BIRT Report Object Model – Report Design

Functional Specification

Draft 5: May 25, 2005

Abstract

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

Document Revisions

Version	Date	Description of Changes
Draft5	05/25/2005	Changed property name includeResources to includeResource.
Draft 4	04/26/2005	Properties changes for Module and Design: Module: CustomColor, helpGuide, description, colorPalette moved to "After the first Release". IncludeScripts moved to "First Release". Translations renamed to includeResources. Removed units, Resource and Translation structures. Design: Base moved to "After the first release"
Draft 3	04/07/2005	Modified Scalar Parameter Default Value property
Draft 2	02/25/2005	Changed the default value of property allowNull from True to False
Draft 1	11/29/2004	First BIRT release.

Contents

1. Introduction **3**

2. Modules **3**
2.1 Module Element 3
2.1.1 `includeResource` Property *3*
2.1.2 `pageSequences` Property *3*

3. Report Design **4**
3.1 Report Design 4
3.1.1 `components` Slot *4*

4. Libraries **5**
4.1 Using Libraries 6
4.2 Including Libraries in a Design 6
4.2.1 Search Path *6*
4.2.2 Packaged Deployment *7*
4.2.3 Non-Packaged Deployment *7*
4.2.4 On-the-Fly Deployment *7*
4.3 Library Element 7

5. Report Parameters **7**
5.1 Filter Parameter Element 8
5.2 List Parameter Element 8
5.3 Table Parameter Element 9

6. Deployment **9**
6.1 Unit of Deployment 9
6.2 Information Application Archives 9
6.3 Deploying Design Files and Partial Archives 10
6.4 On-Demand Report Submission 10
6.5 Resolving Resource References 11
6.5.1 Absolute References *11*
6.5.2 Relative References *11*
6.5.3 Configuration Variables *12*
6.5.4 References from a Library *12*
6.5.5 IAR Files *12*
6.6 Resolving Hyperlinks 13
6.7 Resolving Java Code References 13

7. Configuration Variables **13**
7.1 Using Configuration Variables 13
7.2 Defining Configuration Variables within a Report 14
7.3 Environment Variables 14
7.4 Deployment Variables 14
7.5 Search Path 14

8. Style Sheets **14**

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

Properties

`includeResource`

Properties file containing externalized messages specifically for this report.

`pageSequences`

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

2.1.1 `includeResource` Property

Externalized messages specifically for this report.

Summary

Display Name: Include Resource

Rom Type: String

JavaScript Type: String

Default value: None

Settable at runtime: No

Availability: First release

Description

Many companies have customers, employees or suppliers in many different countries. The includeResource property allows a report to externalize its text in external resource files.

See Also

Text structure

2.1.2 `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.

3. Report Design

3.1 Report Design

Contents

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

`components`

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

3.1.1 `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

4. 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.

4.1 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.

4.2 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.

4.2.1 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.

4.2.2 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.

4.2.3 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.

4.2.4 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.

4.3 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

5. 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.

5.1 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.

5.2 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.

5.3 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.

6. 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.

6.1 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.

6.2 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.

6.3 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.)

6.4 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.

6.5 Resolving Resource References

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

6.5.1 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”).

6.5.2 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.

6.5.3 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}".

6.5.4 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.)

6.5.5 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.

6.6 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.

6.7 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.

7. 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.

7.1 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.

7.2 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.

7.3 Environment Variables

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

7.4 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.

7.5 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.

8. 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.