BIRT Report Object Model – Report Design

Functional Specification

Draft 1: November 29, 2004

Abstract

Defines the properties of the report design itself. Identifies the contents of a BIRT report design.

Document Revisions

Version	Date	Description of Changes
Draft 1	11/29/2004	First BIRT release.

Contents

1. Introduction

The report design element represents the overall design file. It provides a wide variety of contextual and setup information. BIRT reports can include libraries of reusable components. Libraries and report designs contain much of the same information; the base Module element describes this similarity.

2. Modules

BIRT modules include designs and libraries. The module element describes the properties common to both.

2.1 Module Element

Report design information common to report designs and libraries. This is most of the report except for the body.

Summary

Availability: First release

XML Summary

Base Element: Design Element

Properties

`author`

The person who created the report.

`colorPalette`

A set of custom color names.

`comments`

Text that describes the report design.

`createdBy`

Name of the tool that created the design.

`description`

A description presented to the end user of report when it is deployed.

`helpGuide`

External file that provides help information for the report.

`includeLibraries`

The name(s) of BIRT library files for this report.

`includeScripts`

The name(s) of JavaScript files to include into this report.

`title`

A short description of the report presented to the user when the report is deployed.

`units`

The units used for measurements within the report.

`translations`

The list of externalized messages specifically for this report.

`images`

A list of images embedded in this report.

`pageSequences`

A list of page sequences that describe how to create a report with multiple master pages.

Methods

`initialize`

Called when the report starts executing in either the Factory or Presentation engine.

Contents

The report design contains a variety of elements that make up the report. These include:

`parameters`

A list of Parameter elements that describe the data that the user can enter when running the report.

`dataSources`

The connections used by the report.

`dataSets`

Data sets (queries) defined in the design.

`styles`

User-defined styles used to format elements in the report. Each style must have a unique name within the set of styles for this report.

`pageSetup`

The layout of the master pages within the report.

`components`

Reusable report items defined in this design. Report items can extend these items. Defines a “private library” for this design.

Description

The module element describes the common characteristics between libraries and designs. It provides a wide variety of properties to help people understand the module, and provide lists of most components: page setup, data sources, data sets, reusable report items, and so on. It also provides a list of custom color definitions.

See Also

Report Design element

Library element

2.1.1 `author` Property

The person who created the report.

Summary

Display Name: Author

ROM Type: String

JavaScript Type: String

Default value: None

Settable at runtime: No

Availability: First release

Description

The person who created the report. The user can fill this in to keep track of the person who wrote the report, or is responsible for maintaining the report. This is purely documentation; BIRT does not use this property at runtime.

See Also

Module element createdBy Property

2.1.2 `comments` Property

Text that describes the report design.

Summary

Display Name: Comments

ROM Type: HTML structure

JavaScript Type: `PropertyStructure` object

Default value: None

Availability: First release

Description

Text that describes the report design. It can contain embedded HTML. The comments are notes for the people who develop and maintain the report. Comments cannot be externalized. BIRT does not use this information at runtime, it is purely documentation for the report developer.

See Also

See the `helpGuide` and the `description` properties for a way to provide information for the users of the report.

2.1.3 `createdBy` Property

Name of the tool that created the design.

Summary

Display Name: Created By

ROM Type: String

JavaScript Type: String

Default value: None. (Reports created by BIRT set this string to identify BIRT ERD as the creator.)

Settable at runtime: No

Availability: First release

Description

Name of the tool that created the design. Designs created by BIRT will contain the text “BIRT Version xxx”. Customers or third parties who generate designs can insert their product name here. This is purely a documentation property; BIRT does not use it at runtime.

See Also

`author` property

2.1.4 `description` Property

A description presented to the end user of report when it is deployed.

Summary

Display Name: Description

ROM Type: Static HTML structure

JavaScript Type: `PropertyStructure` object

Default value: None

Settable at runtime: No

Availability: First release

Description

A description presented to the end user of report when it is deployed. Can contain HTML. Can be externalized. This is a full description, and can be shown to the user when using the report in the UI. The actual use of this property is determined by the application UI.

See Also

`title` property

`helpGuide` property

2.1.5 `helpGuide` Property

External file that provides help information for the report.

Summary

Display Name: Help Guide

ROM Type: URL

JavaScript Type: String

Default value: None

Settable at runtime: No

Availability: First release

Description

The help guide is an external file that provides help information for the report. Help information can include detailed information about parameters, report content or other information. The file can be in any format supported by the browser. The help guide will open in a separate popup window.

See Also

`description` property

`title` property

2.1.6 `includeLibraries` Property

The name(s) of BIRT library files for this report.

Summary

Display Name: Include

ROM Type: String List

JavaScript Type: Array of String

Default value: None

Settable at runtime: No

Availability: After the first release

XML Summary

Element name: `<include-library>`*`libName`*`</include-library>`

Type: `string`

Description

The name(s) of BIRT library files for this report. A library can include reusable components, styles, page setup, scripts, and so on. A report design can also be used as a library. If so, the Body portion of the included design is ignored.

See Also

Library element

2.1.7 `includeScripts` Property

The name(s) of JavaScript files to include into this report.

Summary

Display Name: Include Script

ROM Type: String List

JavaScript Type: Array of String

Default value: None

Settable at runtime: No

Availability: After the first release

XML Summary

Element name: `<include-script>`*`scriptName`*`</include-script>`

Type: `string`

Description

The name(s) of JavaScript files to include into this report. BIRT reads and executes these scripts before calling the `initialize` method for the design. The scripts can define reusable variables, functions, objects, Java imports and so on. The design can contain any number of script includes.

See Also

`initialize` method

2.1.8 `title` Property

A short description of the report presented to the user when the report is deployed.

Summary

Display Name: Title

ROM Type: Text structure

JavaScript Type: `PropertyStructure` object

Default value: None

Settable at runtime: No

Availability: First release

Description

A short description of the report presented to the user when the report is deployed. The string can be externalized. The use of the string depends on the application UI used at deployment.

See Also

`description` property

`helpGuide` property

`title` property

2.1.9 `units` Property

The units used for measurements within the report.

Summary

Display Name: Units

ROM Type: Choice

JavaScript Type: `String`

Default value: None

Required.

Settable at runtime: No

Availability: First release

Choices

Display Name	XML Name
Inches	in
Millimeters	mm
Centimeters	cm
Points	pt

Description

The units used for measurements within the report. This property is set for a new report based on a user preference. The value should never be changed; doing so will invalidate dimensions that don’t have explicit units set.

See Also

Dimension property type

2.1.10 `colorPalette` Property

A set of custom color names.

Summary

Display Name: Color Palette

ROM Type: List of Custom Color structures

JavaScript Type: Array of `PropertyStructure` objects

Default value: None

Settable at runtime: No

Availability: First release

Description

The developer can define a set of custom color names as part of the design. The developer can then reference these names within properties. Defining the colors in the palette helps the developer achieve a consistent look throughout the report, and allows colors to be refined by changing just one place.

Every custom color has three parts: a display name, an internal name and an RGB value. The display name is what the developer sees. If the palette will be used by people in different countries, then the display name can be externalized and translated. The internal name identifies the color within the design and in the XML design file. The internal name cannot be localized. Finally, the RGB value gives the actual color.

See Also

Custom Color structure

2.1.11 `translations` Property

The list of externalized messages specifically for this report.

Summary

Display Name: Translations

Rom Type: List of Resource structures

JavaScript Type: Array of `PropertyStructure` objects

Default value: None

Settable at runtime: No

Availability: First release

Description

Many companies have customers, employees or suppliers in many different countries. The Translations property allows a report to externalize its text then localize it to any locale. The Translations property is used in conjunction with a resource key in a property that displays static text. The resource key identifies a resource entry within the translations list. Each resource provides a list of locale/string pairs. For example, a label that identifies a name may be tagged with a “Name” resource key. The Translations property has an entry for “Name”. This entry contains locale-name pairs such as:

en/Name

fr/Nom

sp/Nombre

Resources can also be put into an external resource file, though the external file is not supported in the first release.

See Also

Text structure

2.1.12 `images` Property

A list of images embedded in this report.

Summary

Display Name: Images

ROM Type: List of Embedded Image structures

JavaScript Type: Array of `PropertyStructure` objects

Default value: None

Required.

Settable at runtime: No

Availability: First release

Description

A design often uses images. Most images in a report will be accessed via a URL, or a file deployed externally to the report. However, some reports may want to embed an image directly into the report design. Embedded images are especially helpful if a report design is generated in a custom application and sent to the server for one-time execution.

Images have a name. The name allows an image item to reference the embedded image. Scripts can use the image name to retrieve the image from the images array:

```
var img = report.design.images[“MyImage”]
```

See Also

Image item

2.1.13 pageSequences Property

Instructions for creating a report with multiple master pages.

Summary

Display Name: Page Sequences

ROM Type: List of Page Sequence structures

JavaScript Type: Array of `PropertyStructure` objects

Default value: None

Settable at runtime: No

Availability: After the first release

Description

Page sequences describe how to create a report with more than one master page. For example, a report to be printed and bound may want to use different master pages for the left and right pages. See the *ROM Page Setup Specification* for details on master pages and page sequences.

2.1.14 `initialize` Method

Called when the report starts executing in the BIRT Report Engine.

Synopsis

`design.``initialize( )`

Context: Startup

Availability: First release

Description

BIRT calls this method before calling any other method in the design.

BIRT calls this method at the start of the Factory and Presentation engines, just after loading the imports, before opening the report document. Implement this method to define global functions, objects, Java imports and other resources used throughout the report.

Libraries can also contain an init script and imports. BIRT executes these scripts in the following order:

- For each library, in the order that the library is included:
  - Locate, read and execute the imported scripts in the order they appear in the Code Modules element.
  - Execute the init script.

- For the report design itself.
    - Locate, read and execute the imported scripts in the order they appear in the Code Modules element.
    - Execute the init script.

In general, the `initialize` method of a report design or library can reference resources defined in the `initialize` script of any included libraries, but not visa-versa. This is the generally accepted meaning of an include.

See Also

Report Design `beforeFactory` method

Report Design `beforeRender` method

2.1.15 `parameters` Slot

A list of Parameter elements that describe the data that the user can enter when running the report.

Summary

Display Name: Parameters

Cardinality: Multiple

Required.

Availability: First release

Contents

This slot can contain parameters and parameter groups. Contents include:

Content Item	Description
Parameter Group	Organizes parameters visually in the requester UI.
Scalar Parameter	Requests a simple data value such as a string, number, date and so on.
Filter Parameter	Allows the user to define a filter condition that restricts that data that will appear in the report.
List Parameter	Requests a list of simple values.
Table Parameter	Requests a list of rows, in which each row includes a set of name/value pairs.

XML Summary

Element name: `parameters`

Description

A list of parameter elements that describe the data that the user can enter when running the report. BIRT supports four kinds of parameters identified above. The design can also include parameter groups that organize parameters visually.

The order of parameters within this slot in the design file determines the order that they will appear in the requester UI.

The parameters property provides a “flattened” list of parameters. Each parameter identifies the group (if any) to which it belongs.

Parameter values are also available using the `params` global variable.

2.1.16 `dataSources` Slot

The connections used by the report.

Summary

Display Name: Data Sources

Cardinality: Multiple

Availability: First release

Contents

Content Item	Description
Script Data Source	A data source implemented within the application.
JDBC Data Source	A connection to a JDBC data source.
DAX Data Source	A connection to an external data source defined using the Data Access Extension system.

XML Summary

Element name: `data-sources`

Description

This slot holds the list of data sources defined within the report itself. Each data source must have a unique name. The order of items within this slot is unimportant.

2.1.17 `dataSets` Slot

Data sets (queries) defined in the design.

Summary

Display Name: Data Sets

Cardinality: Multiple

Availability: First release

Contents

Content Item	Description
Script Data Set	A data set defined within the application itself.
JDBC Select Data Set	A data set that retrieves results from an SQL SELECT statement executed via a JDBC connection.
DAX Data Set	A data set defined using the Data Access Extension features of BIRT.

XML Summary

Element name: `data-sets`

Description

This slot holds the list of data sets defined within the report itself. Each data set must have a unique name. The order of items within this slot is unimportant.

2.1.18 `styles` Slot

List of user-defined styles used to format elements in the report.

Summary

Display Name: Styles

Cardinality: Multiple

Availability: First release

Contents

Contains Style elements that define values for visual properties.

XML Summary

Element name: `styles`

Description

User-defined styles used to format elements in the report. Each style must have a unique name within the set of styles for this report.

2.1.19 `pageSetup` Slot

Provides the overall description of the pagination rules for the report.

Summary

Display Name: Page Setup

Cardinality: Multiple

Availability: First release (limited, see below.)

Contents

The report design contains a variety of elements that make up the report. These include:

Content Item	Description
Simple Master Page	A master page that contains just a size, header and footer.
Graphic Master Page	A master page that contains free-form page decoration, multiple-columns and other advanced features.
Simple Page Sequence	Repeats a single page with an optional different initial page.
Alternating Page Sequence	Alternates between two different master pages to achieve left/right layout for bound reports.

The page setup can contain zero or more master pages (of either type) and zero or more page sequences (of either type). The simplest report provides no master page. It will appear with the default pages size and margins, and with no page header or footer.

XML Summary

Element name: `page-setup`

Description

The page setup defines the way that the report will appear when printed. It consists of a master page that defines the page size, page “decoration”, margins and so on. Some reports need to use different master pages, perhaps to have the first page of the report appear in letterhead, with the remaining pages on plain paper. Other reports may be printed and bound, requiring differing page layout for the left and right pages.

A report can omit the page setup information. If so, page setup is taken from a library, if provided. If the library is not provided, or does not have a page setup, or the master page does not have page dimensions, then the setup comes from BIRT preferences, or from locale-specific defaults.

If the user lists multiple master pages, but no page sequences, then BIRT uses the first master page by default. If the user defines one or more page sequences, then BIRT uses the first page sequence by default. Other than these two cases, the order of elements within the page setup slot is unimportant.

This slot defines a name space; the name of each master page and page sequence must be unique across the set of other page setup elements.

Only the simple master page is available in the first release, and the design can contain at most one simple master page in the first release.

2.1.20 `components` Slot

Defines report items used in multiple places within this report.

Summary

Display Name: Components

Cardinality: Multiple

Availability: First release

Contents

This slot can contain any visual report item: data items, lists, tables, text items and so on.

XML Summary

Element name: `components`

Description

This slot defines reusable report items defined in this design. Report items can extend these items.

A report developer may want to use the same image, text, label or other item in several places within in the report. Instead of copying & pasting, the developer can instead define the element once in the components slot. Then, he can use the component to create the various instances.

Using a reusable component helps reduce maintenance costs because changes are made in only one place, rather than across several copies.

This slot defines a “private library” for this design. Components that are used by several reports appear in a “shared” library file.

The following rules apply to the components slot:

- All report items in the component area must have a name. (This rule applies only to items directly in the component area, not to items nested inside of other items.)
- All report items share a single name space, including those in the Components and Body.
- Items in the Components area can extend from other items in the Components area of the same type. In the design file, the base components must physically appear before derived items.
- Any type of report item can appear in the Components area.
- Only report items can appear in the components area. Master pages, styles, data sets, data sources, parameters and other elements cannot appear.
- Report items in the components area can contain other items if their definition allows. For example, a grid can contain other items.

See Also

Library element

3. Structures

This section describes structures used by the module element.

3.1 Custom Color Structure

Defines a custom color name.

Summary

Availability: First release

Properties

`displayName`

The user-visible, translatable name of the color

`name`

The internal, non-translated name of the color.

`rgb`

The RBG value of the color.

Description

The developer can define a set of custom color names as part of the design. The developer can then reference these names within properties. Defining the colors in the palette helps the developer achieve a consistent look throughout the report, and allows colors to be refined by changing just one place.

Every custom color has three parts: a display name, an internal name and an RGB value. The display name is what the developer sees. If the palette will be used by people in different countries, then the display name can be externalized and translated. The internal name identifies the color within the design and in the XML design file. The internal name cannot be localized. Finally, the RGB value gives the actual color.

3.1.1 `displayName` Property

The user-visible, translatable name of the color

Summary

Display Name: Display Name

ROM Type: Text structure

JavaScript Type: `PropertyStructure` object

Default value: None

Settable at runtime: No

Availability: First release

Description

A custom color can contain a localizable display name. This is most useful for colors defined in a library and that will be used within the web report designer.

If the design does not provide a display name, then the color name appears in the UI.

See Also

Text Structure

3.1.2 name Property

The internal, non-translated name of the color.

Summary

Display Name: Name

ROM Type: Name

JavaScript Type: String

Required.

Settable at runtime: No

Availability: First release

Description

Each color requires an internal name. This is the name used in a color property when referencing the custom color. Color names must be unique, and must be unique with respect to the 17 standard CSS color names.

See Also

Style element `color` property

Style element `backgroundColor` property

3.1.3 `rgb` Property

The RGB value of the color.

ROM Summary

Display Name: RGB Value

ROM Type: String

JavaScript Type: String

Required.

Settable at runtime: No

Availability: First release

Description

Defines the color. The color can be defined using any of the following formats:

- CSS absolute: `rgb( R, G, B )`, with each value in decimal
- CSS relative: `rgb( R%, G%, B% )`
- HTML-style: `#RRGGBB`
- Java-style: `0xRRGGBB`
- Decimal

3.2 Resource Structure

The resource structure defines a localized message within the report design itself.

Summary

Availability: First release

Properties

`key`

Name of the resource.

`translations`

List of locale/string pairs.

Description

Each resource structure defines one resource. A resource consists of a resource key, and a set of translations. The resource key identifies the resource within the report. For example, to localize the name of a label, set the label's resource key property to the name of a resource.

3.2.1 key Property

The name of the resource.

Summary

Display Name: Resource Key

ROM Type: Name

JavaScript Type: String

Required.

Settable at runtime: No

Availability: First release

Description

This is the name of the resource. Resource names must be unique within the design. The same resource name can appear in both a design and a library. In this case, BIRT first searches the resource in the design, then in a library. If the resource is still not found, BIRT will then search in an external resource bundle, if provided.

3.2.2 `translations` Property

List of localized strings.

Summary

Display Name: Translations

ROM Type: List of Translation structures

JavaScript Type: Array of `PropertyStructure` objects

Default value: None

Settable at runtime: No

Availability: First release

Description

This property provides the localized strings in the form of locale/string pairs. The locales must be unique within the translation list. A blank locale is allowed, this is the default string used if no specific locale match can be found.

See Also

Translation Structure

3.3 Translation Structure

Defines a localized value for a resource and locale.

Summary

Availability: First release

Properties

locale

The locale for this translation.

string

The translated string value.

Description

Each translation structure provides a locale/string pair.

3.3.1 `locale` Property

The locale for this translation.

Summary

Display Name: Locale

ROM Type: Name

JavaScript Type: String

Default value: Blank

Availability: First release

Description

The locale name is one of those supported by Java. Java locales have three parts: a language, a country and a variant. BIRT uses the same locale lookup rules as Java. See Java documentation for details on the available locales and the search algorithm.

The locale name must be unique within the resource. The list can include both a language "en" and a language/region combination: "en_US", "en_UK".

One of the locales can be blank. If so, then BIRT uses this translation if the search would otherwise fail.

3.3.2 `string` Property

The translated string value.

Summary

Display Name: String

ROM Type: String

JavaScript Type: String

Default value: None

Availability: First release

Description

This property provides the translated string value. The string can be blank, in which case no string will be displayed for this locale. The string can contain HTML. HTML is useful when localizing the contents of a text item. If HTML is used for a string that does not support HTML, then the resulting behavior is undefined.

3.4 Embedded Image Structure

Holds an image embedded within the design file.

Summary

Availability: First release

Properties

name

The name of the image.

type

The format of the image.

data

The binary data for the image.

Description

BIRT allows images to be embedded within the design file. This is most useful for reports created in an external application and submitted to a server for immediate, one-time execution. If report designs are created with the BIRT designer, and are saved on disk, the developer will probably find it easier to use external images referenced with a file name or URL.

3.4.1 name Property

The name of the image.

Summary

Display Name: Name

ROM Type: Name

JavaScript Type: String

Required.

Settable at runtime: No

Availability: First release

Description

Provides a name for the image. The image name is used to reference the image in an image item. Image names must be unique within the report. The image name can be any legal name, it need not be related to the original image name on disk. That is, “Logo” is a fine name, it need not be “corplogo.jpg”.

3.4.2 `type` Property

The format of the image.

Summary

Display Name: Image Type

ROM Type: Choice

JavaScript Type: String

Default value: None

Settable at runtime: No

Availability: First release

Choices

The choices are a subset of the image MIME types.

Image Type	Meaning
(blank)	BIRT attempts to infer the image type from the first few bytes of the image itself.

Image Type	Meaning
image/jpeg	JPEG image
image/bmp	Windows bitmap
image/gif	Compuserve GIF format
image/png	Portable Network Graphics (PNG) format
image/x-png	??

Description

This property defines the type of the image using standard MIME types. The type can be omitted. In this case, BIRT attempts to identify the image format by looking at the first few bytes of the image. Many image formats provide a unique "magic number". However, this solution is not guaranteed to work for all images and all image formats. To avoid problems, the application should specify the image type whenever possible.

3.4.3 `data` Property

The binary data for the image.

Summary

Display Name: Image Data

ROM Type: Binary

JavaScript Type: TBD

Default value: None

Settable at runtime: No

Availability: First release

Description

This property contains the image itself as an array of bytes. The data is stored in the XML design file using base-64 encoding.

4. Report Design

4.1 Name Spaces

ROM defines a number of *name spaces* with a report design. A name space holds a set of named components. Each name space is independent of the other. ROM name spaces are:

Name Space	Contents
Styles	Shared styles
Data Sources	Data sources (database connections)
Data Sets	Queries and other data sources
Master Pages	Page definitions.
Layout Elements	All visual elements such as lists, tables, etc.
Parameters	Report parameters

The name is optional for elements that can appear in the Layout Elements name space. The name is required for all other elements.

4.2 Report Design Element

The report design element represents report as a whole.

Summary

Availability: First release

XML Summary

Element name: `report`

Base Element: Module Element

Scripting Summary

Runtime object: `report` (global variable)

Design object: `report.design`

Properties

The report design defines properties that describe the design as a whole. Report design properties do not inherit because a design cannot extend another design.

`base`

A base directory to use when computing relative links from this report.

`refreshRate`

The refresh rate when viewing the report.

Methods

`beforeFactory`

Perform initialization before opening the report document for writing.

`afterFactory`

Perform initialization before opening the report document for writing.

`beforeOpenDoc`

Called just before opening the report document file in the Factory. Allows the report to control the name and other properties of this file.

`afterOpenDoc`

Called just after opening the report document file in the Factory.

`beforeCloseDoc`

Called just before closing the report document file in the Factory. Allows the report to set additional properties on the file before closing the file.

`afterCloseDoc`

Called just after closing the report document file in the Factory. Allows the report to perform Factory cleanup.

`beforeRender`

Called before starting a presentation time action.

`afterRender`

Called after completing a presentation time action.

Contents

The report design contains a variety of elements that make up the report. These include:

Body

The visual layout sections within the report.

Description

The report design element contains the information about a report design. It represents the entire set of design information stored in a BIRT report design file.

See Also

Library element

4.2.1 base Property

A base directory to use when computing relative links from this report.

Summary

Display Name: Base

ROM Type: String

JavaScript Type: String

Default value: None

Settable at runtime: No

Availability: First release

Description

A base directory to use when computing relative links from this report. Most often used for reports created externally to the server and submitted on the fly. Used when computing hyperlinks, library includes, image includes and other relative file references. (See the W3C XML Base standard for background: http://www.w3.org/TR/2001/REC-xmlbase-20010627/).

4.2.2 `refreshRate` Property

The refresh rate when viewing the report.

Summary

Display Name: Refresh Rate

ROM Type: Integer

JavaScript Type: Number

Default value: 0 (the report does not refresh)

Settable at runtime: No

Availability: After the first release

Description

The refresh rate when viewing the report. The viewer will automatically rerun and redisplay the report. The rate is given in seconds. A rate of 0 means no refresh.

4.2.3 `beforeFactory` Method

Perform initialization before opening the report document for writing.

Synopsis

`*design*.beforeFactory( )`

Context: Factory

Availability: First release

Description

Called at the start of the Factory after the initialize( ) method and before opening the report document (if any).

See Also

Module element `initialize` method

`afterFactory` method

4.2.4 `afterFactory` Method

Perform initialization before opening the report document for writing.

Synopsis

`*design*.afterFactory( )`

Context: Factory

Availability: First release

Description

Called at the end of the Factory after closing the report document (if any). This is the last method called in the Factory.

See Also

`beforeFactory` method

4.2.5 `beforeOpenDoc` Method

Perform initialization before opening the report document for writing.

Synopsis

`design.beforeOpenDoc( )`

Context: Factory

Availability: After the first release

Description

BIRT calls this method just before opening the report document in the Factory. It allows the application to customize features of the report document such as its name, whether it is temporary, security, and so on. Code in this script will likely be specific to the deployment platform.

See Also

`afterOpenDoc` method

4.2.6 `afterOpenDoc` Method

Called just after opening the report document file in the Factory.

Synopsis

`design.afterOpenDoc( )`

Context: Factory

Availability: After the first release

Description

Called just after opening the report document file in the Factory.

See Also

`beforeOpenDoc` method

4.2.7 `beforeCloseDoc` Method

Called just before closing the report document file in the Factory.

Synopsis

`design.beforeCloseDoc( )`

Context: Factory

Availability: After the first release

Description

BIRT calls this method just after closing the report document in the Factory. It allows the application to customize features of the report document just before closing. Some deployment platforms require that security, document aging and similar properties be set after closing the document. Code in this method will likely be specific to the deployment platform.

See Also

`afterCloseDoc` method

4.2.8 `afterCloseDoc` Method

Called just after closing the report document file in the Factory.

Synopsis

`design.``afterCloseDoc( )`

Context: Factory

Availability: After the first release

Description

Called just after closing the report document file in the Factory. Allows the report to perform Factory cleanup.

See Also

`beforeCloseDoc` method

4.2.9 `beforeRender` Method

Called before starting a presentation time action.

Synopsis

`design.``beforeRender( )`

Context: Presentation

Availability: First release

Description

BIRT calls this method at the start of each presentation-time activity such as convert a page, print a report, extract data, etc. Presentation activities are usually done in repose to a client request coming into the Presentation Engine. Code is likely to be specific to the deployment platform.

BIRT may cache the same report session across multiple render actions. For example, if a user wants to view pages 1 and 2, see the table of contents, then do a search, the Presentation Engine may elect to keep the report loaded and the JavaScript session in effect. This allows an application to perform costly presentation-time initialization once, simply keep a flag that indicates whether initialization has already been done by another render action.

If the report is being run and rendered in a single pass, then this method is called after the call to `beforeFactory`. If the rendering operation is being done on a report previously created, then this method is called just after the initialize method.

The Presentation Engine may choose to cache a report design and script session across multiple render requests. In this case, each render operation will trigger a call to `beforeRender` and `afterRender`. The `initialize` method will be called only when the report is first loaded.

See Also

`initialize` method

`afterRender` method

4.2.10 `afterRender` Method

Synopsis

`*design*.afterRender( )`

Context: Presentation

Availability: After the first release

Description

BIRT calls this method at the completion of each presentation-time activity such as convert a page, print a report, extract data, etc. Code is likely to be specific to the deployment platform.

If the report is being run and rendered in a single pass, then the call to `afterFactory` will precede the call to `afterRender`.

See Also

`beforeRender` method

4.2.11 body Slot

The visual layout sections within the report.

Summary

Display Name: Body

Cardinality: Multiple

Required.

Availability: First release

Contents

The body contains any number of sections, in any order.

Content Item	Description
Chart	A business graphic such as a pie chart.
Data	A simple data item. (Used infrequently in this context.)
Extended Item	A user-defined item.
Free-form	Content with a free-form layout.
Grid	Content organized into a static table.
Image	An image.
Include	Content defined in another report design.
Label	A short, localizable string. (Used infrequently in this context.)
List	A collection of sections driven by rows from a data set.
Matrix	A cross tabulation.
Table	A table that gets its content from a data set.
Text	A block of formatted text.

Content Item	Description
TOC	The table of contents for this report.

XML Summary

Element name: body

Description

The body portion of a design contains a list of the visual report content. Content is made up of one or more sections. A section is a report item that fills the width of the page. Sections represent relatively independent divisions within the report. In CSS terminology, a section is a block-level element.

The simplest design contains a single section. However, sophisticated designs can contain any number of sections.

5. Libraries

Report developers divide into roughly two groups: those who develop reports for use by a single firm, and those that develop reports to be used by multiple customers. Consultants, OEMs and similar developers fall into the second group. They create reporting applications to be used by a range of third-party customers. Often, the ultimate customer wants to develop their own customer reports, and have those reports follow the style established by the OEM.

Historically it has been labor-intensive to create a suite of reports that look the same, and to provide that same look to OEM or e.Services customers. BIRT proposes to simplify this task though a library feature that is similar to the templates in Microsoft PowerPoint.

A BIRT report library is a collection of report components that can be used in a report design. Report components include reusable visual report items, data sources, data sets, parameters, translations and more. Libraries can also contain templates. A report template is an incomplete report design that has place holders that the user can fill in though a wizard or other UI.

A library is a customizable, pre-defined set of rules that helps a user quickly create a report with a specific look. A BIRT library includes both formatting information and rules that help you the developer create a report. A library is like a “skin” for a report; it takes the data, structure and layout of a report and applies formatting defined in the library to transform the report’s appearance.

A library may start with a “style sheet”: a set of common styles to apply to all reports within an application. The library can then add commonly-used data sources, reusable components, translated text, images and more.

5.1 Using Libraries

Libraries apply to the following activities:

- Creating a new report using a wizard. BIRT ERD uses the rules and styles in the library to create the report.

- Creating a new report using the web report designer.
- Creating a new report using the various APIs.
- When creating report components via the various wizards or manually. For example, if the user adds a new group or group heading, the frame automatically picks up the style specified by the library.

5.2 Including Libraries in a Design

A report design must explicitly identify any Libraries that it uses. The report names the library in the "include" property of the report design. A design can include any number of templates as long as the following holds:

- No duplicate element names can appear in the libraries. Duplicate names cause a fatal load-time error.
- If the libraries define styles of the same name, then the order of includes determines which style will be retained. The general rule is that only the last style of a given name is retained.
- If a library and the design both define the same style, then the style in the design takes precedence.
- Libraries can include other libraries. Elements in a library can extend only elements defined in that same library, or a library that that library explicitly includes.

5.2.1 Search Path

BIRT defines a library search path. The search path is the following:

- The directory given by the "base" property of the report design (if any.)
- If the base property is not set, then the directory that contains the source file (design or library.)
- A search path defined by a BIRT-based application. (Such applications include the BIRT ERD, Factory, Presentation Engine and Web-Report Designer.)

BIRT searches for a library as follows:

- If the file name is absolute, use it as-is.
- If the name is relative, compute an absolute name based on the directory that contains the referencing file. If found, use that file.
- If not found, take the first directory from the search path. Compute an absolute path to the library as above. If the library is found in this location, use it.
- If not found, repeat the above step with each of the directories in the search path.

5.2.2 Packaged Deployment

Customers may find it convenient to create a packaged form of a report design for deployment. This package may put the libraries into a zip file along with the design. The deployment code should modify the library property to reference the library using just its base name. The engine using the packaged file should substitute the zip file itself for the "directory that contains the source file" in the above section.

5.2.3 Non-Packaged Deployment

There may be cases in which a report is run on a server without being packaged. In this case, the same search algorithm as above is used, but substitute “encyclopedia folder” for “directory” in the above algorithms. That is, the included libraries must exist in the Encyclopedia where they can be found relative to the report, or using a server-wide BIRT search path.

5.2.4 On-the-Fly Deployment

We expect that customers will often create and execute report designs on the fly. For example, the application may provide an application-specific web page to define a report, create a design, and send the design to the server for execution. In this case, the report does not exist in the server, and there is no base folder to use when searching for libraries. Instead, the server should provide a search path to use for report designs submitted dynamically.

Alternatively, the application can set the Base property of the report to indicate a base folder for hyperlinks and includes. BIRT acts as if the report were deployed to this location.

5.3 Library Element

Describes a BIRT library.

Summary

Base element: Module element

Availability: After the first release

XML Summary

Element name: `library`

Description

The library element appears at the top of a BIRT reusable library file. The library adds no properties or content to the base module element.

See Also

Module Element

6. Report Parameters

Report parameters provide a way to pass data into a BIRT report. BIRT provides four kinds of parameters:

- Simple (scalar) data value
- Query condition
- List of simple values
- Table (list of structures)

Parameters define a “requester” UI that users see when asking to run reports. Parameters provide extensive support for a state-of-the-art UI including:

- Localized parameter names
- Grouping of parameters
- Custom controls for entering parameter data
- Static pick lists
- Data-set driven pick lists, including “cascading” pick lists
- User-defined attributes that can pass additional data to custom UI

Reports with more than a few parameters often seek to organize parameters into groups. The Parameter Group element exists to satisfy this need.

Note that BIRT defines several types of parameters. This specification discusses *report parameters*. BIRT also defines *data set input parameters* that pass data to a data set, *data set output parameters* that pass data out of a data set, and *nested report parameters* that pass data to another report. While this section uses the word “parameter” as shorthand for “report parameter”, the reader should think “report parameter” when the term “parameter” could be confused with other kinds of BIRT parameters.

6.1 Parameter Group Element

Visual grouping of report parameters.

Summary

Base element: Report Element

Availability: First release

XML Element Name: `parameter-group`

Inherited Properties

The parameter group is a kind of Report Element. Some of the inherited attributes have a special meaning.

`name`

The internal name of the group. The name must be unique among parameters and parameter groups. The user will see this name unless the display name is defined. The name cannot be externalized.

`displayName`

The name displayed to the end-user for the group.

`extends`

A parameter group can extend a group defined in a library. In this case, the report will include all parameters from the library group. The application cannot change the set of parameters within the group. If the application wishes to change the group; create a new group within the report and add the parameters that should appear for the report.

Properties

`helpText`

Additional pop-up help text associated with the group.

`startExpanded`

If the UI can expand and collapse groups, then the UI should assume that groups appear in “collapsed” form by default. Use this attribute to cause the group to appear expanded by default instead. Ignored if the UI does not support group expand/collapse.

Contents

`parameters`

The set of parameters that appear inside the group. The UI should ignore the group is empty. Parameters appear in the UI in the same order that they appear in this list.

Description

A parameter group creates a visual grouping of parameters. The developer controls the order that groups appear in the UI, and the order in which parameters appear in the group. A runtime UI may choose to allow the user to expand & collapse parameter groups independently.

6.1.1 helpText Property

Additional pop-up help text associated with the group.

Summary

Display Name: Help Text

ROM Type: Text structure

JavaScript Type: `PropertyStructure` object

Default value: None

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

Additional pop-up help text associated with the group. This text can explain the purpose of the group, or provide overall explanation of how a set of parameters work together. The help text can be localized.

6.1.2 `startExpanded` Property

Whether the group should start expanded in the UI, or start collapsed.

Summary

Display Name: Start Expanded

ROM Type: Boolean

JavaScript Type: Boolean

Default value: True

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

If the UI can expand and collapse groups, then the UI should assume that groups appear in “collapsed” form by default. Use this attribute to cause the group to appear expanded by default instead. Ignored if the UI does not support group expand/collapse.

6.1.3 `parameters` Slot

Summary

Display Name: Parameters

Cardinality: Multiple

Element name: `parameters`

Availability: First release

Contents

This slot can contain parameters and parameter groups. Contents include:

Content Item	Description
Scalar Parameter	Requests a simple data value such as a string, number, date and so on.
Filter Parameter	Allows the user to define a filter condition that restricts that data that will appear in the report.
List Parameter	Requests a list of simple values.
Table Parameter	Requests a list of rows, in which each row includes a set of name/value pairs.

Description

The set of parameters that appear inside the group. The UI should ignore the group is empty. Parameters appear in the UI in the same order that they appear in this list.

6.2 Base Parameter Element

The base parameter element defines properties common to all types of parameters.

Summary

Base element: Report Element

Availability: First release

Inherited Properties

The base parameter element is a kind of Report Element. Some of the inherited attributes have a special meaning.

`name`

The internal name of the parameter, inherited from Report Element. Expressions in the report reference the parameter using this name. The name must be unique among parameters and parameter groups. The user will see this name unless the display name is defined. The name cannot be externalized.

`extends`

A parameter can extend a parameter defined in a library, but not another parameter defined within the report design.

`displayName`

The prompt text to display in the UI. The display name can be externalized. If the display name is omitted, the Requester page will display the parameter name instead.

Properties

`hidden`

If true, the parameter will not appear in the Requester page. Allows the developer to create parameters for internal use, or for use by scripts. Parameters are visible by default.

`helpText`

Additional text to display for the parameter to explain how to use the parameter. The string can be externalized.

Description

The base parameter element defines properties common to all types of parameters. Parameters can be hidden, meaning that they will not appear in the UI. Parameters also provide additional pop-up help text that can explain the use of the parameter to the end user.

6.2.1 `hidden` Property

Summary

Display Name: Hidden

ROM Type: Boolean

JavaScript Type: Boolean

Default value: False

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

If true, the parameter will not appear in the Requester page. Allows the developer to create parameters for internal use, or for use by scripts. Parameters are visible by default.

See Also

`concealValue` property of the Scalar Parameter

6.2.2 `helpText` Property

Additional text to display for the parameter to explain how to use the parameter.

Summary

Display Name: Help Text

ROM Type: Text structure

JavaScript Type: `PropertyStructure` object

Default value: None

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

Additional text to display for the parameter to explain how to use the parameter. The string can be externalized.

6.3 Scalar Parameter Element

Defines a simple single-value parameter.

Summary

Base element: Base Parameter element

XML Element Name: `scalar-parameter`

Availability: First release

Properties

`dataType`

The data type for the parameter. The data type controls how the Requester formats, parses and validates the parameter. The default is text. Valid types are:

`concealValue`

If true, the UI hides the user's entry by displaying asterisks or similar characters. Often used for passwords.

`defaultValue`

The default value for the parameter.

`allowNull`

Whether the value of the parameter can be null. The default is false. The user must enter a value for the parameter if no default is provided.

`allowBlank`

If the parameter is a string field, this attribute says whether to allow a blank string value. The default is true.

format

Formatting instructions for the parameter value within the parameter UI.

`controlType`

The suggested type of UI control to use when displaying the parameter.

`alignment`

How the items should appear in the UI. Choices are auto (default), left, center or right.

`selectionList`

Defines a selection list for the parameter: static list of values from which the user can choose.

`dynamicList`

Defines a selection list produced by a database query.

`mustMatch`

If true, then the value that the user provides must match one of the values in the static or dynamic selection list.

`fixedOrder`

Whether to display the values in the order defined in the selection list, or whether to resort the list lexicographically based on the actual translated values.

`dataSet`

The name of the data set to execute to obtain a selection list dynamically.

`valueExpr`

An expression on the data set row to return the value of each choice in a dynamic list.

`labelExpr`

An expression on the data row to return the display value for each choice in a dynamic list.

Description

This element defines a single-value parameter. Scalar parameters can have selection lists.

Scalar parameters can provide a selection list. The list can be static (defined by the `selectionList` property) or dynamic (defined by the `dataSet`, `valueExpr` and `labelExpr` properties.) The user can be required to select a value from the list, or can enter a value not in the list (as controlled by the `mustMatch` property).

BIRT uses the following rules to determine the selection list:

- If the `dataSet` property is set, then the selection list is dynamic.
- Else, if the `selectionList` property has one or more entries, use that list.
- Else, assume the parameter has no selection list.

6.3.1 `dataType` Property

Summary

Display Name: Data Type

ROM Type: Choice

JavaScript Type: String

Default value: Text

Inherited: Yes

Settable at runtime: No

Availability: First release

Choices

Display Name	Internal Name	Description
String	`string`	Arbitrary Unicode text.
Decimal	`decimal`	Any type of number including currency. Fixed decimal, arbitrary precision.
Float	`float`	A scientific amount using the usual floating point notation.
Date-Time	`dateTime`	A date, time or combination of date and time.
Boolean	`boolean`	A simple true/false value.

Description

The data type for the parameter. The data type controls how the Requester formats, parses and validates the parameter.

Note that any type can optionally provide a choice list. And, the parameter can require that the user select one of the choices.

See Also

`selectionList` property

`defaultValue` property

6.3.2 `concealValue` Property

Hides the user's entry by displaying asterisks or similar characters.

Summary

Display Name: Conceal Value

ROM Type: Boolean

JavaScript Type: Boolean

Default value: False

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

This property hides the user's entry by displaying asterisks or similar characters. Often used for passwords. The default is to show the entry in plain text.

See Also

`hidden` Property

6.3.3 `defaultValue` Property

Provides a default value for the property.

Summary

Display Name: Default Value

ROM Type: any

JavaScript Type: any

Default value: None

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

The default value for the parameter. The default value can be an expression, but cannot reference any other parameters.

6.3.4 `allowNull` Property

Whether the value of the parameter can be null.

Summary

Display Name: Allow Null

ROM Type: Boolean

JavaScript Type: Boolean

Default value: True

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

Whether the value of the parameter can be null. The default is false. The user must enter a value for the parameter if no default is provided.

6.4 `allowBlank` Property

Whether to allow a blank string value.

Summary

Display Name: Allow Blank

ROM Type: Boolean

JavaScript Type: Boolean

Default value: True

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

If the parameter is a string field, this attribute says whether to allow a blank string value.

6.5 `format` Property

Formatting instructions for the parameter value within the parameter UI.

Summary

Display Name: format

ROM Type: Format String

JavaScript Type: String

Default value: See description.

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

Formatting instructions for the parameter value within the parameter UI. By default the UI should use the following rules:

Text – No formatting.

Number, Float, Date-Time – Default locale formatting.

The format string must be one of the valid BIRT format strings (see the Style element for a list.) The format is used by the UI to display the value after the user moves away from a field. For example, to format a number as a dollar amount.

6.5.1 `controlType` Property

The suggested type of UI control to use when displaying the parameter.

Summary

Display Name: Control Type

ROM Type: Choice

JavaScript Type: String

Default value: See below.

Inherited: Yes

Settable at runtime: No

Availability: First release

Choices

Display Name	Internal Name	Description
Text Box	`text-box`	The user can type in the value.
List Box	`list-box`	Requires a list of choices: static or dynamic. Shown as a list box if set of choices are fixed, combo-box if set of choices is open.
Radio Button	`radio-button`	Only allowed for a fixed set of choices.
Check Box	`check-box`	Only for Boolean values

Description

The suggested type of UI control to use when displaying the parameter.

The default depends on the parameter properties:

text-box – Default for non-Boolean values when no choices are available.

list-box – Default for parameters with choices.

check-box – Default for Boolean values.

The list-box control is rendered in the UI depending on the value of the `mustMatch` property. If `mustMatch` is true, then the user much choose a value from the list, and the control is rendered as a UI list box. However, if `mustMatch` is false, then the user can chose an item from the list, or type in a different value, and the control is rendered as a UI combo box.

See Also

`mustMatch` property

6.5.2 `alignment` Property

How the items should be horizontally aligned in the UI.

Summary

Display Name: Alignment

ROM Type: Choice

JavaScript Type: String

Default value: Auto

Inherited: Yes

Settable at runtime: No

Availability: First release

Choices

Display Name	Internal Name	Description
Auto	`auto`	Numbers are left aligned, all others are right aligned.
Left	`left`	Values are left-aligned.
Center	`center`	Values are centered.
Right	`right`	Values are right-aligned.

Description

How the items should appear in the UI.

6.5.3 `selectionList` Property

Defines a selection list for the parameter.

Summary

Display Name: Selection List

ROM Type: List of Selection Value structures

JavaScript Type: Array of `PropertyStructure` objects

Default value: None

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

The parameter selection list provides a developer-defined list of choices. Every choice has two parts: a choice and a label. The label can be externalized and appears in the UI. The choice is the value passed to the report. For example, labels may be “Open” and “Closed”, while the values are “0” and “1”.

The items in the list are of the type given by the parameter data type.

See Also

`mustMatch` property

`fixedOrder` property

`controlType` property

6.5.4 `dataSet` Property

The data set to use to obtain the selection list dynamically.

Summary

Display Name: Data Set

ROM Type: Element Reference

JavaScript Type: String

Default value: None

Inherited: Yes

Settable at runtime: No

Availability: After the first release

Description

The name of the data set to execute to obtain the list.

This element defines or references a query that defines a dynamic selection list for the parameter. The data set can reference other parameters, but only those that appear in the design before this parameter. Sorting can be done by the data set (if the `fixedOrder` property is true), or can be done by BIRT (if the `fixedOrder` property is false.)

The data set must return a column that contains the choice values. It may also contain a column that returns the labels for the values. All other columns are ignored. The choice is that passed to the report when run. The optional display value is shown to the user in the UI. For example, the display values might be “Residential”, “Commercial”, and “Government” while the corresponding parameter values are “R”, “C”, and “G”.

6.5.5 `valueExpr` Property

An expression that returns the parameter value for each row in the dynamic list.

Summary

Display Name: Value Expression

ROM Type: Expression

Expression Type: must match the parameter data type

JavaScript Type: any

Default value: The value of the first column in the data row.

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

An expression on the data row from the dynamic list data set that returns the value for the choice. If omitted, then BIRT uses the first column.

6.5.6 `labelExpr` Property

An expression that returns the display value for each row in the dynamic list.

Summary

Display Name: Label Expression

ROM Type: Expression

Expression Type: String

JavaScript Type: String

Default value: None

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

An expression on the data row from the dynamic list data set that returns the prompt for the choice. If omitted, then BIRT uses the value as the label.

6.5.7 `mustMatch` Property

Whether the user must select a value from a selection list, or can enter a value not in the list.

Summary

Display Name: Must Match

ROM Type: Boolean

JavaScript Type: Boolean

Default value: False

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

If true, then the value that the user provides must match one of the values in the list. If false, then the values in the list are a reference; the use can enter additional values as well. Ignored if the property does not have a selection list.

See Also

`selectionList` property

`dynamicList` property

6.5.8 `fixedOrder` Property

Whether to display the values in the order defined in the list, or whether to resort the list.

Summary

Display Name: Fixed Order

ROM Type: Boolean

JavaScript Type: Boolean

Default value: True

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

Whether to display the values in the order defined in the list, or whether to resort the list lexicographically based on the actual translated values. The default is true, and the selections are displayed in the order in which they are defined.

See Also

`selectionList` property

`dynamicList` property

6.6 Static Parameter Choice Structure

Describes one item in a static parameter selection list.

Summary

Availability: First release.

Properties

`value`

The parameter value for this choice.

`label`

An externalizable label for the choice.

Description

A static selection list defines a set of property choice values. Each choice provides two values: an internal value and a display value. For example, the display values might be “Residential”, “Commercial”, and “Government” while the corresponding parameter values are “R”, “C”, and “G”. Display values can be localized.

6.6.1 `value` Property

The parameter value for this choice.

Summary

Display Name: Value

ROM Type: any

JavaScript Type: any

Required.

Settable at runtime: No

Availability: First release

Description

The parameter value for this choice. This is the value passed to the report when the user selects this choice. The value must agree with the parameter data type. The value cannot be localized.

6.6.2 `label` Property

The display label for the choice.

Summary

Display Name: Label

ROM Type: Text structure

JavaScript Type: `PropertyStructure` object

Default value: None

Settable at runtime: No

Availability: First release

Description

An externalizable label for the choice. If omitted, then the internal value is shown as the label.

6.7 Filter Parameter Element

Defines a parameter that is a filter condition.

Summary

Base element: Base Parameter

XML Element Name: `filter-parameter`

Availability: After the first release

Description

Users often want to enter “ad-hoc” or “query-by-example” conditions for a report. For example, select all customers in the state of California that have a sales amount greater than $1 million, and that have purchased XYZ product. Such queries are hard to specify using static parameters. Most of the time the user won’t enter most of these conditions, but scalar parameters require a value. Filter parameters provide a solution; they allow the user to specify a condition. The condition can be applied to a data set, or to a specific report element. For example, the report for the above condition may use the filter to display only the customers that meet the above conditions. Or, it can use the filter to display all customers, but only chart those that match the criterial.

6.8 List Parameter Element

Defines a parameter that can accept multiple entries.

Summary

Base element: Base Parameter

XML Element Name: `list-parameter`

Availability: After the first release

Description

A list parameter allows the user to enter multiple values. It is much like a scalar parameter, except that the user can enter several values. For example, a scalar parameter for a state would let the user enter just one state. A list parameter would let the user enter multiple states.

6.9 Table Parameter Element

Defines a parameter that is a list of tuples.

Summary

Base element: Base Parameter

XML Element Name: `table-parameter`

Availability: After the first release

Description

A table parameter provides a list of records, where the records define a fixed set of fields.

7. Deployment

BIRT supports embedded deployment. An embedded solution deploys the BIRT Report Engine onto an application server. The details of the actual deployment are explained elsewhere.

7.1 Unit of Deployment

BIRT reports are deployed in any of the following ways:

- As a BIRT XML design file submitted to the server for one-time execution. Additional resources are found by searching.
- As a BIRT XML file stored on the server. Additional resources are found by searching.
- As an Information Application Archive (IAR) deployed to the server. The archive file can contain all resources needed to run the report.
- As a complete application archive. The archive contains two or more related reports and the needed resources.

7.2 Information Application Archives

The Information application ARchive (IAR) is a zip file that can contain the following:

- One or more report designs
- Libraries (libraries can contain reusable components, messages, data sources, data sets, etc.)
- Images
- Java code
- Archive manifest file that describes the archive contents

The archive ensures that each report is deployed with the resources it requires; and isolates the design from subsequent changes to the resources.

The IAR file is ideal for casual users and junior developers. Everything needed to run a report is put into the IAR file. The file can be freely moved to different folders and different servers without worrying about the deployment environment (except, of course, for the data sources: the IAR file does not isolate the developer from changes to the database schema, etc.)

The key drawback of an IAR is that the report must be repackaged and redeployed if any of its resources changes. For example, if a logo change, then the user must find all reports that use that logo and rebuild, then redeploy the IAR files. If a resource changes frequently (such as promotional text or marketing images), then the user may want to consider storing the resource outside of the IAR file as explained below.

The Eclipse Report Designer (ERD) is responsible for building and deploying IAR files, and for tracking dependencies among IAR file contents.

7.3 Deploying Design Files and Partial Archives

The developer can deploy a design file by itself, or can deploy an archive that contains a subset of the required resources. In the example above, the developer may choose to deploy an archive with all resources except the promotional text and images. The promotional resources are stored externally to the design so that they can change without the need to redeploy the report.

BIRT uses a search algorithm (explained below) to locate any resources that do not appear in the IAR (or all resources, if the design file itself is deployed without an IAR.)

7.4 On-Demand Report Submission

BIRT is designed to allow customers to create custom report “wizards” that are part of a larger web application. For example, a wholesaler could allow its customers to define reports that show purchases over time, to limit scope to certain product areas, to predict purchase based on a certain growth pattern, etc.

The customer application presents the UI and generates an XML design file with the needed report. The application then submits the report to the server for one-time use. The application may provide a way to save a report design, if needed. Since many reports use the same styles, data sources and other resources, such “on demand” reports will usually build on one or more libraries.

The application can submit the report in one of two ways. First, the application can create an IAR file that contains all needed libraries, images, etc. Second, the application can simply submit the XML design file, and define a deployment environment that allows BIRT to find the needed resources when needed.

Later sections discuss the resource resolution process. These processes are based on a starting directory. However, on-demand reports are not deployed to a directory. Instead, the FPE should allow the user to establish a “virtual” deployment directory: a folder that acts as a base for resource searches for on-demand reports. Optionally, the report itself can set the “base” property that gives a folder to use for resource searches.

7.5 Resolving Resource References

Report resources include libraries, images, user-defined data files and so on. BIRT provides a resource search algorithm to locate resources.

7.5.1 Absolute References

An absolute reference identifies a reference on the iServer or application server. Absolute references are helpful when a report will be deployed to just one location, many reports use the same set of resources, and the server is used for just one application. In the example above, the developer may create a “promo” directory that contains the promotional images and text, and reference them with as “/promo/blurb.html” and “/promo/graphic.jpg.”

On the iServer, the absolute name follows iServer rules. It may start with a slash, meaning that the file is relative to the root folder of the same Encyclopedia volume that contains the report itself. The iServer also allows absolute links that reference a user’s home folder: “~bob/reports/mumble.jpg”. (See the IDAPI documentation for details.)

An embedded application uses file system references suitable for the operating system. These may include a slash (“\foo\bar\mumble.jpg”), a drive indicator on Windows (“c:\imgs\mumble.jpg”), a UNC name on Windows (“//mymachine/imgs/mumble.jpg”) or a home folder on UNIX (“~rptadmin/imgs/mumble.jpg”).

7.5.2 Relative References

If multiple applications share a server, or a single report must be deployed to multiple servers, then a relative resource reference may be more convenient. BIRT locates the resource using a *resource search*. This search is similar to how Unix locates executables, C++ locates include files or Java locates classes.

A relative reference starts with a name, or with the relative directory names: “.” or “..”. For example: “mumble.jpg”, “img/mumble.jpg”, or “../img/mumble.jpg.”

The search uses the following:

- The IAR file (if used).
- The Base property for the report (if set).
- The server folder to which the report is deployed (if the report is deployed, not used for on-demand reports.)
- A resource path.

BIRT looks for the file in these locations in the order explained below. In each case, BIRT combines the target folder with the relative name to give an absolute name. Any of the names on the path can be an IAR file. If so, then the search continues inside the IAR. For example, if the target folder is “/myApp/reports” and the relative name is “../img/mumble.jpg” then BIRT produces a candidate absolute name of “/myApp/img/mumble.jpg”. The search stops when BIRT finds a file that matches a candidate name.

The search sequence is as follows:

- The directory given by the “base” property of the report design (if any.)
- The directory that contains the design (unless the report is on-demand.)
- If the report is on-demand, then the assumed deployment location configured for the FPE.
- A search path defined by a BIRT-based application. (Such applications include the BIRT ERD, Factory, Presentation Engine and Web-Report Designer.) The search path is searched in order from first to last.

7.5.3 Configuration Variables

Resource references can contain references to configuration variables. As explained in a later section, configuration variables include both OS environment variables and BIRT-specific configuration values. Config variable references use the usual Unix syntax: “$IMG/mumble.jpg” or “$IMG/ver{$VER}-mumble.jpg”.

That is:

- The dollar sign introduces a config variable name.
- The variable name includes all the characters up to either 1) the end of the string, or 2) the first non alpha-numeric character, whichever comes first.
- The name can be enclosed in curley-brackets (“{“ and “}”) to resolve ambiguities, and to reference names that include non-alphanumeric characters: “{$VER}mumble.jpg” or “mumble.${file-type}”.

7.5.4 References from a Library

Libraries can also reference resources. If so, the above search is done on each reference from the library. However, the search starts with the location of the *library* itself, not with the location of the design that includes the library. (This makes sense: many reports in many locations can reference the library. The person deploying the library knows only of the library’s own location, not of locations of referencing reports.)

For example, a design and a library might both reference “mumble.jpg”. Suppose the library is deployed to “/libs/mylib.xml” and the design is deployed to “/reports/foo.xml.” Suppose both “/libs” and “/reports” contain a file called “mumble.jpg”. In this case, the actual file referenced by the design and library are different. While this may be surprising, it is a direct consequence of the library reference resolution rule. (And, indeed, it is required to ensure that a library is not broken due to differences between reports that use that library.)

7.5.5 IAR Files

The careful reader will note that the above algorithm implements all three deployment cases: IAR files, “bare” XML files and on-demand reports. If a resource appears in the IAR file, then the above search will find it there. If the IAR file does not contain the resource, or if the report is deployed outside an IAR file, then BIRT uses the environment search to locate the file.

A special step is required to package files that are referenced with an absolute name. The IAR manifest contains a mapping from absolute to IAR names with entries such as:

```
<file-map name=”/img/mumble.jpg” local=”mumble/jpg”/>
```

Thus, for absolute files, BIRT first checks the manifest of the IAR file. If the file is not found, then BIRT checks the absolute location.

7.6 Resolving Hyperlinks

BIRT also uses the resource search to resolve the target of “drill-though” hyperlinks: links to other BIRT reports. See the Action element for details.

7.7 Resolving Java Code References

BIRT reports can reference Java code. BIRT locates Java classes as follows:

- Look in the IAR file that contains the report.
- Look in the folder (or directory) that contains the report.
- Search the Java class path as defined by the application.

Typically, a Java class used by only one report will be packaged into the IAR file for that report. A Java class used by several reports will be stored in location identified by the Java class path.

8. Configuration Variables

Reports frequently have deployment-specific dependencies. For example, developers often use a test database during development, but target a production database once the report is deployed. Or, an OEM may use a different company name in report titles for each of their customers.

Experience suggests that changing the report design to reflect these differences is both tedious and error-prone. Customers demand a simpler solution. Therefore, BIRT provides *configuration variables.* A configuration variable is simply a name/value pair very similar to an environment variable on Unix. Indeed, configuration variables include environment variables, along with other BIRT-specific values.

8.1 Using Configuration Variables

The report design can reference configuration variables in a number of ways. References to files can include the names in the typical Unix fashion: “$IMG/mumble.jpg.” Expressions can call a function to get the value:

```
getConfigVar( “CompanyName” )
```

Note: *Syntax is preliminary.*

Configuration variables are assumed to be strings. If the application needs the value as a number or date, the application can convert the string using the proper format depending on whether the application uses a locale-specific or locale-independent encoding:

```
parseDate( getConfigVar( “FiscalYearEndDate” ) )
```

Note: *Syntax is preliminary.*

BIRT recommends that all such values be encoded using XML-defined, locale-independent formats.

8.2 Defining Configuration Variables within a Report

For convenience in deploying and testing, a report can provide default values for any configuration variables that are undefined in a given environment. The report does this with the “config-vars” element within the XML design file.

8.3 Environment Variables

All environment variables are implicitly defined as config variables. BIRT creates a config variable of the same name as the environment variable.

8.4 Deployment Variables

The deployment environment provides an environment-specific way to set configuration variables. For example, an embedded application could use a Java properties file or an XML file to hold configuration variables; while the iServer may use custom server configuration variables. The deployment environment may also allow multiple layers of configuration files: per report, per application and per server. The details are described elsewhere as part of the specifications for the deployment environment.

8.5 Search Path

BIRT defines a search path to locate the value of a configuration variable. The search path is the following:

- Environment variables
- Deployment file. If the deployment environment allows multiple files, then the more specific file (for a report or application) has precedence over more general files (for the entire server, for example.)
- The default value, if any, defined in the report.
- Otherwise, the variable is assigned a value of the empty string.

Note that it is not an error to reference an undefined configuration variable; the variable is simply assumed to have the value of an empty string.

9. Style Sheets

Note: *This section is a work in progress.*

Every report has its own style sheet. However, styles are more powerful when shared across a suite of related reports. This is done using by storing the style sheet separate from a report design in a file called a *library.* The library provides a set of reusable styles, among other things.

Reports designs can start with a library, but then can define additional styles within that report design. If a report includes a library with a style sheet, then the following rules apply:

- The report can define additional styles beyond those in the library. Such report-only styles can inherit from library -defined styles.
- The report can redefine styles in the library. In this case, the report-defined style "hides" the one in the library.
- The library can define its own values for any of the default styles. In this case, report items without a style, or styles that inherit from that default style, will inherit from the one defined in the library.
- A library can reference another template. In this case, the same rules above apply to the chain of library.
- It is not necessary to allow a report to inherit styles from two unrelated library. That is, a report can have just one library (though the library, itself, can include another library.) (This functionality is required to support the idea of using a report as a style sheet, and to keep the concepts simple. The style sheet report may have its own style

sheet; we don't want to complicate the style sheet feature by refusing to use certain reports.)

The name can be the same as a style in the library (if any). If so, then the style defined in the report replaces that defined in the library for the purposes of this one report.