BIRT Report Object Model – Layout Model

Functional Specification

Draft 4: April 26, 2005

Abstract

Describes the layout model for the Java Reporting Platform Report Object Model.

Document Revisions

Version	Date	Description of Changes
Draft 4	04/26/2005	Replaced Image size with width/height
Draft 3	02/25/2005	Fixed typo in section 6.1.1.
Draft 2	02/15/2005	Removed redundant alignment from Column element. Added missing style and styleProp for Column element.
Draft 1	11/29/2004	First BIRT release.

Contents

1. Introduction

This specification is part of the collection of Report Object Model specs. It describes the structural elements that make up a report including sections, grids, and free-form. It also discusses the layout rules used when rendering BIRT reports.

The details of layout follow the CSS 1 (http://www.w3.org/TR/CSS1) and 2.1 (http://www.w3.org/TR/CSS21/) specifications unless otherwise noted. CSS 1 provides a good overview of CSS capabilities; CSS 2.1 provides detailed descriptions of the expected layout algorithms. Those two specifications are considered included in this specification by reference.

2. Sections

Reports are documents. Documents virtually always consist of a sequence of content. For example, a book consists of front matter, a table of contents, the body, and back matter. A report may consist of a title, a list, some charts and some text. We term each of these items a *section.*

Of course, any one section of content may have an arbitrary layout. A page in a textbook may consist of text, charts, sidebars and other content. Similarly, a dashboard-style report may have a block of content that shows charts side-by-side with a short table of data. However, in the large, reports are composed of sequences of content, some of which might be highly structured such as the dashboard.

Sections model the sequential nature of report documents. The user creates the overall structure of a report by defining a series of sections. Each section represents a block of report content that has a common structure.

Sections have two main parts: the properties of the section itself, and the section content. Section properties include things such as page control (page break before or after, the master page on which the section should start, whether to keep the section together on a page, etc.)

Because sections are bands within the document, their width is defined to be the width of the report's content area. That is, sections themselves don't specify a width; they automatically are sized to fit on the page. The height of a section, however, is defined by the section's content as explained below.

Section content fills the entire section. If the content is narrower than the section itself, then the user can elect to place the content at the left, center or right of the section.

The following can act as a top-level section within a report:

- List: a grouped list
- Table
- Text
- Chart
- Matrix
- Grid: a table-oriented collection of report items
- Free-form: an arbitrary collection of other report items
- Table of Contents
- Contents of another design file

(Not all of these are available in the first release. See the specific sections for each for details. In particular, Free-form, Matrix, table-of-contents and included reports are not in the first release. Charts are defined as extensions and are not part of the ROM specs.)

2.1 Conditional Sections

The developer can design a report to have conditional content. For example, sections can be included or omitted depending on report parameters that the user supplies. Or, sections may be excluded based on database values.

Every section has a hide expression that says when to hide the section. A hidden section simply does not appear in the resulting report document. Sections cannot be hidden based on the output type (only individual report items can be hidden in this way.)

2.2 Special Properties of Sections

Report items have a size and position. However, when used as a section, the size and position are ignored. (Actually, the height is used for selected items as noted below.)

2.3 Section Height

The height of a section depends on the content. The following table explains how section height is computed for each type of item.

Section Type	Section Height
List	Defined as the sum of the various sections that make up the list.
Table	Defined as the sum of the various rows that make up the table.
Container	Fixed at design time. May vary at run-time based on the sizes of the contained items.
Grid	Defined as the sum of the rows that make up the grid.
Text	Defined as the sum of the line heights, line spacing, paragraph spacing, and so on.
Matrix	Defined as the sum of the rows that make up the matrix.
Chart	Fixed at design time.

Section Type	Section Height
TOC	Defined as the sum of the lines of text that make up the TOC.
Include	Defined by the included report.

Subsections resize independently of one another. The developer defines the content that moves in response to another item resizing. Most often, the items that move are below the item in question. In this case, the developer simply puts those items into a separate subsection.

2.4 Section Properties

Sections are the primary unit of report structure and provide a number of properties to assist the developer to create the desired output.

2.4.1 Pagination

A section is kept all on the same page by default. However, the developer can allow a section to break across pages even if the section is smaller than a page. This helps eliminate large amounts of white space at the bottom of pages.

Sections can also be marked as “keep with next.” Such a section will stay on the same page as the next section unless the combined size is too large to appear on a page.

2.4.2 Keeping Sections Together

Developers often want the entire section to be kept together on a page even if it is defined using a series of subsections. The developer does this by setting properties that keep an entire section together, or that keep any one subsection with the next subsection.

2.4.3 Ignoring Unused Sections

Sections defined within a list always exist in the design, but developers will seldom need to put content into every section. BIRT simply ignores empty sections and subsections. Of course, there may be times when the developer has intentionally created an empty section or subsection, perhaps to establish extra white-space in the report. In this case, the developer simply sets a property that says to always print the section or subsection, even when it is empty.

2.4.4 Format-Specific Subsections

Developers sometimes include report content that is useful in only some of the possible report formats. Most often, a report will include an interactive control or HTML-based element that works only on the web, but not in print or in PDF. The developer can mark each subsection with the format in which it should appear.

3. Creating Dashboards with Free-form Elements and Grids

Free-form elements and grids provide a way to create dashboards and similar report layouts that include a variety of top-level items.

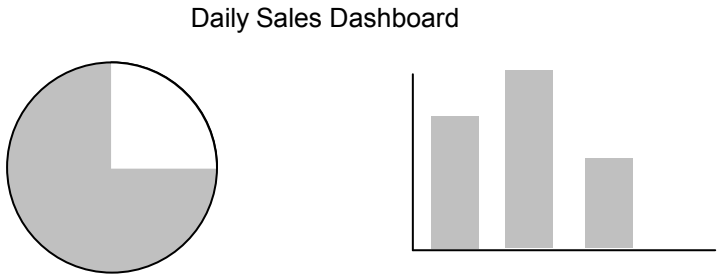
A dashboard displays one or more independent data values into one easy-to-use page. Dashboards often using visualize Key Performance Indicators (KPIs) in graphic forms such as gauges, stop- lights, thermometers, etc. For example, a development dashboard may show incoming bug rate, bug fix rate, bug inventory count, etc. Each value may have its own query, but the query returns a single value, not a list of values. Other dashboard elements are charts and small matrices. The queries for such items return a list of values, but the values are displayed in a single display element.

Typically, each element in a dashboard is associated with its own query. For example, a dashboard might show the order backlog, support case backlog, and sales as a percent of quota. Each comes from a distinct query and perhaps a distinct database.

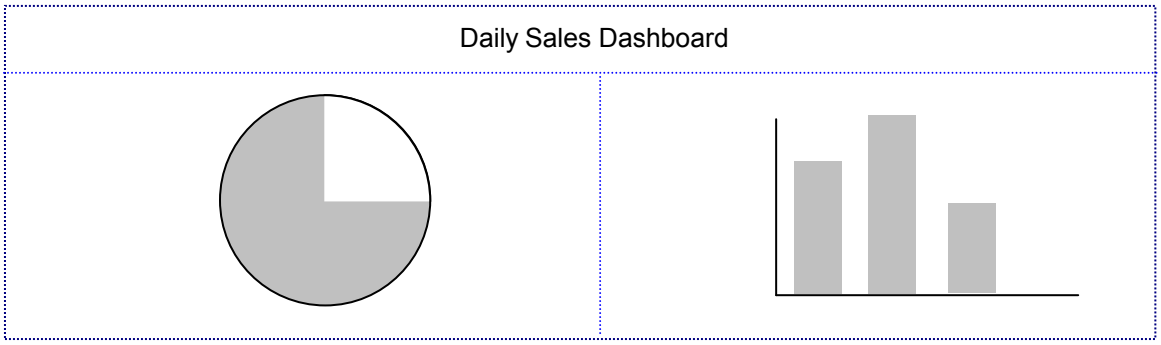
Some dashboards display data that changes infrequently such as metrics for a direct sales force. Others display dynamic data such as the statistics within a call center. Dynamic data automatically is useful only when fresh, so dashboards require the ability to refresh periodically. The developer determines the refresh rate that is best for his application.

It is very common for a user to want to drill down from a dashboard item to the underlying data in any format: BIRT report, third-party report, etc.

One way to create a dashboard is to use a free-form element to place contents at arbitrary locations.



However, many dashboards have a structure defined by a grid (table). The grid section provides a much easier way to create such a dashboard by automatically placing items so that they line up properly.



Note that a grid in this context is different from the table element. A grid is simply a way to arrange a set of items within one section. A table is a way to create a list driven by a

query in which the various sections of the list share a set of column definitions. Both are based on a tabular layout, but the grid is static while the table is dynamic.

3.1 Variable-Sized Content

Many reports include content that can grow or shrink based on a number of factors. The grid layout automatically handles this issue by resizing rows as needed. But, the developer must decide how to manage sizing when using the free-form layout. (The purpose of the free-form layout is to give the developer complete control.) The most common choice is simply to list items down the page with a fixed spacing between them.

4. Automatic Size and Position

BIRT automatically computes the size and position of many types of report items. For example, BIRT will automatically format the contents of a text item, will automatically place the contents of a grid, and will automatically size a container as needed if the contents of the container grow. This section discusses the sizing rules.

4.1 Text Item Layout

Several items determine the size of a text item.

- Data content
- Font settings
- Rendered font size

4.1.1 Data Content

A text item contains three kinds of text:

- Static text defined in the text item itself.
- Externalized text that resides in a message catalog.
- Values computed at run time and inserted into the text stream.

The first step is to “resolve” the text for the second and third cases to determine the actual text to use in the output.

The size of label text is determined by what appears in the message catalog. A label in one language will often be much larger (or smaller) than that in another language.

The size of a computed value is determined by the size of text from a data source, in a parameter, etc.

4.1.2 Font Size Effects

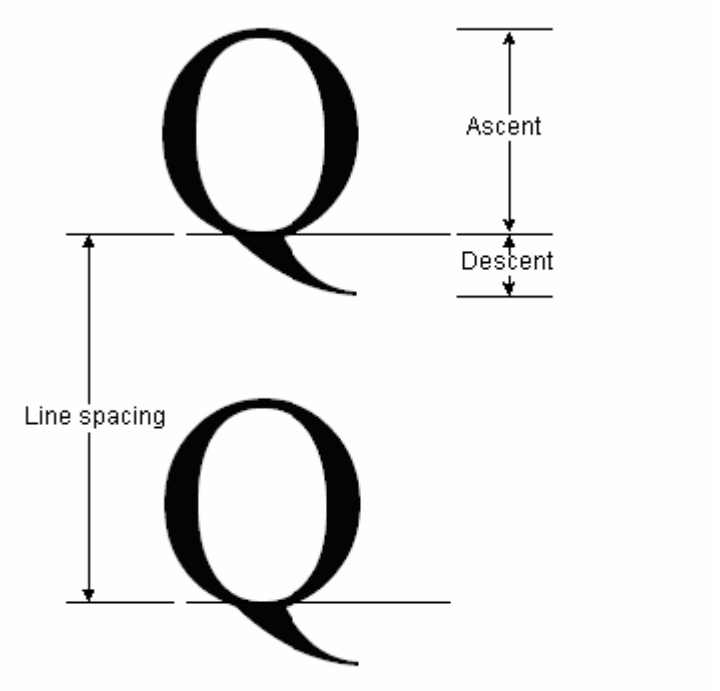
The actual printed size of a text-based item further depends on additional factors. The most important is the *rendered size* of the text in the given font and zoom factor. Fonts are very complex; and computing the size of text when printed in a given font depends on a number of factors.

The first factor is that fonts cannot smoothly scale on all devices. Computer monitors are made up of pixels. A rendered font must fit into an integer number of pixels. That is, if a 10-point font might require 12.49 pixels, then the OS might round the font down to

12 pixels. If on the same system, a 9.5-point font might require 11.9 pixels. The OS may round this font to 12 pixels. The actual number of pixels further depends on factors such as the pitch density (in dots-per-inch or dpi) of a given monitor, and the zoom setting within the software. Many modern printers are better able to render fonts correctly because they have a much higher resolution.

The width of a string rendered in a given font depends primarily on the width of each character. Most fonts are variable-width: the width of a character such as “i” is much smaller than that of “W”. Hence, the actual printed with of a string of characters depends on the width of each character in the given font as rendered on the given device at the given zoom factor.

The height of a line of text in a given font is also complex. While it seems that the height of two different 10-point fonts should be the same, actual height depends on some key font metrics: the descent, ascent and line spacing metrics:[1]



The line between the ascent and descent parts of the font is called the baseline. Most word-processing systems (Word, PowerPoint, etc.) align text at the baseline. If two adjacent characters have different fonts, then the system aligns the characters so that their baselines are all along the same imaginary line.

The result of the font complexity is that the width of a given string of text depends on the actual characters in the text, the font(s) applied to the text, display device and the zoom factor, etc. Given all these factors, it is very hard to predict ahead of time exactly how wide the text might appear on any given output format.

Most existing report tools require that the user guess at design time the size of a rectangle that will hold the text for all possible values of the text on all possible output formats. Experience suggests that doing so is very difficult in practice. Developers

[1] This picture is taken from the Platform SDK documentation provided by Microsoft.

become extremely frustrated that text that looks fine on their screen is truncated on the boss's screen.

4.1.3 Baseline Alignment

Printed text is often aligned along the font baseline. The font baseline is an imaginary line that runs along the base of a character. Descenders (as in the letters "g" or "q") drop below the baseline. Each font has a different baseline position. If the line contains characters in different languages, in different fonts, or different sizes, the baseline ensures that they all align.

4.1.4 Computing Text Lines

BIRT constructs the text item as a series of lines. Text appears within the lines. Lines are sized to hold the largest item in the line. Text within the line is baseline aligned. Non-text items are positioned at the top, middle or bottom of the line. For example:

First Label:	first value	Second Label:	second value
Third Label:	third value	Fourth Label:	fourth value

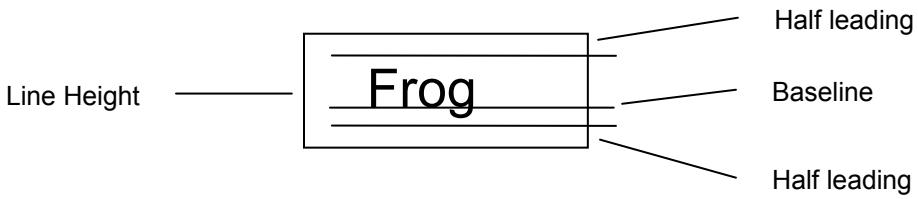
Line-based positioning within a text item has several key advantages:

- Eliminates the tedium of getting several text items to align properly, especially when the items use different fonts.
- BIRT automatically resizes lines based on the actual size of fonts available on a given device or output format. The actual rendered size of a font depends on many factors: the specific font version installed on the device, the zoom factor, and so on. By delegating these details to BIRT, the developer can simply specify what he wants and let BIRT work out the details of dealing with the complexities of fonts.
- Helps BIRT produce very good PDF, RTF and DHTML output since these formats are text and line oriented. The resulting output can easily be edited or copied and pasted into other documents.
- Helps BIRT produce good Excel output. Items on a section line are placed into the same Excel spreadsheet row.

BIRT determines line sizes as follows:

- Lines with text are sized based on the size of the largest font used within that line.
- Lines with images are sized to the height of the image, or the height of the text as computed above.

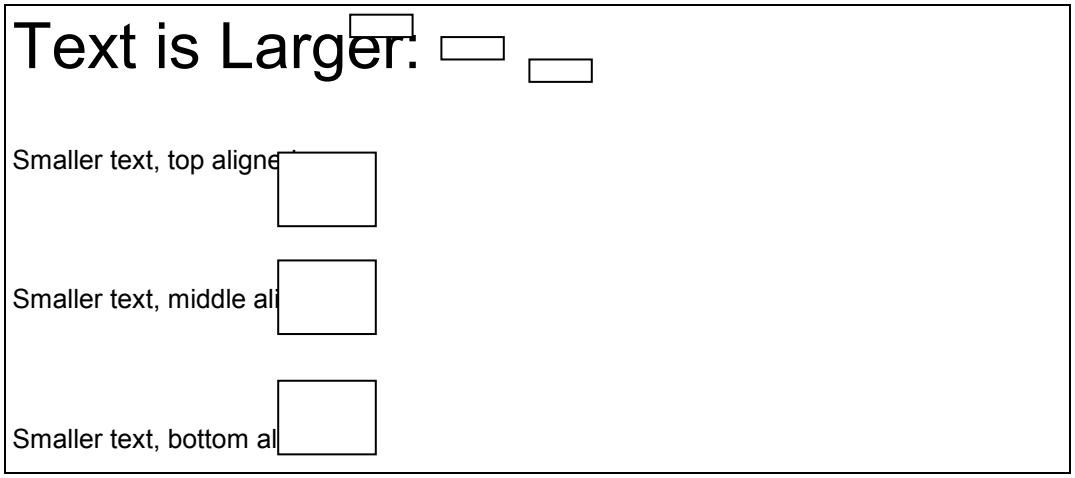
Every line also includes a top and bottom margin made up of the leading of the nominal font for the text item:



BIRT positions report items within the line as follows:

- Text items are always baseline aligned. The font baseline is the imaginary line that runs under the base of characters without descenders. Fonts provide a complex system for determining overall line height: a system that includes measurements for the descender, the part of the character above the descender, the leading around the characters, etc. However, the key characteristic for line positioning is the baseline. The baseline ensures that different fonts, and different sizes, all align properly.
- Non-text items do not have a baseline. Instead, they are aligned at the top, middle or bottom of the line. They align in the middle by default. The actual alignment depends on the relative size of the non-text item to the line size.
- If the overall line is larger than the non-text item, then the item is positioned within the line. If, however, the non-text item is larger than the text items, then the non-text items' positioning applies to the text items. If the line contains more than one non-text item, then the text positioning rule is set by the first one on the line that is larger than the tallest text item.

For example:



In the first line, the text is larger than the rectangle (a kind of non-text item). Three rectangles appear on the line aligned top, middle and bottom.

4.1.5 Managing Font Differences

Yet another complication to report layout arises from the differences in fonts and font rendering. Each font has distinct sizes for each character. Even two instances of the same font may differ in sizes if they represent different implementations or versions. Further, a font will render differently depending on the output device. The pixels on a

video screen are large relative to a typical displayed character, so the rendering software tends to round a font from its ideal size to an even multiple of screen pixels. For example, a font that should ideally take one inch might take 0.95 inches on some screens and 1.05 inches on others. Printed fonts tend to be closer to the ideal size because of the greater resolution of laser printers.

Hence, even if the developer takes the time in e.RD-Pro or Crystal to adjust all the report fields sizes so they look good on his monitor, the result may be quite different on other monitors and when printed. Further, the result will differ across different printers depending on the printer resolution, loaded fonts, and so on. Further, the report will also differ when displayed in a Web browser depending on the available fonts, browser features and other factors. These problems have produced a persistent stream of complaints in e.RD-Pro.

There are two broad ways to solve the problem. First, we can hold the report layout fixed and adjust various factors to achieve the desired layout. Second, we can vary the layout to account for the capabilities and limitations of the target format.

Adobe Acrobat is the best-known example of the first approach. Acrobat attempts to reproduce a page layout faithfully on a wide variety of screens and printers. Acrobat will adjust font sizes, inter-character spacing and other factors to squeeze or expand text to fit. The result is that two people will see the same page layout, but one may see some of the text somewhat compressed.

Microsoft Word is the best-known example of the second approach. Open a Word document. Set your printer to a laser printer. You'll see a certain page layout. Switch to an ink-jet printer. Word will revise the page layout based on the different font metrics for each printer. The result here is that two people with different hardware will see a different page layout, including possibly different page counts.

BIRT will adopt the same approach as Microsoft Word; it will size the report content to based on the characteristics of each output device.

4.2 Container Positioning

Containers hold items at arbitrary positions. Any one of the items can grow or shrink. BIRT provides a number of rules for adjusting the overall container based on the size of contained items.

4.2.1 Nominal and Actual Size

Report items are given a nominal size in the design. For example, an image item may be sized at 1" × 1", but the actual size will depend on the image, and the image itself may be ¾" × 1 ½". BIRT takes the nominal size as a suggestion then adjusts the size based on the content. The developer can control this behavior with the Resize property of a report item.

Some items that can grow vertically include:

- Text item
- List
- Table
- Grid

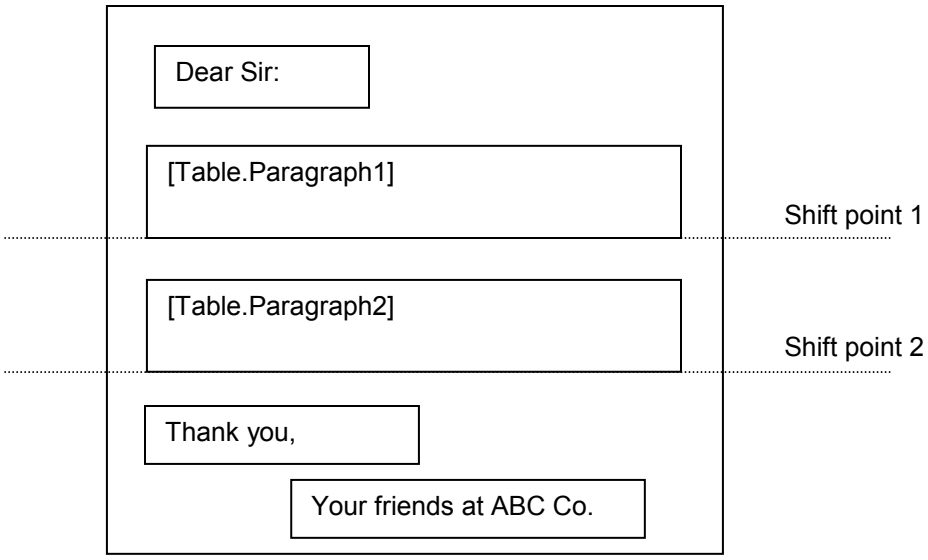
Some items that can grow horizontally include:

- Image
- Matrix
- Grid

4.2.2 Shift Points

If items can shift, the next issue to decide is what to do with other items within the container. BIRT computes this by defining a set of shift-points within the container. A shift-point occurs along the trailing edge of each item. The trailing edges are the bottom (for elements that can grow vertically) or the right edge (for items that can grow horizontally.)

Consider the following container in which the text boxes can grow but the labels cannot:



4.2.3 Repositioning Items

Shift points give a way to reposition items. Conceptually, each item is positioned relative to the nearest shift point above, or to the left, of the top corner of the item. The following table shows the relative positions of each item in the above illustration:

Dir Sir:	Shift point 0 (Upper left corner of container)
Paragraph 1	Shift point 0
Paragraph 2	Shift point 1
Thank you	Shift point 2
Your friends at ABC Co.	Shift point 2

BIRT uses a four-step process to reposition items:

1. Compute the actual size of each item.
2. Compare the new trailing edge of the item with the original size. The difference is the delta.
3. Apply the delta to the shift point. A positive delta moves the shift point down. A negative delta move the shift point up.
4. Compute new absolute positions of items by keeping the same relative position to the shift point, but changing the absolute position based on the movement of the shift point.

4.3 Grid, Table and Matrix Size

The size of a grid depends on the size of each cell. Each cell is sized depending on its content: a text layout, a list, a container, and so on.

The height of a grid row depends on the height of the tallest cell in that row. The width of a grid column depends on the width of the widest cell in that column.

The overall size of the grid is simply the sum of the rows (height) and columns (width.)

A similar rule applies to tables and matrices. For tables and matrices, the set of rows is driven by data. For a matrix, the set of columns is also driven by data.

4.4 List Size

The height of a list is determined by the sum of all sections within the list. The size of each section is computed as described above. The width of the list is the width of the widest section within the list.

5. Block vs. In-Line Sections

HTML and CSS define two concepts that determine how content appears on a page:

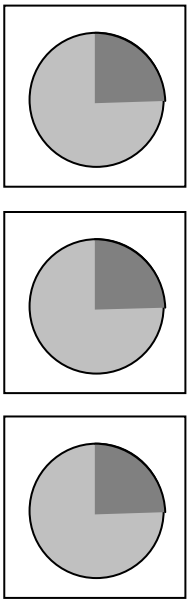
- “in-line” content is placed on the same line as other in-line content. The words in this sentence are in-line content.
- “block” content causes a line break between successive content items. The paragraphs in this document are block content.

ROM defines all sections as block content by default. That is, they appear down the page with a line break between each block. However, the user can change this behavior to put multiple sections on the same line.

The in-line vs. block setting for a section is set by the “Display” style property. The display property is ignored if the section cannot appear in-line.[2]

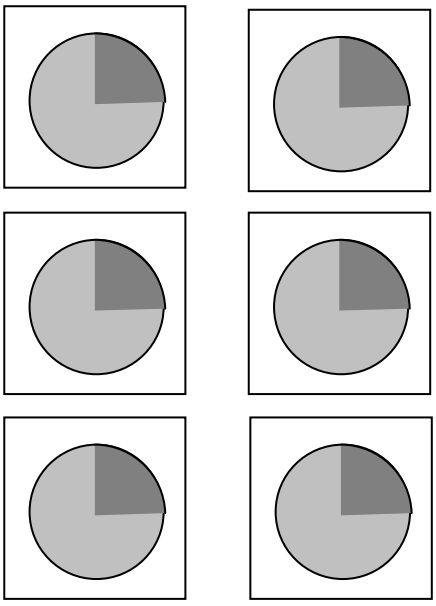
For example, a report may want to display a series of six charts, each of which are three inches square. The page is 6.5” wide. By default, the charts will appear like this:

[2] The in-line vs. block setting replaces the Tile element that appeared in previous drafts of this specification. We believe that the in-line option along with the CSS margin and line-alignment properties, can accomplish all that the Tile element did, but with greater simplicity.



...

However, this is wasteful of space. Two charts can fit onto each line. To do this, we set the charts be in-line content:



Margins and padding controls the spacing between charts. In the above, each chart has margins on the right and the left to provide space between them.

Because the charts are on lines, the line alignment can be used to move the charts to the left, center or right of the page.

5.1 Table of In-line Elements

The following table shows which sections are capable of acting as in-line content.

Element	In-line?
List	No. But, subsections within the list can be marked as in-line.
Table	Yes.
Chart	Yes.
Matrix	??
Text	Yes. The following section appears on the same line as the last line of the text.
Free-form	??
Dashboard item	Yes.
Include.	No.
TOC	No.

If an element is marked as block content, then an implicit new line occurs both before and after that section. For a section to be on the same line as another section, both sections must be in-line. In the example above, if the six charts are followed by a block-content dashboard item, then the dashboard item will cause a break and will appear on a line by itself.

6. Layout Elements

This section explains two of the primary layout elements: grid and free-form. In addition, the following elements, described elsewhere, also help define report layout:

- Text, for creating blocks of text, centered headings, paragraphs, etc.
- Tables, for tabular layout of data retrieved from a data set.
- List, for producing a set of arbitrary content based on a data set.

6.1 Free-form Container Item

Summary

Base element: Report Item

XML Element Name: `free-form`

Predefined Style Name: `free-form`

Availability: After the first release

Inherited Properties

`dataSet`

Allows the container to display items from a data set. The container can reference columns in the data set. The container works only with the first (or only) row in the data set. If omitted, the item works with the data set defined by a parent element.

Contents

`reportItems`

The report items that appear inside the free-form container.

Description

The simplest form of report layout is free-form. Free-form layout is best for layouts that do not have a regular structure, such as a header for a tabular report. This layout option needs little explanation.

A container item holds a collection of other report items. Every item in the container is positioned at an (x, y) location relative to the top left corner of the container. Free-form layout is very much like the layout in e.RD-Pro: elements can be positioned anywhere. A drawback of free-form layout is that items are not automatically repositioned depending on the size of other elements. For example, if a large block of text is followed by another data item, the data item may be hidden if the block of text becomes too large.

Container Size

The size of a section is determined at presentation time based on the size of each line within the section. A section can be come too large to fit on a single page. In this case, the section is automatically split across pages. The split occurs at a line break. If a section line contains multiple text lines, then the break may be between text lines.

Z-Order

Report items have a z-order that determines what happens when one item overlays another. The rule is that report items paint in the order they are defined in the section. To have one item overlay another, the developer simply ensures that the bottom item appears in the item list before the top item.

6.1.1 `reportItems` Slot

Holds the report items that appear within the free-form container.

Summary

Cardinality: Multiple

XML Element name: `report-items`

Availability: After the first release

Description

Supported report items include:

Label	Rectangle	List
Data	Chart	Table
Text	Matrix	Include
Image	Grid	Free-form

Toggle Image	Browser Control	Extended Item
Line	Dashboard Item	

7. Simple Elements

This section describes a number of decorative elements used as part of the report layout.

ROM defines many sophisticated elements described in other specifications. It also defines a set of simple elements that display a single bit of data or text. Simple elements include:

- Label: to display static text. The text can be localized.
- Data: to display one data value.
- Line: to display a horizontal or vertical line.
- Rectangle: to display a rectangle
- Image: to display an image from the web, from a design, from a database, etc.
- Toggle image: to control the expansion state of report that has expand/collapse (drill-down) capabilities.
- Table of Contents item: to insert a TOC into a report, usually for print.

7.1 Line Element

Lines to add graphical interest to a report.

Summary

Base element: Report Item

Availability: After the first release

XML Element Name: `line`

Inherited Properties

`dataSet`

The data set is ignored for a line item.

`toc`

While the line can appear in the TOC, it is generally not very useful to do so.

Properties

`orientation`

The orientation of the line: Horizontal (default) or Vertical.

`thickness`

The width of the line.

`lineStyle`

The line pattern.

Description

Customers often use lines and rectangles to add graphical interest to a report. For example, the customer may draw a box around certain fields to group them. Or, the customer may use alternating background colors to improve report legibility.

Most reports are intended for viewing on the web. Web browsers generally support only horizontal and vertical, but not diagonal lines. Diagonal lines require an image which imposes extra cost and is difficult to correctly format. So, JRP ERD limits support to horizontal and vertical lines, but not diagonal lines.

The user can set the line size, line color and line pattern using the item's style.

A line item describes a line along the edge of a rectangle. If orientation is vertical, the line is along the left edge of the bounding box. If the line is horizontal, the line is along the top edge of the bounding box. That is, the X and Y properties give the line origin. The Width property gives the length of horizontal lines, the Height property gives the width of vertical lines.

Open Issue: *Need to verify that the Dimension is the right type for thickness, and determine the correct set of line styles.*

7.1.1 `orientation` Property

The orientation of the line: Horizontal (default) or Vertical.

Summary

Display Name: Orientation

ROM Type: Choice

JavaScript Type: String

Default value: Horizontal

Inherited: Yes

Settable at runtime: No

Availability: After the first release

Choices

Display Name	Internal Name	Description
Horizontal	`horiz`	
Vertical	`vertical`	

Description

The orientation of the line: Horizontal (default) or Vertical.

7.1.2 `thickness` Property

The width of the line.

Summary

Display Name: Thickness

ROM Type: Dimension

JavaScript Type: String

Default value: 1 pt

Inherited: Yes

Settable at runtime: No

Availability: After the first release

Description

The width of the line.

7.1.3 `lineStype` Property

The line pattern.

Summary

Display Name: Line Style

ROM Type: Choice

JavaScript Type: String

Default value: Solid

Inherited: Yes

Settable at runtime: No

Availability: After the first release

Description

The line pattern.

7.2 Rectangle Element

Summary

Base element: Report Item

Availability: After the first release

XML Element Name: `rectangle`

Inherited Properties

`dataSet`

The data set is ignored for a rectangle item.

`toc`

While the rectangle can appear in the TOC, it is generally not very useful to do so.

Description

The Rectangle element describes a simple rectangle. The user can set the line size, line color, line pattern and fill color using style properties. The rectangle element adds no properties beyond those inherited from the base graphic item.

7.3 Table of Contents Element

Summary

Base element: Report Item

Availability: After the first release

XML Element Name: `toc`

Description

The table-of-contents control allows the user to insert the report TOC into the report itself much like a book or printed document. Look at the similar feature in Microsoft Word as a reference. Requirements include:

- Set the font of the entries. Allow different sizes for each TOC level.
- Specify indentation, dot leaders, etc.
- Set the maximum TOC depth
- When viewed, provide hyperlinks on each TOC field.
- When viewed, provide ability to expand or collapse the TOC dynamically.
- Provide option to omit the TOC when viewing (because the user can use the viewer's own TOC.)
- When printed, provide the page number for the entry. (Pagination is dynamic when viewing so page numbers are not required, and may be quite difficult, when viewing.)
- Choice of a variety of layout styles.
- A table of contents renders as separate stream in PDF.

Note: *This element is a low priority. It is useful, but will be used much less frequently than the other elements.*

7.3.1 Table of Contents Item

The table of contents in a book helps the reader see an overview of the material, and to locate sections of interest quickly. Similarly, the table of contents in a report shows the overall report structure and allows the user to locate data of interest. TOC was a differentiator in e.RD-Pro when first introduced, but has since been adopted by other products such as Crystal.

The TOC for a book shows the book's structure: chapters, sections, subsections, etc. Similarly, the TOC or a report shows the report's structure. The TOC generally includes report groups. If a report has a sequence of top-level reports, the TOC will likely include an entry for each report.