BIRT Report Object Model – Styles

Functional Specification

Draft 1: November 29, 2004

Abstract

BIRT is based on a comprehensive report object model that describes all aspects of the report. The XML design format directly represents the object model. This spec describes the style features of the report object model from the developer's perspective.

Document Revisions

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

Contents

1. Introduction

This document is one of several that describe the BIRT Report Object Model (ROM). This document discusses the styles feature. Please see the overall ROM document for background material required when reading this specification.

1.1 Motivation

Very few customers build just one report – most applications consist of dozens (or even hundreds) of similar reports. Most production reports are based on a set of common styles and formatting rules. This may be in the form of an official style guide created by the Marketing or similar group, or it may simply be a look that has evolved informally over time. One of the more tedious tasks for a report developer is to ensure that items in a report all follow the same formatting rules.

Regardless of how the look evolved, users of the reports will expect that each new report will share a common look and format with existing reports. This is especially true for reports used outside the group that created them; the reports must have a professional, uniform appearance.

BIRT solves these problems by providing a powerful style system based on Cascading Style Sheet feature of HTML.[1]

The BIRT style specification is based on CSS 1[2]. The reader of this section should be familiar with the CSS 1 specification available at http://www.w3c.org/TR/CSS1.

1.2 Style

The developer defines a style for a number of reasons:

- To manage multiple items have the same non-default format.
- To change the default format applied to all items.
- To standardize the look of items across a set of related reports.

Styles in BIRT are similar to HTML cascading style sheets (CSS) , to Microsoft Word styles, and to Microsoft PowerPoint templates.. A style is a named object that defines the set of visual properties to apply to an element. Elements use the style simply by referencing it by name. Elements can *override* the properties from a style.

Styles allow the customer to create a consistent visual look for a set of reports, and to modify that look easily without the need to update reports manually. Styles are defined independent of any behavior or code that may be added to components.

---

[1] CSS was selected as the basis for the style system for a number of reasons. First, CSS is a public specification. Second, the people who use BIRT, are assumed to understand web technology and hence may already understand CSS. Third, many CSS books exist, making information about CSS readily available to users. Finally, our target output formats are HTML and FO, both of which are closely associated with CSS.

[2] The current CSS version at W3C is CSS 2.1, and CSS 3 is under review. However, Internet Explorer supports only CSS 1. Since IE accounts for the lion's share of the browser market, prudence suggests that we limit our support to only that which IE supports, plus those portions of CSS 2.1 that deal with print and are supported by XML-FO.

1.3 Terminology

Style	Visual properties such as font, color, spacing, etc.
Style sheet	A collection of styles.
CSS	Refers to the CSS1 specification unless otherwise noted.
Cascade	The process of applying style properties to a report by working from the root of the report tree down to the leaves.
Property	A named value for a style such as font-family or color.
Property Value	A value for a property such as “Arial” or “red”.
Block content	Report content that has a line break before and after. See CSS for details.
In-line content	Content that appears on the same output line as previous content (assuming there is sufficient space on the page.) See CSS for details.
Container	An element that can hold other elements.
Slot	Some elements provide multiple places to hold other elements. For example, a List provides a header, footer, detail, etc. Each of these is called a “slot.”

1.4 Overview

The developer can apply style to report items in one of several ways:

- Allow the element to inherit style properties from its container.
- Define a style that matches all elements of a given type or in a given context.
- Explicitly identify the style to be used for an element.
- Directly set the style properties for a given element.

Developers can define styles in the report design itself, or in a separate template or CSS document.

These options directly correspond to the way that styles are defined using CSS with HTML.

1.5 BIRT and CSS

Key goals in using CSS include:

- Leverage existing public specifications, such as CSS 1.
- Leverage existing knowledge both within the development team and with customers.
- Simplify translation of BIRT documents to HTML and XML-FO.

BIRT adopts the CSS specification as closely as possible, including properties, units, syntax and more. This section details how BIRT uses CSS. It lists the CSS properties that BIRT supports, and lists any differences between BIRT and CSS. Where BIRT uses the CSS specification, the reader should consult the CSS 1 specification for detailed information about the purpose and operation of the feature.

1.6 Cascading Style Sheets (CSS) and XML-FO

BIRT is designed to target the web environment and open standards. HTML 4.0 separates page content from page style. An HTML page provides the content; a Cascading Style Sheet (CSS) provides style information.

In the print world, XML-FO (Formatting Objects) is an emerging standard for using XML documents for page layout. While CSS targets style for the web, XML-FO targets style for print applications. XML-FO was designed with an eye toward CSS compatibility where possible.

CSS is quite well known by web developers, while XML-FO tends to be known by a small community of people who are concerned about print output.

To simplify the task of learning BIRT, and to ensure that reports work well on the web, the BIRT Report Object Model is designed to follow CSS wherever possible. Of course, neither HTML nor CSS have a concept of a report, or of content created automatically from a database. Hence, HTML and CSS don't have much to say about the structural portion of BIRT. However, HTML and CSS do have quite a bit to say about physical layout and visual style.

While CSS 1 says little about print, XML-FO says quite a bit. (CSS-2 added some print-oriented features, but they are not yet supported by Microsoft Internet Explorer. CSS-3, now in draft form, may add more support. However, without support of IE, the standards are of only academic interest at this point.)

1.7 UI Considerations

Some aspects of the CSS specification reflect its history as a text-based system instead of one designed for a GUI environment. This specification points out areas that are likely to be the most difficult for the UI. These sections are only suggestions; there is no guarantee that the assessment is complete or accurate.

One key consideration is that many CSS properties provide both a set of CSS-defined values along with the option to specify additional values. (See font-family.) Some properties allow lists of values. (Again, see font-family.) Some attributes cannot be set directly, bur rather only in conjunction with other attributes. This means that the UI must somehow link these properties to prevent the user from setting them in a way that cannot be expressed in CSS. (See the border properties: one cannot specify the style, color and width of one border independently of other border attributes.)

The UI may choose to ignore some CSS properties in the first release. For example, word-spacing and char-spacing would be seldom used in reports. The user an see the effect of these properties by previewing the report in HTML.

2. Selectors

This section discusses how ROM associates a style with a report element.

2.1 Background

The industry uses two distinct ways to associate styles with content. The ROM solution lies in the middle of the range.

The first technique, used by Microsoft Word, requires the content to name a style. For example, this paragraph is tagged with the "Body Text" style. The header is tagged with

the “Heading 1” style. Word treats all text the same, the only reason that a bit of text acts like a heading is that it is tagged with the “Heading 1” style. Said another way, the context is a uniform stream of text, with blocks of text “tagged” with a style name.

The second technique, used CSS and HTML, requires the style to “select” content. HTML provides document structure with tags such as <H1> for headings, <P> for paragraph text, etc. Styles associate themselves with the HTML elements using *selectors.* For example, one can set the style for level 1 heading by applying style to the HTML <H1> tag. CSS also provides the “class” attribute that lets an HTML element identify itself. CSS styles can then associate themselves with a “class.” A class is not a style name; it is simply an optional category to assist with creating CSS selectors.

Existing solutions also use two distinct solutions for style inheritance.

Microsoft Word uses inheritance of style definitions. Word has a Normal style that defines and sets all style properties. Derived styles (such as “Body Text”) inherit from the Normal style and override properties as required for that style. A block of content can both specify at most one style, and set style properties locally on that block.

CSS uses a “cascade” form of inheritance in which elements inherit style properties from their container. For example, text in a <P> tag inherits style from the overall HTML document. Zero or more CSS styles may “select” a given HTML element. The effective style for that element is a combination of all these styles as determined by the “cascade” algorithm. For example, we could set the font family for an HTML file to be “Arial”, set the font size for <P> elements to be 10 point, then assign the “important” class to a given paragraph, and write a style that makes “important” paragraphs use a bold weight.

That is, in Word, any block of text can identify just one style by name, and each style inherits from a single parent style. In CSS, individual styles do not inherit from other styles; instead, elements inherit properties from its container, and from any number of styles that select that element.

2.2 Style Selection in ROM

ROM adopts a selection solution based on CSS, but simplified.

- Elements inherit style properties from their container.
- Styles can select element types such as list, list header, chart, etc.
- Elements can identify one or more styles by name

CSS defines the concept of a “selector”: syntax that a style uses to specify the document content to which the style applies. CSS also uses the “class” concept to associate an element with a style. ROM uses a simplified form of this model in which the style name serves both purposes.

2.3 Predefined Style Names (Selectors)

The simplest form of selector is to identify a report element type, or element type and “slot”. A slot is a place where content can appear. For example, a list has slots for header, footer, column header and detail.

CSS specifies that a context is specified by a series of identifiers separated by spaces such as “p” or “h1 p” or “h3 li li”. ROM defines a fixed set of selectors: one for each element and element slot. Given that the set is fixed, ROM can adopt a simplified form of selector: assign each selector a name, and use that name as the style name. For

example, to select a list, use the “list” style. To select a list header, use the “list-header” style.

Unlike CSS, ROM does not allow style selectors to select arbitrary document structures. For example, one cannot select “list header table footer” to apply style to a table footer, but only when it appears inside a list header. This simplification is adopted because of the difference between reports and web documents. Web documents can be arbitrarily complex and are often generated automatically. Reports, however, tend to be created by hand. (Report instances, however, are automatically generated.)

For example, a style sheet writer may choose to apply certain style to top-level headers, another to second level headers. The style applies to any content in the headers. The same style sheet can apply to existing reports, new reports, and even reports created on the web because the style applies to report structure, not specific content.

Developers specify styles for the entire report using the “report” selector. Developers then apply formatting to specific bits of the report by using detailed selectors. See the “Cascade” section below for details on which styles apply to a given element.

The following is a list of the supported selectors. The list was created by creating a selector for each element, for each of the slots that is identical in Table and List, and for each of the element/slot combinations for any element that has multiple slots. Finally, additional styles were defined for elements with internal structure, such as charts.

Selector	Description
report	Overall default
list	List. Applies to all slots within the list.
group-header-*n*	List or table group header for level *n*
group-footer-*n*	List or table group footer for level *n*
list-header	List header
list-footer	List footer
list-group-header-*n*	List group header for level *n*
list-group-footer-*n*	List group footer for level *n*
list-detail	List Detail
table-header	Table Header
table-footer	Table Footer
table-group-header-*n*	Table Group Header *n*
table-group-footer-*n*	Table Group Footer *n*
table-column-header	Table Column Header
table-detail	Table Detail
page	Master page
matrix	Matrix

Selector	Description
matrix-row-header-*n*	Matrix row headers 1 through 9
matrix-row-footer-*n*	Matrix row footers 1 though 9
matrix-column-header-*n*	Matrix column headers 1 though 9
matrix-column-footer-*n*	Matrix column footers 1 though 9
matrix-cell	Matrix cell
chart	Chart
chart-title	Chart Title
chart-legend	Chart Legend
chart-value	Chart Value Label
chart-axis	Chart Axis Label
label	Label item
number	Data item that displays a number
date	Data item that displays a date or time
string	Data item that displays a string
text	Text item
grid	Grid item
toc	Table-of-contents item
toc-*n*	TOC level *n*
free-form	Free-form item
line	Line item
rectangle	Rectangle item
extended-item	Extended item

2.3.1 Note on HTML Rendering

The BIRT selectors work with the BIRT report design structure and clearly are not relevant to an HTML document. Similarly, HTML elements are not defined within a report design. Therefore, it is necessary for the Presentation Engine to convert the BIRT selectors into a format that will work with HTML, perhaps by using CSS classes to emulate the effect that BIRT selectors achieve.

2.3.2 UI Considerations

BIRT may provide a New Report Wizard to create reports. Such a wizard, if provided, will generate a report complete with formatting for titles, group headers, detail, and more. Advanced developers often want to customize this formatting to match a visual format defined by that particular company. The selectors above make this task easy. The

developer simply creates a style sheet that customizes these styles. The result is that the look of the wizard-generated report will follow the customer-defined styles.

2.4 Pseudo-Selectors

BIRT supports the same set of pseudo-selectors as CSS: a:link, a:visited, a:active, :first-line and :first-paragraph. The “a:” pseudo-selectors do not apply to printed output.

MS Internet Explorer supports only the literal “a:” forms of these selectors; the “:link” and related suffixes cannot be added to a CSS style. Therefore, BIRT will support the link pseudo-selectors as follows:

- The user can define styles with the names “a:link”, “a:visited” and “a:active”.
- The Presentation Engine copies these styles into the HTML output. They will apply to all links in the HTML generated for the report.

Customers who have a standard for how links should look on their web site can use the above feature to ensure that their reports use the same look as the rest of the site.

BIRT does not provide a way to customize the style of a link on a link-by-link basis.

Hyperlink styles apply, obviously, only to HTML pages shown in a browser. They do not apply to other forms of output such as print, PDF, etc.

2.5 Named Styles

HTML provides the “class” attribute to associate an HTML element with a specific style. BIRT provides a similar concept called “style.”[3] This attribute works identically to the HTML/CSS feature: it can be used to associate an element to a specific style.

ROM uses a simplified version of the CSS class selection model. ROM styles are named and either match one of the predefined names above (and act as a selector), or have a user-defined name that corresponds to an HTML class.

For example, an element may reference the “important” style. The report would then contain a style called “important”.

3. Style Cascade

Elements can reference any number of styles, directly or indirectly. For example, the styles that apply to a given numeric data item might include:

- The overall style for the report provides the font family.
- The context (table header) might provide the font size.
- The “numeric” selector might provide the numeric formatting and right alignment.
- The “accounting” style might provide red-when-negative behavior.
- The “important” style might cause the number to appear in bold.

This section explains the rules that determine the *effective* set of style properties for a given element. The term “cascade” is taken from CSS and describes the process of

[3] BIRT uses the term “class” for other purposes: referring to Java code attached to a report element. Hence, BIRT uses “style” to associate an element with a style.

styles flowing downward through the document hierarchy to each element. The ROM cascade is a simplified version of the CSS cascade.

3.1 Style Search

The CSS style cascade is defined top-down: one works from the root of the document tree downward applying style information. While the BIRT Factory and Presentation Engine may take this approach, the designer will often need to locate the property for an arbitrary element. To support this operation, ROM defines its cascade in the form of a property search that starts on a given element and works upward though the report structure.

The ROM style search is designed for efficiency. The search begins at an element and terminates when ROM locates a value for a given style property. The ROM cascade does not require the sophisticated search & sort algorithm of CSS.

The search is done for each style property. The search is done in three phases: highlight search, element search and context search.

3.1.1 Highlight Property Search

The first steps apply only to data items. First, we must locate any highlight rules applied to the data item. The highlight search has the following steps. It is done only at runtime in the Presentation Engine, but not at design time. It applies only to data items.

1. Using the style property search described below, determine if a set of highlight rules has been set. If not, skip the highlight rule processing, and continue with 6. Else, continue below to process the highlight.
2. Using the style property search described below, determine if a highlight expression has been set. Is so, evaluate the expression to get the highlight test value. If not, use the value of the data item itself (before applying any map rules) as the highlight test expression.

Note that this algorithm will find at most one highlight rule set. BIRT does not combine rule sets. Instead, a rule set that appears at one level “hides” any rule sets that may appear at a higher level.

3.1.2 Highlight Rule Search

If a set of highlight rules was found above, then we need to determine which rule within the set applies to this particular data item.

3. Scan the list of highlight rules found above. Find the first rule that evaluates true for the highlight test value found above.
4. If a rule is found, check if it defines a value for the property. If so, use it and stop the search.

3.1.3 Style Property Search

If the property is not found using the highlight rule search, or the highlight rule search does not apply, then perform a search on the element. The element search has the following steps.

5. Check if the element provides a value for the property. If so, use it.
6. Check if the element has named a style. If so, check if the style exists. If so, check if the style provides a value for the property. If so, use it.

7. Check if the element derives from another element. If so, repeat the search on the base element. The base element search uses only steps 5-7.

If the style property has not been found, and it is defined as one that can cascade, ROM then uses the context search.

8. Check if the report defines a style that selects this item. If so, check if that style provides a value for the property. If so, use it.
9. Determine the container and slot in which the element appears.
10. Check if the report defines a style that selects this container/slot combination. If so, check if that style provides a value for the property. If so, use it.
11. Otherwise, repeat the search (starting with step 5) using the container element.

As in CSS, some style properties (such as background color) do not cascade. The property definitions below define whether a property cascades or not. If a property does not cascade, then ROM skips the context search and simply uses the property's initial value.

3.1.4 Initial Values

If a property is not found using the above search, then it takes an initial value. The initial value is located by searching:

The initial value of a property in CSS is somewhat complex. If an HTML page does not specify a value, then it is set by the browser software or user preferences. For example, IE specifies some default font size and family. However, you can use the Internet Preferences dialog to change the size (perhaps make it bigger) and change the family. These user-defined defaults take affect if the HTML page does not override them.

There are two common models for handling initial values in reports. Both are valid, and the model to use for a given application depends on the needs of the customer.

- Web-like model: Initial values should be handled as for any other HTML page: use the browser defaults or the values that the user set in the Internet Preferences.
- Print-like model: The report should appear the same in every browser independent of any user settings. That is, the report should always use Arial, 10 point font (say) even if the user has indicated that they prefer 12 point font for web pages.

Internal discussions reveal that both models have strong proponents, and so ROM should support both. Luckily, as we'll see below, the print-like model is just a special case of the web-like model.

BIRT uses the following rules for defaults:

- Unset style properties are said to use the *initial* style property value as described here.
- In the web environment, the initial style property is set by the CSS User Agent (UA, browser). The values, and how the user overrides them, are UA-implementation-specific.
- A developer who wants to use application-specific defaults creates a BIRT style that selects the report as a whole. The developer then sets values for all properties (or at least the properties that the UA allows the user to customize.) For example, the

developer can “lock down” font family, font size, colors, hyperlink coloring, etc. The report-specific defaults “hide” the initial values set for the UA.

- A developer who wants to create application-specific defaults for a set of reports can use a library to apply the same report-wide style to the set of reports.
- The DE client is responsible for providing the initial values for a given environment. For example, the Eclipse Report Designer (ERD) may choose initial values that emulate Internet Explorer, and may allow the user to override these values using preferences. The Presentation Engine may choose defaults that work well for the target output environment.
- ROM defines an ultimate set of initial values that take effect if the application does not provide its own initials. The application can choose whether to use these initial values or not. (For example, the property sheet may always want to show a value for a property, but the Presentation Engine may want to use a null value in place of the initial value to avoid generating CSS properties unnecessarily.)
- To simplify initial value management, the Design Engine provides a means to set the initial values. The DE will then add the defaults to its cascade algorithm so that each client need not reinvent the wheel.

The property detail shows the set of initial value for each property. This information is taken from Appendix F of the CSS 2.1 specification (http://www.w3.org/TR/CSS21/propidx.html).

3.1.5 Alternative Algorithm

Note that, in the Design Engine, steps 1-3 cannot be done, as items do not yet have actual data values. Note also that the Presentation Engine the actual implementation algorithm may be different. For example, the Presentation Engine can achieve the same results by working with sets of properties:

- Create a list that contains properties as name/value pairs.
- Traverse to the most-base element. Copy all its properties into the property list.
- Repeat with each child element down though the inheritance hierarchy.
- If the element has a style selector, copy that style’s properties into the list.
- If the element has private style properties, copy those into the list.
- If the element has an associated highlight rule, copy the rule’s properties into the list.

The result of this is a list that contains all properties that apply to this item specifically. Such properties would override any properties inherited by context though containment. Note also that later steps in the above algorithm **replace** values found in earlier steps. The net result is that any given property takes the same value in the above batch algorithm as it does from using steps 1-7 above in the property-by-property algorithm.

4. Formatting Model

The ROM formatting model follows that of CSS. BIRT defers most formatting to the browser or FO processor. Hence, the CSS user agent (UA) will do the actual visual formatting, and so all formatting must be expressed in a form that can be passed, via CSS, to the UA.

See the CSS1 specification for an explanation of the formatting model. This section discusses formatting issues unique to BIRT.

4.1 Block-Level Elements

ROM uses the term “section” and “subsection” to identify top-level content within the report. By default, all sections are block-level elements. Block-level elements have an implied line break before and after the element. For example, when a chart appears as a section, BIRT will put the chart on a line by itself. Items inside a grid, free-form, matrix, or table container are never block-level elements.

4.2 In-line Elements

An in-line element is one that appears on a line. All text and value fields within a text element are in-line elements. The contents of a grid, table or matrix are in-line elements.

The user can make some items act as in-line elements even when those items act as sections. Use the CSS ‘display’ property to set the item to ‘inline’. For example, a report that produces many small charts may want to “tile” them across the page.

Note: *Need to determine which elements won’t work as in-line content, perhaps due to the limitations of HTML.*

4.3 Font Matching

ROM adopts the CSS font matching algorithm. See the CSS 1 specification, section 5.2.1 for details.

4.4 Unsupported Features

ROM does not support HTML-style lists, nor does it support floating content.

5. Style Definition

A style is defined by a name and a set of style property values. Most styles define values for a small subset of possible values. Unset values indicate that ROM should use the style cascade (discussed below) to find the values elsewhere.

5.1 The Style Sheet

Styles are collected into a *style sheet.* The style sheet is can be defined as part of a single report, or can be places in a template so that it can be shared by multiple reports. The style sheet contains any number of styles. Each style provides formatting rules for fonts, borders, colors, numeric formatting and more. A developer can create one style for showing group headers, another for data in a detail frame and a third for totals.

5.2 Style Properties

A style is a collection of *properties* such as font face name, font color, fill color and so on. A property is simply a (name, value) pair. The name identifies the property, and the value is what has been set for the property: “red” or “Arial” or “10 pt.”

A property value can be blank, meaning that the user has not specified anything for that property. The idea of a blank property is the basis for style inheritance.

5.2.1 Unsupported CSS properties

The following CSS1 properties are not supported by ROM.

- list-style-type
- list-style-type
- list-style-position
- list-style
- height, width: These are element properties in ROM, not style properties.
- clear

The user can define CSS properties that are passed though to the browser. If a style contains an unsupported CSS property, then that style is included with the style when sent to the browser, though it will not appear in the BIRT UI.

5.3 Style Element

A BIRT report can define “shared” styles. Every style must have a name, and that name must be unique with the report design.

Summary

Base element: Report Element

Availability: First release

XML Element Name: `style`

Inherited Properties

`name`

Required. The name must be unique across the set of styles within the design file.

`extends`

As in CSS, ROM styles cannot inherit from another style.

Properties

The following table identifies the style properties that BIRT supports. Most of these are taken from CSS 1, though some print-related properties come from CSS 2.1. Further, BIRT defines some additional properties specific to reporting.

`backgroundAttachment`

TBD.

`backgroundColor`

The background or “fill” color for each element.

`backgroundImage`

An image to appear in the background of page or report elements.

`backgroundPositionX`

Implements the CSS background-position property.

`background-position-y`

Implements the CSS background-position property.

`backgroundRepeat`

TBD.

`borderTopColor`
`borderLeftColor`
`borderBottomColor`
`borderRightColor`

Sets the color of the border. Implements the CSS border-color property.

`borderTopStyle`
`borderLeftStyle`
`borderBottomStyle`
`borderRightStyle`

The line style of the border. Implements the CSS border-style property.

`borderTopWidth`
`borderLeftWidth`
`borderBottomWidth`
`borderRightWidth`

The width of the border.

`can-shrink`

Whether an item can shrink based on its contents.

`color`

Foreground color, usually for text.

`dateTimeFormat`

The display format for date/time values displayed in a data item.

`display`

Specifies if a top-level element should be a block or in-line element.

`fontFamily`

The font family name: user defined or a CSS generic family.

`fontSize`

The size (height) of the text.

`fontStyle`

Normal or italic.

`fontVariant`

Normal or small caps.

`fontWeight`

Level of “boldness.”

`highlight`

Rules for changing the visual formatting of data items based on a data value.

`letterSpacing`

The spacing between individual letters.

`lineHeight`

Height of a line. Implies spacing between lines.

`map`

Provides a mechanism for mapping internal database values to user-visible display values.

`marginTop`

`marginLeft`

`marginBottom`

`marginRight`

Provides spacing between paragraphs in text, and around block-level content.

`masterPage`

If the item starts a new page, this property selects the master page to display.

`numberFormat`

The format string to apply to data items that display numbers.

`orphans`

Controls whether orphans appear on a page.

`paddingTop`

`paddingLeft`

`paddingBottom`

`paddingRight`

The amount of space between the border and the contents of a report item.

`pageBreakAfter`

Causes a page break before the report item.

`pageBreakBefore`

Causes a page break after the report item.

`pageBreakInside`

Causes a page break between detail items of a list.

`showIfBlank`

If false, hides a data item if it is blank.

`stringFormat`

The format string to apply to string data shown in a data item.

`textAlign`

How to align the text: left, center or right.

`textDecoration`

TBD.

`textUnderline`

Part of implementation of CSS text-decoration.

`textOverline`

Part of implementation of CSS text-decoration.

`textLineThrough`

Part of implementation of CSS text-decoration.

`textIndent`

The indentation for text within a Text or Multi-line text item.

`textTransform`

Transforms for text such as all-upper case.

`verticalAlign`

The vertical alignment: top, middle or bottom.

`whiteSpace`

Whether to ignore or keep white space in a Text or Multi-Line text item formatted in HTML.

`widows`

Whether to suppress widows in pagination

`wordSpacing`

The amount of spacing between words.

5.4 Font Properties

The next several sections discuss the various style properties. Subsections describe the properties in detail using the information from the CSS 1 spec where applicable. See the CSS 1 specification for a description of the syntax used to describe values.

5.4.1 `fontFamily` Property

The font name: a user-defined name or a CSS-defined name.

Summary

Display name	Font Family
Property group	Font
Value	[[<family-name> | <generic-family>],]* [<family-name> | <generic-family>]
Initial	UA specific
Applies to	all elements
Inherited	yes
Percentage values	N/A

CSS equivalent	font-family
CSS compatibility	Defined in CSS 1
CSS section	5.2.2
ROM type	String that contains a list of fonts formatted according to CSS rules.
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Choices

The following generic families are defined:

- 'serif' (e.g. Times)
- 'sans-serif' (e.g. Helvetica)
- 'cursive' (e.g. Zapf-Chancery)
- 'fantasy' (e.g. Western)
- 'monospace' (e.g. Courier)

Description

Provides the font name. This can be a simple name (“Arial”), or a list. (“Arial, sans-serif”).

5.4.2 `fontStyle` Property

The font style: italic, etc.

Summary

Display name	Font Style
Property group	Font
Value	normal | italic | oblique
Initial	normal
Applies to	all elements
Cascaded	yes
Percentage values	N/A
CSS equivalent	font-style
CSS compatibility	Defined in CSS 1
CSS section	5.2.3
ROM type	choice
JavaScript type	String
Settable at runtime	Yes

Availability	First Release

Choices

Display Name	Internal Name	Description
Normal	normal	Normal, vertical text
Italic	italic	Italic text
Oblique	oblique	Oblique text

Description

Controls the italic attribute of the text.

5.4.3 `fontVariant` Property

Whether to show the data in small-caps.

Summary

Display name	Font Variant
Property group	Font
Value	normal | small-caps
Initial	normal
Applies to	all elements
Cascaded	yes
Percentage values	N/A
CSS equivalent	font-variant
CSS compatibility	Defined in CSS 1
CSS section	5.2.4
ROM type	choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Choices

Display Name	Internal Name	Description
Normal	normal	Normal text
Small Caps	small-caps	Text in SMALL CAPS

Description

Allow the user to format the text in small caps.

5.4.4 `fontWeight` Property

Controls the boldness of the text.

Summary

Display name	Font Weight
Property group	Font
Value	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
Initial	normal
Applies to	all elements
Cascaded	yes
Percentage values	N/A
CSS equivalent	font-weight
CSS compatibility	Defined in CSS 1
CSS section	5.2.5
ROM type	choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Choices

Display Name	Internal Name	Description
Normal	normal	Normal weight text
Bold	bold	Bold setting for the font
Bolder	bolder	Relatively bolder than the current font
Lighter	lighter	Relatively lighter then the current font
	100	
	200	
	300	
	400	
	500	
	600	
	700	
	800	
	900	

Description

Controls the boldness of the font using the CSS system.

5.4.5 `fontSize` Property

The size of the font.

Summary

Display name	Font Size
Property group	Font
Value	<absolute-size> | <relative-size> | <length> | <percentage>
Initial	medium
Applies to	all elements
Cascaded	yes
Percentage values	relative to parent element's font size
CSS equivalent	font-size
CSS compatibility	Defined in CSS 1
CSS section	5.2.6
ROM type	Dimension with extended choices
JavaScript type	String
Settable at runtime	Yes

Availability	First Release

Choices

Display Name	Internal Name	Description
	xx-small	absolute-size as defined by CSS.
	x-small	
	small	
	medium	
	large	
	x-large	
	xx-large	
	larger	Relative-size values as defined by CSS.
	smaller	

Description

Possible values for absolute-size are: [xx-small | x-small | small | medium | large | x-large | xx-large].

Possible values for relative-size values are: [larger | smaller].

Negative values are not allowed.

On all other properties, 'em' and 'ex' length values refer to the font size of the current element. On the 'font-size' property, these length units refer to the font size of the parent element.

5.4.6 `color` Property

The foreground color for text and similar items.

Summary

Display name	Foreground Color
Property group	Font
Value	<color>
Initial	UA specific
Applies to	all elements
Cascaded	yes
Percentage values	N/A
CSS equivalent	color
CSS compatibility	Defined in CSS 1
CSS section	5.3.1

ROM type	Color
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Description

Note that CSS defines the color property in a “color and background property” group. However, the color property generally applies to text in ROM, and so is described here.

5.5 Background Properties

5.5.1 `backgroundColor` Property

The background (fill) color of an item.

Summary

Display name	Background Color
Property group	Background
Value	<color>
Initial	transparent
Applies to	all elements
Cascaded	no
Percentage values	N/A
CSS equivalent	color
CSS compatibility	Defined in CSS 1
CSS section	5.3.2
ROM type	Color
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Description

CSS defines a “transparent” value that says that the background color is unset. ROM achieves the same result if the property is unset.

5.5.2 `backgroundImage` Property

TBD

Summary

Display name	Background Image

Property group	Background
Value	<url>
Initial	none
Applies to	all elements
Cascaded	no
Percentage values	N/A
CSS equivalent	background-color
CSS compatibility	Defined in CSS 1
CSS section	5.3.3
ROM type	URI
JavaScript type	String
Settable at runtime	Yes
Availability	After the first Release

Description

CSS defines a “none” value as a valid choice. ROM defines this choice as an unset property.

5.5.3 `backgroundRepeat` Property

TBD

Summary

Display name	Background Repeat
Property group	Background
Value	repeat | repeat-x | repeat-y | no-repeat
Initial	repeat
Applies to	all elements
Cascaded	no
Percentage values	N/A
CSS equivalent	Background-repeat
CSS compatibility	Defined in CSS 1
CSS section	5.3.4
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	After the first Release

5.5.4 `backgroundAttachment` Property

TBD

Summary

Display name	Background Attachment
Property group	Background
Value	scroll | fixed
Initial	scroll
Applies to	all elements
Cascaded	no
Percentage values	N/A
CSS equivalent	background-attachment
CSS compatibility	Defined in CSS 1
CSS section	5.3.5
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	After the first Release

5.5.5 `backgroundPositionX` Property

Summary

Display name	Background Position X
Property group	Background
Value	<percentage> | <length> | left | center | right
Initial	0%
Applies to	block-level and replaced elements
Cascaded	no
Percentage values	refer to the size of the element itself
CSS equivalent	background-position
CSS compatibility	ROM property that implements the CSS background-position property defined in CSS 1.
CSS section	5.3.5
ROM type	Dimension with extended choice
JavaScript type	String

Settable at runtime	Yes
Availability	After the first Release

Description

The CSS 'background-position' property sets both the x and y position for the background. The ROM model prefers to have each value defined by a distinct property. Hence, ROM defines two properties to represent this CSS property: 'background-position-x' and 'background-position-y'.

5.5.6 `backgroundPositionY` Property

TBD

Summary

Display name	Background Position X
Property group	Background
Value	<percentage> | <length> | top | center | bottom
Initial	0%
Applies to	block-level and replaced elements
Cascaded	no
Percentage values	refer to the size of the element itself
CSS equivalent	background-position
CSS compatibility	ROM property that implements the CSS background-position property defined in CSS 1.
CSS section	5.3.5
ROM type	Dimension with extended choice
JavaScript type	String
Settable at runtime	Yes
Availability	After the first Release

5.6 Text Properties

5.6.1 `wordSpacing` Property

Summary

Display name	Word Spacing
Property group	Text

Value	normal | <length>
Initial	normal
Applies to	all elements
Cascaded	yes
Percentage values	N/A
CSS equivalent	'word-spacing'
CSS compatibility	Defined in CSS
CSS section	5.4.1
ROM type	Dimension with extended choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.6.2 `letterSpacing` Property

The spacing between individual letters.

Summary

Display name	Letter Spacing
Property group	Text
Value	normal | <length>
Initial	normal
Applies to	all elements
Cascaded	yes
Percentage values	N/A
CSS equivalent	letter-spacing
CSS compatibility	Defined in CSS
CSS section	5.4.2
ROM type	Dimension with extended choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.6.3 `textUnderline` Property

Summary

Display name	Text Underline
Property group	Text
Value	none | underline
Initial	none
Applies to	all elements
Cascaded	no, but see clarification in the CSS1 spec.
Percentage values	N/A
CSS equivalent	text-decoration
CSS compatibility	ROM property that implements the text-decoration property defined in CSS 1.
CSS section	5.4.3
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Description

CSS defines the `text-decoration` property to set underline, `overline` and strikethrough styles. ROM divides this CSS property into three: `textUnderline`, `textLineThrough` and `textOverline`. ROM does not support the blink attribute of text-decoration.

5.6.4 `textOverline` Property

Summary

Display name	Text Overline
Property group	Text
Value	none | overline
Initial	none
Applies to	all elements
Cascaded	no, but see clarification in the CSS1 spec.
Percentage values	N/A
CSS equivalent	text-decoration
CSS compatibility	ROM property that implements the text-decoration property defined in CSS 1.

CSS section	5.4.3
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.6.5 `textLineThrough` Property

Summary

Display name	Text Line Through
Property group	Text
Value	none | line-through
Initial	none
Applies to	all elements
Cascaded	no, but see clarification in the CSS1 spec.
Percentage values	N/A
CSS equivalent	text-decoration
CSS compatibility	ROM property that implements the text-decoration property defined in CSS 1.
CSS section	5.4.3
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.6.6 `horizAlign` Property

Summary

Display name	Horizontal Align
Property group	Text
Value	auto | left | center | right
Initial	auto
Applies to	Elements inside table or grid cells, sections within a page.
Cascaded	no
Percentage values	N/A
ROM type	Choice

JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.6.7 `verticalAlign` Property

Summary

Display name	Vertical Align
Property group	Text
Value	baseline | sub | super | top | text-top | middle | bottom | text-bottom | <percentage>
Initial	baseline
Applies to	inline elements
Cascaded	no
Percentage values	refers to the 'line-height' of the element itself
CSS equivalent	vertical-align
CSS compatibility	Defined by CSS 1
CSS section	5.4.4
ROM type	Percentage with extended choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.6.8 `textTransform` Property

Summary

Display name	Text Transform
Property group	Text
Value	capitalize | uppercase | lowercase | none
Initial	none
Applies to	all elements
Cascaded	yes
Percentage values	N/A
CSS equivalent	text-transform

CSS compatibility	Defined by CSS 1
CSS section	5.4.5
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.6.9 `textAlign` Property

Summary

Display name	Text Align
Property group	Text
Value	left | right | center | justify
Initial	UA specific
Applies to	block-level elements
Cascaded	yes
Percentage values	N/A
CSS equivalent	text-align
CSS compatibility	Defined by CSS 1
CSS section	5.4.6
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.6.10 `textIndent` Property

Summary

Display name	Text Indent
Property group	Text
Value	<length> | <percentage>
Initial	0
Applies to	block-level elements
Cascaded	yes
Percentage values	refer to parent element's width

CSS equivalent	text-indent
CSS compatibility	Defined by CSS 1
CSS section	5.4.7
ROM type	Dimension
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.6.11 `lineHeight` Property

Height of a line. Implies spacing between lines.

Summary

Display name	Line Height
Property group	Text
Value	normal | <number> | <length> | <percentage>
Initial	normal
Applies to	all elements
Cascaded	yes
Percentage values	relative to the font size of the element itself
CSS equivalent	line-height
CSS compatibility	Defined by CSS 1
CSS section	5.4.8
ROM type	Dimension plus extended choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Description

Height of a line. Implies spacing between lines.

5.6.12 `whiteSpace` Property

Summary

Display name	White Space
Property group	Text

Value	normal | pre | nowrap
Initial	normal
Applies to	block-level elements
Cascaded	yes
Percentage values	N/A
CSS equivalent	white-space
CSS compatibility	Defined by CSS
CSS section	5.6.2
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.7 Box Properties

5.7.1 `marginTop, marginRight, marginBottom, marginLeft` Properties

Provides alignment of block-level content on the page, and space around some in-line content.

Summary

Display name	Margin Top, Margin Right, Margin Bottom Margin Left
Property group	Box
Value	<length> | <percentage> | auto
Initial	0
Applies to	all elements
Cascaded	no
Percentage values	refer to width of the closest block-level ancestor
CSS equivalent	
CSS compatibility	Defined by CSS 1
CSS section	5.5.1
ROM type	Dimension plus extended choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Description

Amount of space around the report item outside of the border.

5.7.2 `paddingTop`, `paddingRight`, `paddingBottom`, `paddingLeft` Property

Amount of space between report item content and the border.

Summary

Display name	Padding Top, Padding Right, Padding Bottom, Padding Left
Property group	Box
Value	<length> | <percentage>
Initial	0
Applies to	all elements
Cascaded	no
Percentage values	refer to width of closest block-level ancestor
CSS equivalent	
CSS compatibility	Defined by CSS 1
CSS section	5.5.6
ROM type	Dimension
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.7.3 `borderTopWidth`, `borderRightWidth`, `borderBottomWidth`, `borderLeftWidth` Properties

Summary

Display name	Border Top Width, Border Left Width, Border Bottom Width, Border Right Width
Property group	Box
Value	thin | medium | thick | <length>
Initial	'medium'
Applies to	all elements
Cascaded	no
Percentage values	N/A
CSS equivalent	border-top-width, border-left-width, border-bottom-width, border-right-width

CSS compatibility	Defined by CSS 1
CSS section	5.5.11
ROM type	Length plus extended choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.7.4 `borderTopColor`, `borderLeftColor`, `borderBottomColor` and `borderRightColor` Properties

Summary

Display name	Border Top Color, Border Left Color, Border Bottom Color, Border Right Color
Property group	Box
Value	<color>
Initial	the value of the 'color' property
Applies to	all elements
Cascaded	no
Percentage values	N/A
CSS equivalent	border-color
CSS compatibility	ROM properties that implement the CSS border-color property defined in CSS 1.
CSS section	5.5.16
ROM type	Color
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Description

CSS defines the ‘border-color’ property to set the colors of the four borders. Again, ROM prefers to set one attribute with each property. ROM defines four properties to implement this CSS property: ‘border-top-color’, ‘border-right-color’, ‘border-bottom-color’ and ‘border-left-color’.

ROM creates the CSS ‘border-color’ property by following these rules:

- Check if any of the four border color properties are set. If not, don’t create the CSS border-color property.
- Check if the border-top-color property is set. If it is, use that color as the first border-color attribute. Otherwise, use the element color as the first border-color attribute.

- Check if the border-left-color property is set. If so, use it as the second border-color attribute. If not, then check if the border-bottom-color or border-right-color properties are set. If not, stop. If so, fill in the element color as the second border-color attribute.
- Check if the border-bottom-color property is set. If so, use it as the third border-color attribute. If not, then check if the border-right-color property is set. If not, stop. If so, fill in the element color as the third border-color attribute.
- Check if the border-right-color property is set. If so, use it as the fourth border-color attribute.

5.7.5 `borderTopStyle`, `borderLeftStyle`, `borderBottomStyle` and `borderRightStyle` Properties

Summary

Display name	Border Top Style, Border Left Style, Border Bottom Style, Border Right Style
Property group	Box
Value	none | dotted | dashed | solid | double | groove | ridge | inset | outset
Initial	none
Applies to	all elements
Cascaded	no
Percentage values	N/A
CSS equivalent	border-style
CSS compatibility	ROM properties that implement the CSS border-style property defined in CSS 1.
CSS section	5.5.17
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Description

CSS defines the ‘border-style property to set the line style of the four borders. Again, ROM prefers to set one attribute with each property. ROM defines these four properties to implement this CSS property.

ROM creates the CSS ‘border-style’ property by following rules similar to those for border-color.

5.8 String, Number and Date/Time Format Properties

The string format rules apply when a data control displays a string (text) value; the number rules when the data item displays a numeric value; and the date/time rules when the item displays a date/time value.

5.8.1 `numberFormat` Property

Format string applied to numeric results.

Summary

Display Name	Number Format
Property group	Format
ROM Type	String
JavaScript Type	String
Default value	None
Inherited	Yes
Settable at runtime	No
Availability	First release

Description

BIRT uses Java numeric formatting defined by the `DecimalFormat` class. The Java formats are closely patterned after Excel, but there are some differences. See the `DecimalFormat` Javadoc for details of the supported format strings.

Java and Excel Format Similarities

Both Excel and Java format supports:

- 0, # for numbers
- . for decimal point, substituted by locale-specific decimal point
- , for group separator, substituted by locale-specific decimal point
- % for percentage
- ; for conditional formats

Java and Excel Format Differences

Excel and Java have the following differences in syntax:

- Excel supports “ as quotation mark; Java uses ‘.
- In Excel, anything in quotation mark is treated as literal, i.e., “\” gives you \. In Java, \ is not special; to input ‘ character you use ‘’. Also, ‘abc’’def’ means abc’def, while in Excel “abc\”def means abc\def, “abc\”def” is wrong syntax, \”abc\” means “abc”.
- Excel uses e+, e-, E+, E- all to represent scientific notation (according to spec). Tests in Excel shows one cannot use e+ or e-. Java only allows E. Also, in Excel, you have to have – or + following E or e. In Java, you can have E5 instead of E+5.
- In Excel format, you explicitly quote the currency symbol, except for $. $ causes no substitution, and always renders as $. In Java, a Unicode char \u00A4 causes the substitution of the locale-specific currency symbol. If you want specific currency symbol, like $, you say ‘$’0 in Java. Note that in Excel, you can directly say $0.

- Java supports a – for minus sign. Java substitutes a locale-specific minus character in the formatted string.
- Java supports localized “per mille” (parts per thousand) using Unicode char \u2030.

Excel Features not Supported by Java

Excel has the following features not supported by Java formatting:

- Adding Space for alignment. Excel supports the _ character. For example, (_)) creates a space after the number whose width is the same as the width for character ‘)’. Similarly, (_a) creates a space after the number whose width is the same as the width for character ‘a’. The former is most frequently used in conditional formats to display negative numbers with () yet keeping the alignment of both positive and negative numbers.
- Repeating characters. A character following a * character causes the character to be appended to the end of the number, and fills the whole cell width. For example, 0*- displays 100---------- if the number is 100. BIRT cannot support this in HTML as we do not know the display width.
- Use ? to force decimal align. In fact, CSS supports decimal alignment. Since Java does not support this, we do not support it for open-source release. For commercial release, we can decide later.
- Scientific notation. In Excel format, the scientific notation has the format of E+, e+, E-, e-. the result might be E(e)5, E(e)-5 or E(e)+5, E(e)-5. The E5 is also possible result for 00E-0. BIRT formats support E only, not e.
- Conditional formats. Excel supports arbitrary condition, i.e., you can have up to 3 formats for each format string, i.e., “[Red][<100];[Blue][>100]:General”. We will not support conditions other than >0, <0 and =0. Excel limits the user to only eight colors: black, blue, cyan, green, magenta, red, white and yellow. Java formatting does not support colors. Instead, the user can apply conditional colors using highlighting rules. (BIRT highlighting rules are not limited to the eight Excel colors, the user can use any CSS, user-defined or RGB color.) The UI might help the user build the format and color together.

See Also

See the following for more information:

- http://java.sun.com/j2se/1.4.2/docs/api/java/text/DecimalFormat.html for the Java number formatter class.
- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vblr7/html/vafmtuserdefinednumericformats.asp for information on Excel formatting.
- http://office.microsoft.com/assistance/hfws.aspx?AssetID=HP051986791033&CTT=1&Origin=EC010229841033&QueryID=8Ax53w4jL0

5.8.2 `dateTimeFormat` Property

Format string for displaying date/time values.

Summary

Display name	Date Time Format

Property group	Format
ROM type	String
JavaScript type	String
Default value	None
Inherited	Yes
Settable at runtime	No
Availability	First release

Standard Choices

In addition to the Excel rules, ROM defines the following additional locale-independent date formats:

Keyword	Description	Result
General Date	Returns a date and time in the Short Date Long Time format as defined in the user's Control Panel/Locale Map file	01/23/2001 8:53:03PM
Long Date	Returns a Long Date as defined in the user's Locale Map file	January 23,2001
Medium Date	Returns a date with the month name abbreviated to 3 letters: dd-mm-yy	23-Jan-01
Short Date	Returns a Short Date as defined in the user's Control Panel/Locale Map File	01-23-2001
Long Time	Returns a Long Time as defined in the user's Control Panel/Locale Map file	8:45:00 PM
Medium Time	Returns hours and minutes in 12-hour format, including AM/PM designation (hh:nn AM/PM)	8:45:00 PM
Short Time	Returns hours and minutes in 24-hour format (hh:nn)	20:45

Description

The format string follows the rules defined by the Java `DateFormat` class, which is based on Excel formatting.

See Also

See the following for more information:

- http://java.sun.com/j2se/1.4.2/docs/api/java/text/DateFormat.html for the Java number formatter class.
- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vblr7/html/vafmtuserdefinednumericformats.asp for information on Excel formatting.

5.8.3 `stringFormat` Property

Format string to apply to string data values.

Summary

Display Name	String Format
Property group	Format
ROM Type	String
JavaScript Type	String
Default value	None
Inherited	Yes
Settable at runtime	No
Availability	First release

Description

The following table shows the symbols or characters you can use to create your own user-defined formats, and the effect of each symbol on the resulting format string.

Symbol	Description	Example/Result

@, &	Character placeholders. Returns a character, or a space if there is no character in the corresponding position. Placeholders are filled from right to left unless <format pattern> uses a ! character.	`Format$("1111212111","(@@@) @@@-@@@@")`
		
		`Returns (111) 121-2111`
		`Format$("6789999","(&&&) &&&-&&&&")`
		
		`Returns (   ) 678-9999`

<	Lowercase conversion. Converts <exprs to format> to lowercase.	`Format$("Smith, John","<")` `Returns smith, john`
>	Uppercase conversion. Converts <exprs to format> to uppercase.	`Format$("Smith, John",">")` `Returns SMITH, JOHN`

!	Specifies that placeholders are to be filled from left to right.	`Format$("5551212","(&&&) &&&-&&&&")`
		
		`Returns (   ) 555-1212`
		`Format$("5551212","!(&&&) &&&-&&&&")`
		
		`Returns (555) 121-2`

In addition the BIRT-defined “HTML” format has special meaning for strings. This format treats the string as a literal HTML value to be included in the web page. By default, and with all other formats, BIRT will encode the string value so that it appears correctly in HTML. If the format is “HTML”, this escaping is not done, and the value can contain HTML. This is most useful for inserting custom HTML into a <value-of> element in a Text element.

5.8.4 `numberAlign` Property

Alignment of numeric data.

Summary

Display Name	Number Align
Property group	Format
ROM Type	Choice
JavaScript Type	String
Default value	Right
Inherited	Yes
Settable at runtime	No
Availability	First release

Choices

Same choices as Text Align.

Description

A special alignment property is provided to set alignment for numbers. This property allows a single style to work for both textual and numeric data: BIRT will choose the proper alignment for each data type. Dates are assumed to use the same alignment as strings.

5.9 Section Properties

5.9.1 `display` Property

Controls whether the section appears as block or in-line content.

Summary

Display name	Display
Property group	Section

Value	block | inline | none (BIRT does not support the 'list-item' choice.)
Initial	block
Applies to	all elements
Cascaded	no
Percentage values	N/A
CSS equivalent	display
CSS compatibility	Defined in CSS 1.
CSS section	5.6.1
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.9.2 `masterPage` Property

Name of the master page on which to start this section.

Summary

Display Name	Master Page
Property group	Section
ROM Type	Element Reference
JavaScript Type	String
Default value	None
Inherited	Yes
Settable at runtime	No
Availability	First release

Description

Name of the master page on which to start this section. If blank, the normal page sequence is used. If defined, the section starts on a new page, and the master page is the one defined here. The subsequent pages are those defined by the report's page sequence.

5.10 Pagination Properties

5.10.1 `orphans` Property

Controls whether the first line of a paragraph appears by itself at the bottom of the page.

Summary

Display name	Orphans

Property group	Text
Value	<integer> | inherit
Initial	2
Applies to	block-level elements
Cascaded	yes
Percentages	N/A
CSS compatibility	Defined in CSS 2.1
CSS section	CSS 2.1, Section 13.3.2
ROM type	Integer
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Description

A “widow” occurs when the last line of a multi-line paragraph appears on its own at the top of a page due to a page break. An “orphan” occurs if the first line of a multi-line paragraph appears on its own at the bottom of a page due to a page break.

The unset value in ROM corresponds to the inherit value in CSS.

5.10.2 `widows` Property

Controls whether the last line of a multi-line paragraph appears on its own at the top of a page.

Summary

Display name	Widows
Property group	Text
Value	<integer> | inherit
Initial	2
Applies to	block-level elements
Cascaded	yes
Percentages	N/A
CSS equivalent	widows
CSS compatibility	Defined in CSS 2.1
CSS section	CSS 2.1, Section 13.3.2
ROM type	Integer
JavaScript type	String
Settable at runtime	Yes

Availability	First Release

Description

A “widow” occurs when the last line of a multi-line paragraph appears on its own at the top of a page due to a page break. An “orphan” occurs if the first line of a multi-line paragraph appears on its own at the bottom of a page due to a page break.

The unset value in ROM corresponds to the inherit value in CSS.

5.10.3 `pageBreakBefore` Property

Summary

Display name	Page Break Before
Property group	Section
Value	auto | always | avoid | left | right | inherit
Initial	auto
Applies to	block-level elements
Cascaded	no
Percentages	N/A
CSS equivalent	page-break-before
CSS compatibility	Defined in CSS 2.1
CSS section	CSS 2.1, Section 13.3.1
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.10.4 `pageBreakAfter` Property

Summary

Display name	Page Break After
Property group	Section
Value	auto | always | avoid | left | right | inherit
Initial	auto
Applies to	block-level elements
Cascaded	no
Percentages	N/A
CSS equivalent	page-break-after

CSS compatibility	Defined in CSS 2.1
CSS section	CSS 2.1, Section 13.3.1
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

5.10.5 `pageBreakInside` Property

Summary

Display name	Page Break Inside
Property group	Section
Value	avoid | auto | inherit
Initial	auto
Applies to	block-level elements
Cascaded	yes
Percentages	N/A
CSS equivalent	page-break-inside
CSS compatibility	Defined in CSS 2.1
CSS section	CSS 2.1, Section 13.3.1
ROM type	Choice
JavaScript type	String
Settable at runtime	Yes
Availability	First Release

Description

Using a value of ‘avoid’ produces the same effect as a “keep together”, “keep with next” or “keep with previous” property.

5.11 Visibility Properties

5.11.1 `showIfBlank` Property

Controls whether to display a report item if it contains no data.

Summary

Display Name	Show If Blank
Property group	Visibility
ROM Type	Boolean

JavaScript Type	Boolean
Default value	True
Inherited	Yes
Settable at runtime	No
Availability	First release

Description

Show this report item even if it is empty, or all its data elements are empty. If false, the report item is automatically hidden when empty.

5.11.2 `canShrink` Property

Whether a report item shrinks to fit its content.

Summary

Display Name	Can Shrink
Property group	Visibility
ROM Type	Boolean
JavaScript Type	Boolean
Default value	None
Inherited	Yes
Settable at runtime	No
Availability	After the first release

Description

Whether the section can shrink if the actual content is smaller than the design size. The default is to shrink. (Note that sections will automatically grow if their content is too large. This is necessary to prevent truncating the content and losing information.)

5.11.3 `highlightTestExpr` Property

Expression for conditional formatting.

Summary

Display Name	Highlight Test Expression
Property group	Highlight
ROM Type	Expression
Expression type	any
JavaScript type	any
Default value	The value of the data item itself.
Inherited	Yes

Settable at runtime	No
Availability	First release

Description

Each highlight rule has a condition and a set of formatting options to apply if the rule is true. BIRT evaluates each rule in term, and applies the first one that evaluates to true. As a result, the rules need not be mutually exclusive, and the order of the rules matters.

A highlight is based on a test value. The rules then match this value.

See Also

`highlightRules` Property

5.11.4 `highlightRules` Property

Conditional formatting rules.

Summary

Display Name	Highlight Rules
Property group	Highlight
ROM Type	List of Highlight Rule Structure
JavaScript Type	Array of `PropertyStructure` object
Default value	None
Inherited	Yes
Settable at runtime	No
Availability	First release

Description

Each highlight rule has a condition and a set of formatting options to apply if the rule is true. BIRT evaluates each rule in term, and applies the first one that evaluates to true. As a result, the rules need not be mutually exclusive, and the order of the rules matters.

See Also

`highlightTestExpr` Property

5.11.5 `mapExpr` Property

An expression used to map internal values to display values.

Summary

Display Name	Map Expression
Property group	Map
ROM Type	Expression
Expression type	`any`

JavaScript Type	any
Default value	The value of the data item itself.
Inherited	Yes
Settable at runtime	No
Availability	First release

Description

The map rules convert internal values into display values. For example, a null value can be converted to the string “null” for display. Or, internal “M” and “F” codes can be converted to “Male” and “Female” for display.

See Also

`mapRules` Property

5.11.6 `mapRules` Property

Summary

Display Name	Map Rules
Property group	Map
ROM Type	List of Map Rule Structure
JavaScript Type	Array of `PropertyStructure` object
Default value	None
Inherited	Yes
Settable at runtime	No
Availability	First release

Description

A map rule transforms a value in the input into a different value for display. It works best for fields with a limited set of values, such as converting “Y” to “Yes” and “N” to “No”. Mappings with many rules are better handled in the data access layer.

Another common use of mapping is to convert a null value into a display value, such as “No Data.”

If a particular item has both highlight and map rules, the highlighting rules apply to the original value. Formatting, however, is applied to the mapped value. The mapped value determines the type of formatting rule to apply. For example, if a numeric value of 1 is mapped to “On Order”, then the string formatting rules will apply to the item.

See Also

`mapExpr` Property

6. Property Structures

6.1 Highlight Rule Structure

Represents one highlight rule: a value and corresponding format style.

Summary

Availability: First release.

Properties

`operator`

The operator for a simple condition

`value1`

The value for simple conditions with the operators: <, <=, =, <>, >=, >, between, not between, like

`value2`

The value for simple conditions with the operators: between, not between

`style`

The name of the style to apply when the condition is met.

Description

A highlight rule gives a set of conditional style properties. The style is applied only if the rule “fires.”

6.1.1 `operator` Property

Summary

Display Name: Operator

ROM Type: Choice

JavaScript Type: String

Default value: =

Settable at runtime: No

Availability: First release

Choices

Display Name	Internal Name	Description
<, <=, =, <>, >=, >,		
is null, is not null,		
between, not between		
true, false		
like		

Display Name	Internal Name	Description
any		

6.1.2 `value1` Property

The value for simple conditions with the operators: <, <=, =, <>, >=, >, between, not between, like

Summary

Display Name: Value 1

ROM Type: Expression

Expression Type: any

JavaScript Type: any

Required.

Settable at runtime: No

Availability: First release

Description

The value for simple conditions with the operators: <, <=, =, <>, >=, >, between, not between, like

6.1.3 `value2` Property

The value for simple conditions with the operators: between, not between

Summary

Display Name: Value 2

ROM Type: Expression

Expression Type: any

JavaScript Type: any

Required.

Settable at runtime: No

Availability: First release

Description

The value for simple conditions with the operators: between, not between

6.2 Map Rule Structure

Represents one map rule: a value and corresponding translated value.

Summary

Availability: First release.

Properties

`operator`

The operator for a simple condition

`value1`

The value for simple conditions with the operators: <, <=, =, <>, >=, >, between, not between, like

`value2`

The value for simple conditions with the operators: between, not between

`style`

The name of the style to apply when the condition is met.

`display`

The value to display when the map rule “triggers”.

Description

A map rule is part of the map style property. The map property defines a test value and a set of map rules. Each map rule contains a condition that matches the test value, and a value to display when the rule “triggers.” For example, a set of map rules for a “gender” database column may map the values null, M, F into the display values “No data”, “Male” and “Female”.

6.2.1 `operator` Property

Summary

Display Name: Operator

ROM Type: Choice

JavaScript Type: String

Default value: =

Settable at runtime: No

Availability: First release

Choices

Display Name	Internal Name	Description
<, <=, =, <>, >=, >,		
is null, is not null,		
between, not between		
true, false		
like		
any		

6.2.2 `value1` Property

The value for simple conditions with the operators: <, <=, =, <>, >=, >, between, not between, like

Summary

Display Name: Value 1

ROM Type: Expression

Expression Type: any

JavaScript Type: any

Required.

Settable at runtime: No

Availability: First release

Description

The value for simple conditions with the operators: <, <=, =, <>, >=, >, between, not between, like

6.2.3 `value2` Property

The value for simple conditions with the operators: between, not between

Summary

Display Name: Value 2

ROM Type: Expression

Expression Type: any

JavaScript Type: any

Required.

Settable at runtime: No

Availability: First release

Description

The value for simple conditions with the operators: between, not between

6.2.4 `display` Property

The value to display when the map rule “triggers.”

Summary

Display Name: Display Value

ROM Type: String

JavaScript Type: String

Default value: None

Settable at runtime: No

Availability: First release

Description

The display value appears when the map rule triggers. The value is a string and is not further formatted.