BIRT Report Object Model – Textual Elements

Functional Specification

Draft 1: November 29, 2004

Abstract

The BIRT Text element is a powerful tool for including formatted text within a report. Text can be in HTML or RTF as well as plain text. Text can include embedded expressions.

Document Revisions

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

Contents

1. Introduction

Reports present information in many formats both graphical (charts) and textual. This specification describes the textual options.

Because text is so important within a report, ROM provides four distinct textual element, each designed to solve a particular reporting problem. These elements are:

- Label: displays a short bit of text and often appears as a table header, as a label of a data value, and so on. The text cannot contain internal formatting.
- Data: displays a single data value from a data set, from an expression, and so on. An optional format string determines how date and numeric values appear when displayed.
- Text: displays longer textual passages. It can contain internal formatting in HTML, and can contain embedded data values. A text element is often used to create report titles, disclaimers, form letters, explanations and similar content.
- Multi-line data: Displays one data value, but that value can contain internal formatting specified using either HTML or Rich Text Format (RTF).

The following table shows the categorization of the four textual report items:

	Text Size	
Source of Text	**Short**	**Multi-line**
Report Design	Label	Text
Data set	Data	Multi-Line Data

Labels and data item are often used in tables, grids and free-form layouts. Text and multi-line data items are often used to create entire paragraphs within a report.

1.1 Terminology

- Rich Text Format" ("RTF") – A format defined by Microsoft for text markup. RTF is recognized by Microsoft Word and many other applications.
- HTML – As used here, HTML refers to a subset of the HTML 4 specification.
- Cascading Style Sheets (CSS) – CSS works in conjunction with HTML to provide advanced style settings.

1.2 Multi-line Items: Text and Multi-Line Data

While many reports present simple tabular lists of data, others need to display data as part of a document. Typical examples are form letters, promotional messages, contracts, and so on. Often such text is formatted with specific paragraph breaks, fonts, highlights, etc.

BIRT provides two kinds of text elements for this purpose. Both present a block of text. The text can contain formatting. The two text items are:

- Text item: the user provides the text, which can be localized.

- Multi-line data: the text comes from a file, a data set, or an expression.

Both items display formatted text in multiple lines. The key difference between them is the source of the text. Text for the text item is created by the developer as part of the report design, while the text for the multi-line data item is read from a data set.

The text item is often used to create report content such as headers, explanations, and standard paragraphs. The multi-line data is used when the text itself is defined outside the report; perhaps in the form of a standard disclaimer provided by the Legal department, a monthly promotional message provided by Marketing, and a product description stored in the database.

What the two items have in common is the ability to format the text using HTML for the text item, and RTF or HTML for the multi-line data item. Text often has first-line formatting such as the typical first line, subsequent line, and right indent options available in CSS.

Think of the multi-line items as tools for displaying paragraphs of text: the goal is for the block of text to flow smoothly together, perhaps as a series of centered lines, perhaps as indented paragraphs and so on.

When using HTML formatting, hyperlinks can appear within the text using the HTML <a> tag. The hyperlinks work in output formats that support links.

1.3 Inserting Data into Text

Reports often find the need to insert data into a block of text. For example, a report heading may want to display report parameters within a block of centered text:

Monthly Sales Summary

For Month of January

Run by Charles Babbage

Or, a report may wish to create a form letter:

> Dear Mr. Boole,
>
> Thank you for your interest in our Model 200 Differential Engine. You'll be pleased to know that our sales representative in your area, Ada Lovelace, will contact you shortly.

The Text element provides comprehensive support for this feature. This feature is often called "Mail Merge" in word processing programs.

1.4 Short-text Items: Label and Data

Short-text items are ideal for text displayed in a grid, a table, or in a form-like layout with label & data pairs. The label and data items are meant to be part of a larger structure. Labels display static text. Data items display the value of a column or expression.

The data item displays a simple value that can be a number, date or string. The data item provides options to format that data using a format string. A multi-line data item always displays blocks of text, with optional embedded text formatting in HTML or RTF.

The label item displays a simple bit of text and cannot contain formatting, but the text item can.

Short-text items have no first-line formatting applied. For example, if the report style sheet establishes a quarter-inch indent for the first line in a paragraph, that formatting applies only to multi-line data items, not to the short text items (label and data).

1.5 Localizing Text

BIRT allows the developer to localize text within a Label or Text element. If a report is to appear in only one language, the developer simply enters the text directly into the element. However, if the report is to be translated, then the developer can instead supply a *resource key*. The key allows the text to be *externalized* into a *resource bundle*. The resource bundle provides distinct translations based on the viewer's *locale*. Externalization is the process of moving user-visible strings into a resource bundle. Localization is the process of providing translations of these strings.

Externalization applies only to those elements for which the user enters text: Label and Text items. It does not apply to the items that obtain their values from a data set: Data and Multi-line Data items.

See sections elsewhere in the ROM specs for additional detail on this process.

1.6 Style

A wide range of style options control the appearance of text in a report. BIRT uses the CSS style system to define fonts, font sizes, colors, underlining, and other effects. As you review the four textual items, you'll see that none of them list style properties. Instead, like all BIRT report items, the textual items include all the BIRT style properties automatically. For example, each textual item has a font family, font size, color, background color and other properties. See the *ROM Style Specification* for the list of available style properties.

There are three main ways to control the style properties:

- Using the pre-defined style for each element, the slot in which the element appears, for the entire report and so on.
- Explicitly assigning a shared style name to the item.
- Explicitly setting the value of the style property for the item.

Using an implicit style makes it easy to give the entire report a consistent look with very little work. Use an explicit style for special cases such as when you want to call the user's attention to a specific item. Set individual style properties on an item when that item has unique requirements.

1.6.1 First-Line and Paragraph Style

You can control the first line style of a multi-line textual item. Use this effect to indent lines, apply special formatting to the first word or sentence, and so on. For example:

> First word letter lets you adjust the appearance of the first word. First line formatting allows you to adjust the spacing on the first line separately from that of following lines.

BIRT uses the CSS system for specifying these effects. The `:first-line` style name suffix selects the first line, and the `:first-letter` suffix selects the fist letter. To achieve the above effect for a Text item, you can define the following styles:

- `text`, the built-in style for text elements, gives the formatting to apply to the paragraph as a whole.
- `text:first-line` gives the formatting to apply to the first line. In this case, an additional indent.
- `text:first-letter` gives the formatting to apply to the first letter, in this case, a larger font and bold.

You could also use a custom style. For example, if you only want selected text to have the above effect, create a styles called `my-text`, `my-text:first-line` and `my-text:fist-letter`. Then, set the style property of your text item to `my-text`. Even though you don't set the text item to use the suffix, BIRT will automatically apply them as needed.

First-line and first-letter formatting applies only to multi-line text items (Text and Multi-line Data), but not to the short text items (Label and Data.)

1.6.2 Link Style

BIRT also allows you to apply special formatting to hyperlinks. BIRT again follows the CSS system and uses the :link and :visited suffixes. In HTML, these effects apply only when the web browser supports them. In other formats, the :link format applies, but not the :visited format. For example, suppose that you have a project standard that links are to appear green, and visited links are to appear in navy. You would define the following styles:

- `report` is the overall default style for the report.
- `report:link` gives the formatting for unvisited links, in this case green. (This is equivalent to the `:link` style in CSS.)
- `report:visited` gives the formatting for visited links, in this case navy. (This is equivalent to the `:visited` style in CSS.)

Again, you can create a specialized style to apply these rules selectively. For example, you could add to the `my-text` style above to apply link formatting only to items that have that style. Define `my-text:link` and `my-text:visited`. Again, you don't set these styles on the element explicitly, BIRT applies them implicitly.

1.7 Summary of Textual Elements

The following table summarizes the features, capabilities and purposes of the four BIRT textual report items:

	Label	Text	Data	Multi-line Data
Source	Static Text		Value-Expression	
Localizable	Yes		No	
Multi-/Single-Line	Single	Multi	Single	Multi
Mail Merge	No	Yes	No	No
Purpose	Fixed String	Headers, explanations,	Database column	Text created outside the

	Label	Text	Data	Multi-line Data
	String	standard-paragraphs.		current document
Visual Format	No	HTML	No	HTML/RTF
Data Format	String	String	String/Number/Date	String
Visual Format for *value-of*	N/A	HTML	N/A	N/A
Data Format for *value-of*		String/Number/Data		
Content-Type	N/A	Plain/HTML	N/A	Plain/HTML/RTF
Embedded Hyperlink	No	Yes	No	Yes
Action	Yes	No	Yes	No
Help Text				

1.8 References

- Cascading Style Sheets, level 1, W3C Recommendation 17 Dec 1996, revised 11 Jan 1999, http://www.w3c.org/TR/CSS1. Provides description of the text formatting model
- Rich Text Format (RTF) Specification, RTF Version 1.7, 8/2001, from Microsoft. Available from the MSDN web site.

2. Short Text Items

As noted above, Label and Data both display short bits of text without internal formatting. Label displays static text, while Data displays information from the data set, from parameters, from an expression, and so on.

2.1 Label Item Element

Displays a static piece of text. The text can be externalized.

Summary

Base element: Report Item

Availability: First release

XML Element Name: `label`

Predefined Style Name: `label`

Properties

`text`

The text to display in the label. The text can be externalized.

`action`

Optional hyperlink defined for the label.

`help-text`

Optional “tool-tip” text.

Description

A label is just that: a static piece of text displayed in the report. In the simplest case, the label directly contains the text. Labels items are for use with grids, free-form containers, matrices, tables and other items that need a short, discrete block of text.

The text in the label does not normally contain multiple lines. However, if the text provided is too long to fit into the available width, BIRT will automatically break the text into multiple lines. However, one would normally want to use the Text element if the intent is to display multiple lines, especially if control over the indentation, line spacing and formatting is needed.

The Label element is a kind of Report Item and inherits the `dataSet` property. However, labels display static text, so no data set should be associated with a label.

See Also

Text Item element

Data Item element

2.1.1 `text` Property

The text to display in the Label.

Summary

Display Name: Text

ROM Type: Text structure

JavaScript Type: `PropertyStructure` object

Default value: None

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

The text to display in the label. The text can be externalized. The text cannot contain internal formatting.

See Also

Text structure

2.1.2 `action` Property

Optional hyperlink defined for the label.

Summary

Display Name: Action

ROM Type: Action structure

JavaScript Type: `PropertyStructure` object

Default value: None

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

Optional hyperlink defined for the label. Like the label itself, the hyperlink is static. Use the Data Item to create a hyperlink that varies depending on the value of data rows.

See Also

Action structure

2.1.3 `help-text` Property

Optional tool-tip text that appears when the user hovers the cursor over the item.

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

Reports often display large amounts of data on a single page. Labels are often short: acronyms, single words and so on. While this works well once the user is familiar with the labels, it can make the report confusing to the first-time user. Help-text helps bridge the gap. It allows the user to us a label such as “ROI” in the report. If the user needs more information, he can hover the cursor over the label and help text will appear that says something like, “Return on Investment: defined as the ratio of profit to funds invested.”

The help text is optional. If no text is present, then no tool-tip text will appear. The help text can be externalized.

See Also

Text structure

2.2 Data Item Element

Displays a data set or computed value with optional formatting.

Summary

Base element: Report Item

Availability: First release

XML Element Name: `data`

Predefined Style Name: `number`, `date` and `string`

Properties

`action`

Optional hyperlink defined for the label.

`help-text`

Optional “tool-tip” text.

`value`

The value to display in the item.

Description

A data element displays data from a data set, an expression, a parameter, etc. Data items display the value of a column, parameter or expression. Data items are for use with grids, containers, matrices, tables and other items that need a short, discrete data value.

Data items allow Excel-like multi-part formats: One of the common Excel formats used by some people is: #,##0_);(#,##0);"---". See the Style element for a description of the available styles.

BIRT applies one of the predefined styles depending on the data value of the item. If the expression returns a number, then BIRT applies the `number` style. If the expression returns a string, then BIRT applies the `string` style. And, if the expression returns a date, then BIRT applies the `date` style.

See Also

Label element

Multi-line Data element

Style element

2.2.1 `value` Property

The value to display in the item.

Summary

Display Name: Value Expression

ROM Type: Expression

Expression Type: String, Date or Number

Expression Context: Element

Default value: None

Inherited: Yes

Settable at runtime: Yes

Availability: First release

Description

The value to display in the item. The expression is most often simply a reference to a data set column, but it can also reference a report parameter, a formula, a special value or other data item.

The expression is usually evaluated when the report is rendered, not in the Factory, though it may be evaluated in the Factory also. Expressions should depend only on variables and objects available both in the Factory and Presentation Engine. See the *ROM Scripting Specification* for details.

BIRT automatically determines the type of formatting to apply based on the actual return value of each expression. If the return value is a Number, BIRT applies numeric formatting. If the return value is a Date, BIRT applies date/time formatting. If the return value is a String, BIRT applies string formatting. The data item itself does not have a type; the type is computed independently for each and every data item based on the return value of this expression.

2.2.2 `action` Property

Optional hyperlink defined for the data item.

Summary

Display Name: Action

ROM Type: Action structure

JavaScript Type: `PropertyStructure` object

Expression Context: Element

Default value: None

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

Optional hyperlink defined for the data item. Like the value expression, the hyperlink is computed, and can link to a different resource for each data item instance.

See Also

Action structure

2.2.3 `help-text` Property

Optional tool-tip text that appears when the user hovers the cursor over the item.

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

Reports often display large amounts of data on a single page. Labels are often short: acronyms, single words and so on. While this works well once the user is familiar with the labels, it can make the report confusing to the first-time user. Help-text helps bridge the gap. It allows the user to display a value, such as the Return on Investment (ROI) for an asset in the report. If the user needs more information, he can hover the cursor over the label and help text will appear that says something like, “Return on Investment: defined as the ratio of profit to funds invested.”

The help text is optional. If no text is present, then no tool-tip text will appear. The help text can be externalized. The help text is static, it cannot vary for each data item.

See Also

Text structure

3. Multi-line Textual Items

The text item allows the developer to provide the text as part of the report design. The text can be localized. Text can be in HTML or plain text format. HTML text can contain placeholders for data and expressions. The Multi-line Data item, on the other hand, displays text obtained from a data set, from a file, from an expression and so on. The Multi-line Data item allow optional formatting in either HTML or RTF.

3.1 Using a Text Item as a Container

A text section can be used to create report headings and other textual content. Using a text item is often easier than using a grid, free-form or other container. With text, the item contains text instead of rectangular report items. Text is placed onto lines. Lines grow and shrink depending on the size of their contents just as in Word or HTML. Text can be centered, or can be left or right justified. ROM provides control over line spacing, paragraph spacing and similar properties.

For example:

```
<c><b><span style=”font-size: larger”>
Monthly Sales Summary</span><br>
For the month of:
<value-of>Date.monthName( params.MonthParam )</value-of>,
<value-of>Date.year( params.MonthParam )</value-of>
<\b></c>
```

Displays a report title that looks like this:

Monthly Sales Summary

For the month of: November, 2004

3.2 Text Item

Displays a multi-line block of text defined within the report. Can contain embedded formatting and value expressions.

Summary

Base element: Report Item

Availability: First release

XML Element Name: `text`

Predefined Style Name: `text`

Properties

`text`

Text for the item.

`contentType`

An optional property that defines the type of text.

Description

The text item allows the developer to provide the text as part of the report design. The text can be localized. Text can be in HTML or plain text format. HTML text can contain placeholders for data and expressions: something line the mail merge feature of a word processing program. “Mail merge” is feature in which a standard block of text contains “place holders” for data retrieved from a database. It is often used to create form letters, or to format specific blocks of text within a report. For example, a report can use the text item to display a single line with the US-style city/state/zip address line:

```
<value-of>row.city</value-of>, <value-of>row.state</value-of>
<value-of>row.zipcode</value-of>
```

Which might appear as:

```
South San Francisco, CA 94080
```

Another example is “mail-merge”, incorporate personalized data into a block of text. Mail-merge is especially powerful when combined with sections to create conditional paragraphs that further customize each letter.

The Text item provides many powerful features:

- Ability to enter formatted text that contains markers for data fields, expressions, and so on. When run, the value of each field is concatenated with any static text to produce the final output.
- Control text formatting on a character-by-character basis within a single text control. For example, displaying one word in a paragraph in a different font and color from the rest.
- Automatically adjust element size based on content.
- A single text element can span multiple pages to accommodate large amounts of text.
- Specify formatting using embedded HTML.

The text object is designed to allow continuous flow of text. Creating the same affect with a set of report items has the well-known problems that the appearance varies depending on characteristics of the printer or display format. (Font widths & heights vary across presentation formats, making it difficult to align free-form labels and data items.) The user can combine a series of text elements, each in a separate section, to create conditional blocks of text.

BIRT leverages HTML and other open standards for rendering. For the most part, formatting within a block of text is simply passed to the browser or other formatting engine for processing.

See Also

Label Item element

Multi-line Data element

Style element, especially the Font and Text properties.

HTML Support section below.

Embedded Expressions section below.

Plain Text section below.

3.2.1 `text` Property

The text for the item.

Summary

Display Name: Text

ROM Type: Text structure

JavaScript Type: `PropertyStucture` object

Default value: None

Inherited: Yes

Settable at runtime: No

Availability: First release

Description

The text for the item. See the `contentType` property for how to identify the text formatting. The text can be externalized.

3.2.2 `contentType` Property

The formatting within the text: HTML or plain text.

Summary

Display Name: Content Type

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	BIRT will infer the format as explained below.
Plain Text	plain	Plain text with no formatting.
HTML	html	Formatting using a subset of HTML tags.

Description

The user can explicitly identify the format of the text using the Content Type property. Or, the user can set the property to “auto”, (or omit the property) and BIRT will infer the format from the text itself.

BIRT determines the text format by examining the first few characters of the string. If the first characters are “<HTML>” (in either upper case or lower case), then the string is assumed to be HTML. Otherwise, the text is plain text. Any white space before these characters is ignored. That is, “<html>” and “ <html>” are both taken to indicate that the text is formatted in HTML.

3.3 Multi-Line Data Item

Summary

Base element: Report Item

Availability: First release

XML Element Name: `multi-line-data`

Predefined Style Name: `text`

Properties

`value`

An expression that provides the text.

`contentType`

An optional expression that defines the type of text.

Description

The multi-line data item displays blocks of text retrieved from the database, from a file, or from an expression. The text can be plain text, HTML or RTF. The format of the text can be fixed at design time, or can be dynamically selected at run time to match the format of the incoming text.

The user can search text within the multi-line data item.

See Also

Text Item element

Data Item element

Style element, especially the Font and Text properties.

ROM Scripting Specification for information on expressions.

HTML Support section below.

Embedded Expressions section below.

Plain Text Support section below.

RTF Support section below.

3.3.1 `value` Property

Expression that returns the text to display.

Summary

Display Name: Value Expression

ROM Type: Expression

Expression Type: String

JavaScript Type: String

Default value: None

Inherited: Yes

Settable at runtime: Yes

Availability: First release

Description

The value expression provides the text to display. The expression is most often simply a reference to a query column. But, it can also reference a report parameter, a formula, a special value, a file, or other data item.

See Also

`contentType` property

3.3.2 `contentType` Property

3.3.3 `contentType` Property

The formatting within the text: HTML or plain text.

Summary

Display Name: Content Type

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	BIRT will infer the format as explained below.
Plain Text	plain	Plain text with no formatting.
HTML	html	Formatting using a subset of HTML tags.
RTF	rtf	Microsoft Rich Text Format encoding.

Description

The user can explicitly identify the format of the text using the Content Type property. Or, the user can set the property to “auto”, (or omit the property) and BIRT will infer the format from the text itself.

BIRT determines the text format by examining the first few characters of the string. If the first characters are “<HTML>” (in either upper case or lower case), then the string is assumed to be HTML. If the first characters are “\rtf”, then BIRT assumes that the text is formatted in RTF. Otherwise, the text is plain text. Any white space before these characters is ignored. That is, “<html>” and “ <html>” are both taken to indicate that the text is formatted in HTML.

4. HTML Support

The Text and Multi-line Data items provide comprehensive HTML support. HTML is a rich specification. BIRT supports a subset of features. In the web environment, BIRT passes the formatting to the web browser for processing, so support is browser-specific. In the commercial product, BIRT itself will process the formatting.

HTML text is identified with the “<HTML>” starting element. This, and other elements, are case insensitive. The HTML text should not contain the <HEAD> HTML section. The <BODY> tag is optional.

Example:

```
{ <HTML>
   { <BODY> } }
         This <i>is</i> an example.
   { { </BODY> }
{ </HTML> }
```

In the above, items in curly-brackets are optional.

If the user sets the Content Type property to HTML, then the user can omit the <HTML> tags:

```
This <i>is</i> an example.
```

4.1 Supported Elements

The following table identifies those HTML 3.2 elements that fit into the context of the Text element. The actual set of supported elements depends on the UI and Presentation Engine in use.

Tag	Description	Priority
<!-- … -->	Comment. Contents of the comment are ignored.	Med.
<A>	Anchor	High
<B>	Bold text.	High
<BODY>	Body element. Ignored.	Low
 	Line break.	High
<CENTER>	Centers text. Deprecated in HTML 4. But, kept for convenience and compatibility with older HTML.	High
<CODE>	Computer code fragment	Low
<DD>	Definition description	Low
<DEL>	Deleted text	Low
<DIV>	Generic language/style container	Med
<DL>	Definition list	Low
<DT>	Definition term	Low
<FONT>	Sets font attributes. Deprecated in HTML 4. But, kept for convenience and compatibility with older HTML.	Med
<EM>	Emphasis.	Med
<HEAD>	Ignored except for the STYLE section.	Low
<H*n*>	Heading level	Low
<HTML>	Marks the text as HTML.	High
<I>	Specifies italicized text. Deprecated in HTML 4. But, kept for convenience and compatibility with older HTML.	High
<IMAGE>	BIRT-specific variant of the image tag that allows referring to images in a data source or embedded in the report design.	High
<IMG>	Standard HTML image reference (via a URI)	High
<INS>	Inserted text	Low
<LI>	List item.	Med
<OL>	Ordered list.	Med
<PRE>	Block of fixed-width text. The text truncates if it is wider than the dynamic text control. The text does not wrap.	High
<P>	Paragraph.	High
<SPAN>	Generic language/style container	High
<STRONG>	Strong emphasis	Med
<SUB>	Subscript	Med

Tag	Description	Priority
<SUP>	Superscript	Med
<TITLE>	Ignored.	Low
<UL>	Unordered list.	Med
<TT>	Teletype or mono-spaced text style	Low
<U>	Underline. Deprecated in HTML 4. But, kept for convenience and compatibility with older HTML.	High
&*sym*	Replaced with the symbol defined in HTML 4 specification, section 24 Character entity references in HTML 4	High
&#*D*	where D is a decimal number, refers to the ISO 10646 (Unicode) decimal character number D. As defined in the HTML 4 specification, section 5.3.1: “Numeric character references”.	High
&#x*H* &#X*H*	where H is a hexadecimal number, refers to the ISO 10646 hexadecimal character number H. Hexadecimal numbers in numeric character references are case-insensitive. As defined in the HTML 4 specification, section 5.3.1: “Numeric character references”.	High

4.1.1 HTML 4.0 Compatibility

Early versions of HTML included elements such as <FONT>, <B> (bold), <I> (italic), and <U> (underline) to control formatting. HTML 3.2 and 4.0 deprecated these elements in favor of the more general CSS style support. The following are the equivalent HTML and CSS style settings:

Old-Style HTML	HTML and CSS
<B>	<SPAN style=”font-weight: bold”>
<CENTER>	<DIV style=”text-align: center”>
<FONT	<SPAN
color=”*color*”	style=”color: *color*”
face=”*name*”	style=”font-family: *name*”
size=”*size*”	style=”font-size: *size*”
<I>	<SPAN style=”font-style: italic”>
<U>	<SPAN style=”text-decoration: underline”>

The CSS items are, in general, more powerful and flexible than the old HTML elements. Further, the CSS items can be applied to any element, not just the <SPAN> element

shown. However, for quick formatting, the HTML style is concise, easy-to-remember and convenient. This is why BIRT supports the old-style HTML elements.

BIRT formatters may translate the HTML syntax to the HTML/CSS syntax for user agents that do not support the old syntax. Further, the user can use custom style sheets to customize the meaning of the <B>, <I> and <U> elements.

4.2 Formatting

When shown in the browser, the Text and Multi-line Data item are part of the BIRT-created HTML document containment hierarchy. The HTML/CSS style information in effect at the start of the text element is that defined by BIRT's style inheritance rules. Specifically, in order of priority of a style property is:

- Set on the text item itself.
- Style properties inherited from a base element.
- Style properties inherited from a named style.

See the *ROM Styles Specification* for details. The report can then override these styles using CSS information within the HTML as described below.

4.3 Style Information

The features described below are "nice-to-have" for the text item, but will not be provided in the first release of BIRT. Indeed, the complexity of merging customer-defined and BIRT-defined styles may prevent BIRT from ever supporting the feature in this subsection. We leave it here for completeness.

4.3.1 External Style Information

HTML text can define CSS styles. BIRT copies the style information into the each HTML file that BIRT creates. The following rules apply:

- The style element can contain a reference to an external CSS style sheet. This is the preferred solution. If multiple HTML elements refer to the same style sheet, BIRT will ensure that the generated HTML file lists the external file reference only once.
- BIRT concatenates the style definitions from each text element onto the set of styles defined for an HTML page.
- If two elements define conflicting styles, the resulting behavior is undefined. The style used by the browser may be browser-specific, and may be determined by the undefined order that the duplicate styles appear in the generated HTML.
- User-defined styles appear before BIRT-generated styles. If a user style happens to duplicate a BIRT-generated style, the BIRT style will usually win, though actual behavior is undefined.
- BIRT computes the set of styles and file includes from the Static Text attribute of the text element. If a localized value defines different styles, then these styles *will not* appear in the generated HTML.

A typical use case is to create a CSS page to customize, say, a form letter. The text element includes the form letter text, and references the external CSS style sheet.

For example:

```
<HTML>
   <HEAD>
      <STYLE TYPE="text/css">
         @import url(http://style.com/basic);
         p.formLetter { color: blue }
      </STYLE>
   </HEAD>
   <BODY>
   <p class=”formLetter”>Dear Sir, …
   </BODY>
</HTML>
```

The above imports a style sheet, and creates a paragraph style class that is then used inside the HTML content.

4.3.2 In-line Style

Another, simpler, approach is to specify the style information in-line in the HTML:

```
<HTML>
   <p style=”color: blue”>Dear Sir, …
</HTML>
```

The drawback, of course, is that the style information must be repeated. Still, it is a good solution for short bits of HTML.

4.3.3 Overall Style

To apply a style to the entire HTML section, create a <DIV> that holds the style:

```
<HTML>
   <DIV style=”color: blue; font-size: 11”>
         Sample text that will appear in 11-point blue text
   </DIV>
</HTML>
```

The above could also be done by defining a style (`div.myStyle`) and referencing it from the `<DIV>` element (`<DIV style=”myStyle”>`).

4.4 Character Encoding

BIRT designs are in Unicode. All static text is also in Unicode. For this reason, BIRT ignores any HTML character encoding information (“charset=*xxxx*”) that appears in the static HTML text.

The character encoding of database data is defined by the database interface. BIRT converts the data to Unicode when reading the data. Therefore, the character encoding information is ignored for expressions as well.

The user is responsible for creating HTML in Unicode. If the user has existing HTML that is to be used with a report, the user must convert that text to Unicode.

4.5 Syntax Rules

The following are syntax rules for HTML in the Text element:

- Even though the HTML is embedded in an XML file, the HTML text itself follows the HTML syntax rules, not the XML rules.
- The HTML must satisfy syntax rules as defined in the HTML 4.0 specification.

- End tags are optional, when marked as such in the HTML 4.0 specification.
- If an end tag is omitted, an implicit end tag is assumed at the end of the HTML text block. For example, the following text:

  ```
  <b>this is bold text
  ```

  Is treated internally as if the HTML were:

  ```
  <b>this is bold text</b>
  ```

- Elements and attribute names are case-insensitive.
,
 and
 are equivalent.

4.5.1 Unsupported Tags and Parameters

HTML tags that are not supported will be treated as white-space. HTML parameters that are not supported will be ignored. For example, the following HTML-tagged text:

```
<table width="100%" border="0" cellspacing="4">
  <tr>
    <td align="center">Call for</td>
  </tr>
  <tr>
    <td align="center">special pricing.</td>
  </tr>
</table>
<p style="color: red;">Offer ends next month!</p>
```

will be rendered like this:

Call for special pricing. Offer ends next month!

because the “table”, “tr” and “td” tags, and the “style” parameter will not be supported.

Unsupported tags will not cause a text chunk break.

4.5.2 Malformed HTML

With the exceptions of the relaxations from the standard described above, malformed HTML (for example, incorrectly nested tags) may not be rendered as the user might expect. However, the dynamic text control will not actually generate an error when it encounters malformed HTML. In other words, the behavior will be the same as that of a Web browser – the dynamic text control will make the best attempt it can to interpret whatever it encounters.

4.6 Embedded and Computed Images

A Text element (but not multi-line data element) with HTML formatting can also refer to an image embedded within the design file, or obtained from a data set. The HTML <IMG> tag already provides the ability to reference an image by URL. The ROM-defined <IMAGE> tag provides access to the other two types of images that the Image item supports.

The <IMAGE> element supports all the attributes of the HTML <IMG> tag except for the “src” element.

To reference an image embedded within the report design, use the “name” attribute:

```
<IMAGE name=”logo”/>
```

To reference an image obtained from an expression, use the “type” attribute with the value of “expr”:

```
<IMAGE type=”expr”>customer.photo</IMAGE>
```

The other valid value for “type” is “embedded”, which is the default.

4.6.1 Runtime Behavior

The behavior of the <IMAGE> tag is as follows:

- BIRT replaces the ROM-specific syntax with the equivalent HTML syntax:
- The IMAGE tag is replaced with IMG.
- The name=”*image*” element is replaced by src=”*link*” where *link* is a URL reference to obtain the embedded image.
- The type=”expr”>*expr*</IMAGE> content is replaced by src=”*link*” where *link* is a URL reference to obtain the computed image.

The URLs created are identical to those created by the Image element.

5. Embedded Expressions

The text item (but not the multi-line data item) formatted with HTML allows the developer to insert expression values within the text. Expression values are given by a “value-of” element. The expression is any valid BIRT expression. Expressions can appear with HTML and plain text.

For example:

```
Constant: <value-of>10</value-of> <br>
Parameter: <value-of>params.MyParam</value-of> <br>
Column: <value-of>row.CustomerName</value-of>
```

The expression can also contain any valid BIRT script or expression as described elsewhere.

5.1 Value

The Value element identifies a value to insert into the text for a Text element. The Value element contains the expression to evaluate to obtain the data. Value elements are cannot be inserted into text returned from the Value Expression property.

For example:

```
Dear <value-of>
if [Sex] = “M” then “Mr.”
else if [Sex] = “F” then “Ms.”
</value-of> <value-of>row.firstName</value-of>
<value-of>row.lastName</value-of >,
<p>
<b>Thank you</b> for your purchase of an exciting new
<value-of> row.productName</value-of > on
<value-of> row.purchaseDate</value-of>.
```

The above shows that the value can be either a simple value, or a script.

5.2 Expressions that Return HTML

The user can create highly formatted text when the content type is HTML by writing an expression that generates HTML:

```
<value-of format=”html”>
   String value;
   if ( row.customer_balance < 0 )
         value = “<span style=\”red\””;
   value += format( row.customer_balance, “(##,###.##”) );
   if ( row.customer_balance < 0 )
         value += “</span>”;
   value;
</value-of>
```

In this case, the “value-of” element is somewhat like the <script> element in HTML. The difference is that the last value mentioned in the script is the “return” value of the script.

Note that the special “format=’html’” attribute is required so that BIRT treats the above text as HTML instead of as a literal string.

HTML created in this way must follow certain rules:

- Cannot include additional “<value-of>” elements.
- Elements started in the value should complete in the value.
- The HTML must include only the subset of HTML supported by the Text element.

5.3 Formatting

The “format” attribute of the “value-of” element provides a BIRT format string appropriate to the type of the value. For example:

```
Your order shipped on
<value-of format=”yyyy-mm-dd”>row.ShipDate</value-of>. The total
amount was <value-of format=”$###,###.##”>row.OrderAmount</value-of>.
```

The allowed formatting strings are those described for the Format property in the ROM Styles specification.

If no format attribute is supplied, BIRT will use the formatting that applies to the Text element as a whole using the standard style property search as described in the Styles spec. That is, the rules for the style for a value-of element are the following:

- The format string, if any, that is provided for that value-of element.
- Otherwise, determine the type of the expression (date, number or string). Use the style search algorithm to find the corresponding format string as specified in the style.

This algorithm ensures that value-of expressions use the formatting in effect for the Text element, but can be overridden on a value-by-value basis if needed.

5.3.1 HTML Formatting

In addition, the special format “html” identifies the data item as raw HTML, processed according to the description above. Note that the “html” format is available only in Text items with a content type of HTML. (The content type can be set using the Content Type property, or detected automatically from the text itself.)

5.3.2 Automatic Formatting

The special format “auto” identifies that BIRT should infer the format based the rules explained below. The “auto” value is a convenience for the GUI: it allows the format attribute to appear in the <value-of> tag, even with it does nothing.

The following:

```
<value-of format=”auto”>row.myColumn</value-of>
```

Is equivalent to:

```
<value-of>row.myColumn</value-of>
```

5.3.3 Default Formats

If the user does not define a format, then BIRT applies a default format according to the following rules:

Type or Value	Default Formatting
Java null or database NULL	Value is an empty string (blank)
String	The string itself. BIRT escapes all special characters in the string.
Number	Standard Java formatting with locale-specific decimal separator.
Date	Locale-specific “General Date” format.

5.4 Values with Externalized Text

Users can externalize their text. The actual text is known only at run time. However, it is necessary for the Factory to obtain any columns required by the expression. To ensure that all columns are available, the developer should provide a default text that includes references to all needed columns.

6. Plain Text Support

Plain text will contain no formatting characters. Plain text cannot contain <value-of> tags for inserted values. (To create plain text with value-of tags, simply set the content type to HTML.)

Plain text can contain control codes. The Text element supports the following control codes:

Control Code	Meaning	HTML Equivalent
Space	Non-breaking space. That is, spaces are preserved in the output.	
CR, LF or CR/LF	Paragraph break	<p>
Tab	Single space	Space
Other	Ignored	

Text itself is encoded in UTF-8 within the design file and in Unicode within the browser. Plain text does now allow character escapes or other formatting.

If the user wishes to show literal HTML or RTF (perhaps for debugging), the user can set the Content Type property to “Plain”. This will force BIRT to treat the text as plain text, even if it starts with an HTML or RTF marker.

For example:

```
<HTML>This looks like HTML, but will show as plain text if Content
Type is set to Plain.</HTML>
```

And:

```
\RTF This looks like RTF, but will show as plain text if Content Type
is set to Plain.
```

7. RTF Support

The Text element will support a subset of the RTF 1.7 standard. For detailed information about RTF syntax, please refer to that standard available on the MSDN subscription site.

The typical use case for RTF support is to retrieve the RTF from a file or data source. The BIRT will probably not provide an RTF editor (at least in early releases.) Hence, the Text element is designed to process an entire RTF file, including header.

RTF must start with “\rtf” as the first few characters.

7.1 Terminology

This section uses the term “tag” to mean what the RTF specification calls a “control word”.

7.2 Character Encoding

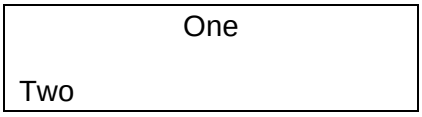
Character encoding information (“\ansi” and “\ansicpg*N*”) is ignored. This is because all characters in the tagged text is converted to UCS-2 automatically as they are read from the data source. The Text element follows the RTF Unicode standards introduced in RTF 1.7.

7.3 Groups

The use of braces (“{ … }”) to denote groups is supported. In particular, the use of groups to indicate default paragraph formatting is supported. For example:

```
{\rtf1\qc One\par\pard Two}
```

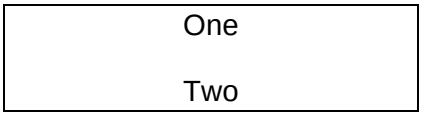
will render as:



because the paragraph break resets the text alignment, but:

```
{\rtf1\qc{One\par\pard Two}}
```

will render as:

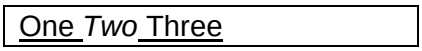


because the two paragraphs are grouped within braces, and the text alignment is applied to the whole group.

Braces are often used to provide localized character formatting. For example:

```
{\rtf1\ul One {\plain\i Two} Three}
```

will render as:



7.4 Supported Structural and Definition Tags

The following RTF structural and definition tags are supported:

Tag	Behavior
\rtf	Marks the text as RTF.
*	Extended feature group indicator – the contents of groups identified in this way are ignored.
\blue*N*	Blue value within color table.
\colortbl	Color table.
\deftab*N*	Default tab spacing. *N* is the spacing in twips.
\fonttbl	Font table.
\green*N*	Green value within color table.
\red*N*	Red value within color table.
\rtf*N*	Indicates RTF dynamic text. *N* indicates the major RTF version number. The value of *N* must be “1”.
\uc*N*	Specifies the number of plaintext characters following a “\u*N*” tag.

The unit “twip” refers to a 1/20 of a point, or 1/1440 of an inch.

7.5 Supported Paragraph Formatting Tags

The following RTF paragraph formatting tags are supported:

Tag	Behavior
\fi*N*	First line left indent. *N* indicates the indent in twips. Note that "\fi" is additive with "\li", and *N* may be negative. For example, "\li1440\fi-720" makes the first line indented 0.5" and remaining lines indented 1.0".
\li*N*	Left indent. *N* indicates the indent in twips.
\pard	Reset all paragraph formatting attributes.
\qc	Centre text.
\ql	Left-align text.
\qr	Right-align text.
\ri*N*	Right indent. *N* indicates the indent in twips.
\sa*N*	Vertical space after the paragraph. *N* indicates the space in twips.
\sb*N*	Vertical space before the paragraph. *N* indicates the space in twips.
\sl*N*	Vertical line spacing. In simple terms, *N* indicates the total line height in twips (but please refer to the RTF specification for a full definition of the behavior of this tag). *Note that the value of N appears to be scaled such that 240 is "single" spacing; the RTF specification does not explain this.*
\slmult*N*	Vertical line spacing multiplier. *N* indicates the multiplication type (but please refer to the RTF specification for a full definition of the behavior of this tag).
\tx*N*	Set tab stop. *N* indicates the tab stop position in twips.

7.6 Supported Character Formatting Tags

The following RTF character formatting tags are supported:

Tag	Behavior
\b	Start bold text.
\b0	End bold text.
\cf*N*	Color number. *N* indicates the color number within the color table.
\f*N*	Font number. *N* indicates the font number within the font table.
\fs*N*	Font size. *N* indicates the font size in half points. Since AFC does not recognize fractional font sizes, half point sizes will be truncated to integers. For example \fs21 indicates font size 10.5pt; this will be truncated to 10pt.
\i	Start italic text.

Tag	Behavior
\i0	End italic text.
\plain	End all character attributes (bold, font, italic, etc.)
\ul	Start underline.
\ul0	End underline.

7.7 Supported Special Characters

The Text element recognizes any Unicode character encoded in the standard RTF form "\u*n*" (*"n"* is the decimal Unicode value of the character).

The Text element recognizes any character encoded in the standard RTF form "\'*xx*", where *"xx"* is the 2-digit hex value of the character. The value] is treated as a Unicode value, regardless of the original document encoding.

In either case, it is possible that the specified character will not exist in the requested font; when this occurs, the "null" symbol (" ", hex value 0x7f) is used. Note that the representation of undefined characters is CSS UA (browser) specific in the web environment, but is controlled by BIRT when printing.

The following RTF special character tags are supported:

Character	Unicode	Tag	Meaning
		\par	Paragraph break.
		\<*LF*>	Paragraph break (<*LF*> is ASCII 10).
		\<*CR*>	Paragraph break (<*CR*> is ASCII 13).
		\line	Line break.
		\tab	Move to next tab stop.
		<*TAB*>	Move to next tab stop (<*TAB*> is ASCII 8).
		\emspace	Non-breaking space equal in width to an "m".
		\enspace	Non-breaking space equal in width to an "n".
\	\u92	\\	Backslash.
{	\u123	\{	Left curly brace.
}	\u125	\}	Right curly brace.
‘	\u145	\lquote	Left single quote.
’	\u146	\rquote	Right single quote.
“	\u147	\ldblquote	Left double quote.

Character	Unicode	Tag	Meaning
”	\u148	\rdblquote	Right double quote.
•	\u149	\bullet	Bullet symbol.
–	\u150	\endash	En-dash.
—	\u151	\emdash	Em-dash.
	\u160	\~	Non-breaking space.
		_	Non-breaking hyphen.

7.8 Bulleted and Numbered Lists

Correctly formatted RTF contains simple textual representations of bulleted and numbered lists. The Text element uses these textual representations, and hence does not need to understand the complex RTF tags that describe list formatting rules for editing purposes.

7.9 Matching and Nesting Tags

RTF does not require that tags match. For example, the following RTF is valid, even though there is no start bold tag, nor any finish italic tag:

```
{\rtf1\b0\i italic}
```

It follows from the above that RTF does not require nesting of tags. For example, the following RTF (with overlapped character formats) is valid:

```
{\rtf1\b bold \i bold & italic\b0  italic\i0}
```

7.10 Unsupported Tags

RTF tags that are not supported are silently ignored. The exact behavior will be as described in the RTF standard.

7.11 Relaxations from Standard RTF

The following relaxations from standard RTF are desirable; however it may not be possible to implement any or all of them:

- The outermost braces may be omitted.
- The “\rtf” tag may be omitted.
- The numeric value of the “\rtf” tag may be omitted.
- There will be no requirement for any character set tags to be present. If any character set tags are present, they will be ignored.
- The “\deff” tag will not be required. If it is absent, the default font will be that defined by the dynamic text control’s *Font* property.

7.12 Malformed RTF

If the Text element encounters malformed RTF that it is unable to handle, it will fail with an runtime error.

7.13 Compatibility with Other RTF-Aware Applications

While RTF text rendered in the Text element should normally appear similar to the same RTF text rendered by another RTF-aware applications, this will not be guaranteed. Discrepancies between how any particular piece of RTF text is rendered in the Text element and in, for example, WordPad will not automatically be defined to be defects. In particular, rendering of formatting on the Web will be influenced by CSS User Agent (browser) settings and limitations.

7.14 Embedded Expressions

The RTF text can contain embedded expressions as discussed below. Such expressions cannot include RTF formatting. In the rare case in which the user wants to include "<value-of>" as literal text, the user can do so in one of two ways:

- Apply formatting to any of the characters in the string. (BIRT will not recognize the text as a tag if it contains formatting.)
- Use the special form "<&value-of" for the text which BIRT will replace with the literal value "<value-of".

8. Text Element Processing

Text element processing is done at presentation time, based on the locale of the user viewing the report. This allows the user to retrieve a localized version of the HTML, and to insert locale-specific text and formatting.

BIRT processes a text element using rules defined in CSS. Briefly, the process includes:

- Parse the HTML or RTF into an abstract parse tree.
- Evaluate each embedded expression in the order they appear in the text. Replace the expression with the formatted result of the expression.
- If the format is HTML, and the format type is HTML, parse the resulting value to produce a revised abstract parse tree.
- Divide the text into blocks (paragraphs). Blocks are separated by line breaks. Each text element contains at least one block. The HTML <P> element introduces a new block.
- Divide each block into text spans. A span is a block of characters that have the same formatting. A span is introduced by HTML tags such as <B>, <I>, <EM>, <SPAN>, etc. or by the RTF equivalents.
- Flow words onto lines.
- Compute line height. Line height is given by the tallest element within the line such as a large font, and image, etc.
- Compute block height. A block is simply the sum of the lines within the block

In many cases, BIRT does not perform the above formatting itself. When rendering to HTML, the HTML browser (CSS User Agent) does the actual formatting. When rendering to FO, the FO processor does the actual formatting. However, the BIRT report editor (and in particular, the “live” editor) must perform the formatting as part of its overall implementation of a CSS UA.

The reader should consult the CSS specification at http://www.w3c.org/TR/CSS1 for a detailed explanation of the formatting workflow.

8.1 Design File Encoding

The design file treats the contents of the text element as literal text. That is, whatever text is given to the BIRT for the value of the text property of Text element is guaranteed to be preserved when the user saves and reopens the design, even if the HTML is badly formed, or does not adhere to the XML syntax rules.