returns the current debug level specified in the environment variable GREDEBUG, or 0 if no debug level was set.
`getEnvironmentVariable(` `  String` *`variableName`*`)`	Returns a String containing the value of the specified environment variable.
`getOutputDeviceName()`	Returns a STRING indicating the output device name selected in the API call fgl_report_selectDevice. Can be one of the following: • PDF • XLS • XLSX • HTML • Image • Printer • Postscript • SVG • Browser • RTF • OORTF For more information on fgl_report_selectDevice, refer to the *Genero Report Writer User Guide*.
`getPrinterMediaName()`	Returns a STRING indicating the media name specified in the API call fgl_report_setPrinterMediaName.
`getPrinterMediaSizeName()`	Returns a STRING indicating the media size name specified in the API call fgl_report_setPrinterMediaSizeName.
`getPrinterMediaTray()`	Returns a STRING indicating the media tray name specified in the API call fgl_report_setPrinterMediaTray.
`getPrinterName()`	Returns a STRING indicating the printer name specified in the API call fgl_report_setPrinterName.
`getSVGPaperSource()`	Returns a STRING indicating the paper source specified in the API call fgl_report_setSVGPaperSource; can be:

Name	Description
	"Auto", "Cassette", "Envelope", "EnvelopeManual", "FormSource", "LargeCapacity", "LargeFormat",

Name	Description
	"Lower", "Middle", "Manual", "OnlyOne", "Tractor" or "SmallFormat"
`getSVGPrinterName()`	Returns a STRING indicating the printer name specified in the API call fgl_report_setSVGPrinterName.
`producingBrowserOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "Browser"; otherwise FALSE.
`producingExcelOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "XLS" or "XLSX"; otherwise FALSE.
`producingHTMLOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "HTML"; otherwise FALSE.
`producingImageOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "Image"; otherwise FALSE.
`producingOORTFOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "OORTF"; otherwise FALSE.
`producingPDFOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "PDF"; otherwise FALSE.
`producingPostscriptOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "Postscript"; otherwise FALSE.
`producingRTFOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "RTF"; otherwise FALSE.
`producingSVGOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "SVG"; otherwise FALSE.
`producingXLSOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "XLS"; otherwise FALSE.
`producingXLSXOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "XLSX"; otherwise FALSE.
`producingPrinterOutput()`	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice wasis "Printer"; otherwise FALSE.
`producingForPreview()`	BOOLEAN; returns TRUE if preview was selected in the API call fgl_report_selectPreviewis selected; otherwise FALSE.
`xlsPagesAreMerged()`	BOOLEAN, returns TRUE if page merging was selected in the API call fgl_report_configureXLSDevice; otherwise FALSE.

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()"
```

The String Class

Details about the String class and its public members.

- Syntax
- Methods
- Usage and Examples

Syntax

```
String
```

Methods

Table 56: Object Methods

Name	Description
`charAt( index Numeric ) RETURNING result String`	Returns a String containing the character at the specified *index* in the current String.
`contains( s String ) RETURNING result Boolean`	Returns a Boolean value (TRUE/FALSE) specifying whether *s* is contained within the current String.
`endsWith( s String ) RETURNING result Boolean`	Returns a Boolean value (TRUE/FALSE) specifying whether the current String ends with *s*.
`equals( s String ) RETURNING result Boolean`	Returns a Boolean value (TRUE/FALSE) specifying whether *s* matches the current String. If one of the Strings is null the method returns FALSE.
`equalsIgnoreCase( s String ) RETURNING result Boolean`	Returns a Boolean value (TRUE/FALSE) specifying whether *s* matches the current String, ignoring

Name	Description
	character case. If one of the Strings is null the method returns FALSE.
`format ( number Numeric, format Enum )` `  RETURNING result String`	Sets the format of the page number string for a PAGENOBOX. The value for format can be: ARABIC, LOWERROMAN, UPPERROMAN
`indexOf( s String )` `  RETURNING result Numeric`	Returns a Numeric value representing the index of *s* within the current String.
`indexOf( s String , index Numeric )` `  RETURNING result Numeric`	Returns a Numeric value representing the index of *s* within the current String, starting from byte position *index*. Returns zero if the substring was not found. Returns -1 if the current String is null.
`isEmpty()` `  RETURNING result Boolean`	Returns a Boolean value. Returns true if the current string has a length of zero (`length()=0`), otherwise false.
`isNull()` `  RETURNING result Boolean`	Returns a Boolean value. Returns true if the current string has a length of zero (`length()=0`) and the string is tagged as null, otherwise false. This is the case for null valued input variables read from the input stream. For backward compatibility, null values do not have special behavior when used with the various operators. Specifically an input variable of type string that is null behaves like the empty string.
`lastIndexOf( s String )` `  RETURNING result Numeric`	Returns a Numeric value representing the index within the current String of the last occurrence of *s*, searching backward.
`lastIndexOf( s String , index` ` Numeric )` `  RETURNING result Numeric`	Returns a Numeric value representing the index within the current String of the last occurrence of *s*, searching backward. starting at the specified position *index*.
`length()` `  RETURNING result Numeric`	Returns a Numeric value representing the length of the current String.
`matches( s String )`	Returns a Boolean value specifying whether the current String matches the regular expressions.

Name	Description
`RETURNING result Boolean`	
`replace( old String, new String )`	Replaces *old* in the current String with *new*.
`replaceAll( old String, new String )`	Replaces every occurrence of *old* in the current String with *new*.
`replaceFirst( old String, new String )`	Replaces every occurrence of *old* in the current String with *new*.
`startsWith( s String )` `RETURNING result Boolean`	Returns a Boolean value specifying whether the current String begins with *s*.
`startsWith( s String , index Numeric )` `RETURNING result Boolean`	Returns a Boolean value specifying whether the substring in the current String beginning at the position *index* contains *s*. The first byte of a string is at position 0.
`subString( sindex Numeric, eindex Numeric )` `RETURNING result String`	Returns the substring starting at byte position *sindex* and ending at the character at *eindex*-1. The first byte of a string is at position 0.
`subString( index Numeric )` `RETURNING result String`	Returns the substring starting at the position *index* in the current String. The first byte of a string is at position 0.
`toLowerCase()`	Converts the current String to lowercase.
`toString()`	Converts the current Numeric value to a String.
`toUpperCase()`	Converts the current String to uppercase.
`translate()`	Uses the current String as the key for a lookup in Genero localization files; returns any entry found, otherwise returns the current String itself.

Name	Description
`RETURNING result String`	
`trim()`	Removes white space characters from the beginning and end of the current String.
`trimCompress()`	Removes white space characters at the beginning and end of the current String, as well as any contained white space.
`trimLeft()`	Removes white space characters from the beginning of the current String.
`trimRight()`	Removes white space characters from the end of the current String.
`urlEncode()` `RETURNING result String`	Returns a URL encoding of the current String.

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 4GLdata 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
```

Upgrading Genero Report Designer

These topics talk about what steps you need to take to upgrade to the next release of Genero Report Designer, and allows you to identify which features were added for a specific version.

- What's new in Genero Report Designer, v 3.00 on page 6
- What's new in Genero Report Designer, v 2.50 on page 198

Upgrading reports from prior versions

The `gsreport` command line utility updates report design documents (`.4rp`) from previous versions to the current version.

Syntax

```
gsreport [OPTIONS] filename [filename [...]]
```

Table 57: gsreport options

Option	Description
`-h`	Show help.
`-V`	Display program name and version.
`-c`	Convert old format 4rp files to new format
`-encoding` *ENCODING*	Sets the encoding to *ENCODING*. **Note:** When you use the `-h` option to display command help, the default encoding for your current environment displays.
`-translate` *LOCALE*	Sets the translation file to *LOCALE*.

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.

What's new in Genero Report Designer, v 3.00

This publication includes information about new features and changes in existing functionality.

These changes and enhancements are relevant to this publication.

Table 58: Genero Report Designer, Version 3.00

Overview	Reference
Genero Report Designer provides a LastPageFooter section property.	See section (Section) on page 108.
Support of Intelligent Mail bar code type.	See intelligent-mail on page 150.
New `smartParse` bar code property for bar code Code-128. When enabled, this allows you to enter the bar code value, and the internal code will be computed for you resulting in the shortest visual representation.	See smartParse (Smart Parse) on page 109 and code-128 on page 124.
New gs1* bar code aliases.	See Bar Code type listing on page 120.

What's new in Genero Report Designer, v 2.50

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

Overview	Reference
Tables support.	See Working with tables on page 43
Pivot tables support.	See Working with Pivot Tables on page 56.
Distributed mode. Allows the report engine to be started as a daemon to which Genero applications can connect to process the reports, allowing for vastly faster processing for short documents and improved scalability.	See Distributed Mode.
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.	No further reference.
Null value support.	See The String Class on page 192 and The Numeric Class on page 178, Conditional Expressions.
Improved trigger updates. Algorithm improved to remove the need for frequent manual adjustments for each change within the data schema (`rdd`) file.	See Triggers on page 19.
Runtime localization. Report can now be localized independent of the language settings of the application.	See Change localization settings at runtime and fgl_report_configureLocalization.
QR code barcode support.	See qr-code on page 152.

Overview	Reference
Display position of footers. Layout nodes designated as footers display at the bottom of the Mini Page, providing a WYSIWYG experience for the report designer.	See Page headers and footers on page 21.
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.	See Adding data values and captions on page 17.
Splitting of oversized elements across pages to prevent overfill.	See splitOversizedItem (Split Oversized Items) on page 110.
Rotation of items. The transformTransparently property changes the effect of the properties layoutDirection and swapX. When set, the transformation extends to the entire fragment so that entire documents can be rotated.	See transformTransparently (Transform transparently) on page 113.
Backside printing support.	See Backside printing on page 43.
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.	See sortBy (Sort By) on page 110 and sortAscending (Sort Ascending) on page 110.
Fallback image support when the requested image for an Image Box is not found.	See Image Box on page 81.
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.	See Place a trigger within the report structure on page 21.
Class property added for report elements.	See class (Class) on page 95.
Display and modify the sizing policy of containers.	See Modify the sizing policy of containers on page 14.
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.	See Business Graphs on page 85.
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.	See layoutDirection (Layout Direction) on page 104.
Preference added to control the appearance of RTL expressions in the document view.	See Customize Report Designer: preferences on page 161.
Added options to facilitate the mass generation of images that are sized by their content (e.g. for web sites).	See fgl_report_setImageUsePageNamesAsFileNames and fgl_report_setImageShrinkImagesToPageContent.