
Elixir Report Raw Model

Elixir Technology Pte Ltd

Copyright © 2006-2008 Elixir Technology Pte Ltd

Table of Contents

Introduction	2
Shorthand Notation	2
Attributes	2
Collections	2
Elements	3
Barcode	3
BarcodeBasic	4
BasicRawElementHolder	4
Box	4
CallbackElement	4
Cell	5
CellRender	5
Chart	5
CheckBox	6
Chunk	6
ControlSource	6
CubeColumn	7
CubeHeader	7
CubeHierarchy	8
CubeLevel	8
CubeMeasure	8
CubeRow	10
CubeTable	10
DataSource	11
Detail	11
Field	11
Format	12
Grid	12
Group	13
GroupFooter	13
GroupHeader	13
HBox	14
Holder	14
Image	14
Line	14
Metadata	15
PageBreak	15
PageFooter	15
PageHeader	16
PageSetup	16
Parameter	16
RawElement	17
RawElementHolder	17
RawModelElement	18
RawReport	18
RectHolder	19
Rectangle	19

RenderDetails	19
RTF	20
RulerMark	20
SVG	20
Script	20
Section	20
SectionFooter	21
SectionHeader	21
SectionInvocation	22
Security	22
ShapeGroup	22
Style	22
StyleItem	23
StyledElement	23
SubReport	23
TOCElementHolder	23
Table	23
TableBody	24
TableFooter	24
TableHeader	24
TableSection	24
TextElement	25
VBox	25

Introduction

This document describes the internal object model of Elixir Report. The object model consists of a number of classes that are all in the `com.elixirtech.report2.raw.model` package. The information is generated directly from the tool. While we endeavour to ensure this API will remain unchanged, it is possible that changes will be needed in future releases to correct errors or to add significant functionality.

All location and dimension values are in twips, where we assume 1 twip = 1/20 of a pixel and there are 72 pixels per inch (giving 1440 twips per inch). Font sizes are also measured in twips.

Shorthand Notation

Attributes

Where an object is shown to contain another object, for example every `Style` has a `String` called `Name`, this implies there are methods called `getName()` and `setName(String name)`. In JavaScript it is also allowable to access the property directly as `style.Name`, which is internally mapped into calls to the `get` or `set` method, depending on the context. The only exception to this pattern is `Boolean` values, where an attribute of type `Boolean` called `Enabled` is accessed by `isEnabled()` and `setEnabled(Boolean b)`. This is consistent with Java bean naming conventions.

Collections

Many objects hold collections of other objects. In this object model these are `Lists` and `Maps`. Each `List` and `Map` is exposed through an API described here.

List collections are documented as in this example:

```
List: Item : List<CubeLevel>
```

This shorthand defines the following functions:

- `int getItemCount()`
- `void addItem(CubeLevel item)`
- `void addItem(int idx, CubeLevel item)`
- `void addAllItems(Collection collection)`
- `int indexOfItem(CubeLevel item)`
- `void removeItem(CubeLevel item)`
- `void removeAllItems(Collection collection)`
- `void removeAllItems()`
- `CubeLevel getItem(int idx)`
- `java.util.Iterator getItemIterator()`

Each shorthand can be trivially expanded into these ten functions, varying the name of the contents (Item in this case) and the type of the contents (CubeLevel in this case).

Map collections are documented as in this example:

```
Map: RenderDetails : Map<String,RenderDetails> key is value.getMimeType()
```

This shorthand defines the following functions:

- `void addRenderDetails(RenderDetails renderDetails)`
- `void removeRenderDetails(RenderDetails renderDetails)`
- `RenderDetails getRenderDetails(String mimeType)`
- `java.util.Iterator getRenderDetailsIterator()`
- `int getRenderDetailsCount()`

Each shorthand can be trivially expanded into these five functions, varying the name of the key type and value type (String and RenderDetails respectively in this case). The add method requires more explanation, the key is not passed in to the function, as with a typical `Map.put(key,value)` call, instead the key is derived by invoking a method on the value. In this example, the key is derived by calling the function `getMimeType()` on the `RenderDetails`, which returns a `String` to be used as the key. This is done inside the API, so you just have to add a `RenderDetails` and the key will be determined automatically.

Elements

Barcode

Barcode extends `RawElement` (view `RawElement`)

Represents a Barcode component. The attributes of the barcode are currently held in `BarcodeBasic` as there are plans to add alternate kinds of barcode in future.

Implicit Style Name: `barcode`

- `Category : String`

- Type : String
- ControlSource : ControlSource (view ControlSource)
- BarcodeBasic : BarcodeBasic (view BarcodeBasic)

BarcodeBasic

Holds the attributes for a one dimensional barcode component.

Implicit Style Name: type-barcode-basic

- Height : Integer
- BarWidth : Integer
- BarcodeAngle : Double
- CheckDigit : Boolean
- ShowText : Boolean

BasicRawElementHolder

BasicRawElementHolder extends RawModelElement (view RawModelElement)

An abstract element holder that allows iteration over the contents.

Implicit Style Name: basic-element-holder

- Fill : Boolean

List: RawElement : List<RawModelElement>

Box

Box extends RectHolder (view RectHolder)

An abstract element defining the capabilities for horizontal and vertical boxes. All boxes hold cells which are weighted so that the layout can be easily managed.

Implicit Style Name: box

- Name : String
- Visible : Boolean
- KeepTogether : Boolean

List: Cell : List<Cell>

CallbackElement

CallbackElement extends RawElement (view RawElement)

This component acts as an intermediary for external callback components. Two are provided as samples with Elixir Report: HTML and RTF. Additional components can be easily added by implementing the callback interface. Contact Elixir if you are interested in doing this, we may already have a renderer for the content you are interested in (eg. mathematic formulae in MathML etc.).

Implicit Style Name: callback

- ShowAsImage : Boolean
- ImageResolution : Integer
- Type : String
- ControlSource : ControlSource (view ControlSource)
- Document : com.elixirtech.org.jdom.Document

Map: Parameters : Map<String,Parameter>key is value.getKey()

Cell

Cell extends RectHolder (view RectHolder)

A cell is held inside an HBox or VBox to give a weight to the contents. If the weight of a cell is changed dynamically (eg. through scripts) then the layoutCells() function of the HBox or VBox should be called to ensure the cell positions are correctly updated.

Implicit Style Name: cell

- Weight : Integer

CellRender

A CellRender is used to control the rendering of cells in a cube table. A cube contains a list of CellRenderers and checks whether any of them are appropriate (using the When and Test attributes).

Implicit Style Name: cell-render

- Foreground : java.awt.Color
- Background : java.awt.Color
- IconName : String
- When : String
- Test : String

Chart

Chart extends RawElement (view RawElement)

Defines a chart component, which is held in a separate chart markup language (cml) structure. See the Elixir Report Chart Model for more details.

Implicit Style Name: chart

- Locale : Locale
- URL : String
- URLDescription : String
- URLTarget : String

- ImageResolution : Integer
- Preview : Boolean
- DataSource : String
- DataRange : String
- ChartURL : String
- Document : com.elixirtech.org.jdom.Document

CheckBox

CheckBox extends RawElement (view RawElement)

Represents a Check box component. The component allows any two images to be used to represent "On" and "Off" - they need not necessarily look like checkboxes - you could use a smiley face and a sad face...

Implicit Style Name: checkbox

- HorizontalAlign : String
- VerticalAlign : String
- OnImage : ControlSource (view ControlSource)
- OffImage : ControlSource (view ControlSource)
- ControlSource : ControlSource (view ControlSource)

Chunk

Chunk extends RawElementHolder (view RawElementHolder)

Represents a band in the report, such as a Header or a Detail.

Implicit Style Name: chunk

- URL : String
- URLDescription : String
- URLTarget : String
- OnLayout : Script (view Script)

List: RulerMark : List<RulerMark>

ControlSource

Provides data from datasources, operations, scripts, URLs or literals. If Type == "Field" then the attributes required are DataSource and Field. If Type == "Operation" then the attributes required are DataSource, Field, Operation, RunningValue and Range. If Type == "Script"; then the attributes required are Text (preserves space) and Operation. If Type == "URL" then the attributes required are URL and URLType. Finally, if Type == "Literal" then only Text (preserves space) is required.

Implicit Style Name: control-source

- Type : String
- DataSource : String
- Field : String
- Operation : String
- RunningValue : boolean
- Range : String
- URL : String
- URLType : String
- DisplayName : String

CubeColumn

CubeColumn extends CubeHeader (view CubeHeader)

Represents the horizontal dimension in a cube. All the characteristics are defined by CubeHeader.

Implicit Style Name: cube-column

CubeHeader

CubeHeader extends RawModelElement (view RawModelElement)

Represents a dimension in a cube. The presentation characteristics defined here are applied to all header labels within the dimension.

Implicit Style Name: cube-header

- TextAlign : String
- FontName : String
- FontSize : Integer
- FontBold : Boolean
- FontItalic : Boolean
- FontColor : String
- FontUnderline : Boolean
- FontStrikethrough : Boolean
- BackgroundTopLeft : String
- BackgroundBottomRight : String
- PaddingLeft : Integer
- PaddingRight : Integer
- PaddingTop : Integer

- PaddingBottom : Integer
- BorderStyle : String
- BorderWidth : Integer
- BorderColor : String
- BorderLeft : Boolean
- BorderRight : Boolean
- BorderTop : Boolean
- BorderBottom : Boolean
- Title : String

List: Item : List<CubeLevel>

CubeHierarchy

A CubeHierarchy is a named sequence of levels. It is intended to represent a strict hierarchy of contents, for example Country/State/City. It should not be used to collect together non-strict hierarchies like Year/Month, because the same Month occurs in multiple years, so it isn't strictly a hierarchy.

Implicit Style Name: cube-hierarchy

- Name : String

List: Level : List<CubeLevel>

CubeLevel

Represents a single level of grouping in a cube. A cube will consist of dimensions, each dimension consists of zero or more levels.

Implicit Style Name: cube-level

- Name : String
- Sort : String
- GroupOn : String
- GroupData : String
- DerivedField : String
- HideTotals : boolean

CubeMeasure

CubeMeasure extends RawModelElement (view RawModelElement)

Represents both the cube calculation to be performed on individual intersecting cube levels and the presentation of those calculation results through a sequence of CellRenderers.

Implicit Style Name: cube-measure

- Name : String
 - Function : String
 - Pattern : String
 - TextAlignment : String
 - IconAlignment : String
 - ShowText : Boolean
 - ShowIcon : Boolean
 - Locale : Locale
 - BackgroundColor : String
 - FontName : String
 - FontSize : Integer
 - FontBold : Boolean
 - FontItalic : Boolean
 - FontColor : String
 - FontUnderline : Boolean
 - FontStrikethrough : Boolean
 - PaddingLeft : Integer
 - PaddingRight : Integer
 - PaddingTop : Integer
 - PaddingBottom : Integer
 - BorderStyle : String
 - BorderWidth : Integer
 - BorderColor : String
 - BorderLeft : Boolean
 - BorderRight : Boolean
 - BorderTop : Boolean
 - BorderBottom : Boolean
 - WidthOverrideType : String
 - WidthOverride : Integer
 - Format : Format (view Format)
- List: CellRender : List<CellRender>

CubeRow

CubeRow extends CubeHeader (view CubeHeader)

Represents the vertical dimension in a cube. Most of the characteristics are defined by CubeHeader.

Implicit Style Name: cube-row

- WidthOverrideType : String
- WidthOverride : Integer

CubeTable

CubeTable extends RawElement (view RawElement)

CubeTable is the main component for viewing cubes, containing the list of rows, columns and measures needed. This object also provides default presentation attributes that will apply unless overridden by the individual presentation elements.

Implicit Style Name: cube-table

- CollapseRows : Boolean
- CollapseColumns : Boolean
- KeepRowTotals : Boolean
- KeepColumnTotals : Boolean
- ShowColumnHeaders : Boolean
- ShowRowHeaders : Boolean
- ShowColumnTotalsAfterDetails : Boolean
- ShowRowTotalsAfterDetails : Boolean
- KeepTogether : Boolean
- Cache : String
- DataRange : String
- WidthGrowable : Boolean
- WidthShrinkable : Boolean
- GrowHPageCount : Boolean
- MergeHeaderBorders : Boolean
- FontName : String
- FontSize : Integer
- FontBold : Boolean
- FontItalic : Boolean
- FontColor : String

- FontUnderline : Boolean
- FontStrikethrough : Boolean
- Column : CubeColumn (view CubeColumn)
- Row : CubeRow (view CubeRow)

List: Hierarchy : List<CubeHierarchy>

List: Measure : List<CubeMeasure>

DataSource

This object maps an RML datasource name, eg. "Master" to a physical datasource in the repository, eg. "/ElixirWorkspace/Sample.ds". It also holds the parameter values that the datasource needs, either literal values, like User="Jon", or propagation into report parameters, like User="\${User}".

Implicit Style Name: datasource

- Name : String
- DataSourceName : String
- Tabulate : boolean

Map: Parameter : Map<String,Parameter>key is value.getKey()

Detail

Detail extends TOCElementHolder (view TOCElementHolder)

Detail holds the components that form the detail chunk (band) of the report.

Implicit Style Name: detail

- KeepWithNext : Boolean

Field

Field extends TextElement (view TextElement)

Field is the basic text component in a report. It allows character data from a ControlSource to be displayed. The text may be a literal, in which case the Field is acting as a label. Both single and multi-line text is supported.

Implicit Style Name: field Note that literal fields have the special style name field-literal, so that they can be distinguished from fields with dynamic content.

- Locale : Locale
- HideDuplicates : Boolean
- Orientation : Integer
- LineHeight : String
- Format : Format (view Format)
- ControlSource : ControlSource (view ControlSource)

Format

A Format describes the presentation of text data to represent numeric values, currency, percentages, dates and times. The value of Type can be: "None", "Currency", "Number", "Percent" or "Date/Time".

Implicit Style Name: format

- Type : String
- MinIntegerDigits : int
- MaxIntegerDigits : int
- MinFractionDigits : int
- MaxFractionDigits : int
- GroupingUsed : boolean
- GroupingSize : int
- DecimalSeparatorAlwaysShown : boolean
- CustomPattern : String
- DateFormat : String
- TimeFormat : String

Grid

Grid extends TextElement (view TextElement)

The Data Grid component is used to display data in tabular form. It is mainly used to handle CJK and other top to bottom languages, however it also supports boxed text for use in simulating boxed form-filling for OCR solutions.

Implicit Style Name: grid

- Locale : Locale
- AutoWrap : Boolean
- LineGridMode : String
- LineGridProgression : String
- WritingMode : String
- GlyphOrientation : String
- BoxBorderColor : String
- BoxBorderSide : Boolean
- BoxBorderTop : Boolean
- BoxBorderBottom : Boolean
- BoxBorderHeight : Integer

- CharacterOffsetX : Integer
- CharacterOffsetY : Integer
- Format : Format (view Format)
- ControlSource : ControlSource (view ControlSource)

Group

A Group defines how data records are grouped for output. A group contains a group header and footer which are shown before and after the group records.

Implicit Style Name: group

- Field : String
- SortOrder : String
- GroupHeaderVisible : Boolean
- GroupFooterVisible : Boolean
- GroupOn : String
- GroupData : String
- GroupHeader : GroupHeader (view GroupHeader)
- GroupFooter : GroupFooter (view GroupFooter)

GroupFooter

GroupFooter extends Chunk (view Chunk)

The GroupFooter chunk (band) belongs to Group and contains the components that are rendered after the last record of a group.

Implicit Style Name: group-footer

- KeepWithNext : Boolean
- TableOfContents : boolean

GroupHeader

GroupHeader extends TOCElementHolder (view TOCElementHolder)

The GroupHeader chunk (band) belongs to Group and contains the components that are rendered before the first record of a group.

Implicit Style Name: group-header

- RepeatSection : Boolean
- KeepWithNext : Boolean

HBox

HBox extends Box (view Box)

The horizontal version of Box, that lays cells out left to right. All cells are set to the height of the tallest cell. The cell width varies with weight. If the weight of a cell is changed dynamically (eg. through scripts) then the layoutCells() function of HBox should be called to ensure the cell positions are correctly updated.

Implicit Style Name: hbox

Holder

Holder is an internal object used to copy objects to and from the clipboard. If you copy any object, eg. a Page Setup, Rectangle etc. to the clipboard and paste it into a text document, you will see it is wrapped with a Holder. This is a useful technique for looking at small RML samples without having to keep saving and viewing the RML in a text editor.

Implicit Style Name: holder

List: Content : List<Object>

Image

Image extends RawElement (view RawElement)

The Image component reads an image from a control source. Either as a URL, or as a byte array which can be interpreted into a known image format, for example PNG.

Implicit Style Name: image

- URL : String
- URLDescription : String
- URLTarget : String
- SizeMode : String
- HorizontalAlign : String
- VerticalAlign : String
- ControlSource : ControlSource (view ControlSource)

Line

Line extends RawElement (view RawElement)

The Line component represents a straight line with optional end decorations, such as arrowheads.

Implicit Style Name: line

- Divisor : int
- LineColor : String
- X1 : Integer

- Y1 : Integer
- X2 : Integer
- Y2 : Integer
- LineWidth : Integer
- LineStyle : String
- ArrowStyle1 : String
- ArrowWidth1 : Integer
- ArrowColor1 : String
- ArrowStyle2 : String
- ArrowWidth2 : Integer
- ArrowColor2 : String

Metadata

Metadata is held by the report to track information such as which version of the designer was used to edit the template.

Implicit Style Name: metadata

- Version : int
- Location : String

PageBreak

PageBreak extends RawModelElement (view RawModelElement)

A page break represents an explicit page break. You can choose whether to reset the page numbering for the next page.

Implicit Style Name: page-break

- Name : String
- Position : Integer
- ResetPageCount : Boolean

PageFooter

PageFooter extends Chunk (view Chunk)

A PageFooter is shown at the bottom of each page of a Paged report. No footer is shown for a Streamed report.

Implicit Style Name: page-footer

PageHeader

PageHeader extends Chunk (view Chunk)

A PageHeader is shown at the top of each page of a Paged report. No header is shown for a Streamed report.

Implicit Style Name: page-header

PageSetup

PageSetup holds the attributes for a section when output in Paged mode. Each section in the report can have a different PageSetup, which may even mix Prtrait and Landscape modes.

Implicit Style Name: page-setup

- Name : String
- Paper : String
- Width : int
- Height : int
- Orientation : String
- HorizontalPageCount : int
- Top : int
- Left : int
- Bottom : int
- Right : int
- ColumnCount : int
- ColumnSpacing : int
- RowSpacing : int
- ColumnLayout : String

Parameter

A Parameter is used to hold parameterized information for the report. The value may be a literal, which saves having to repeat the same text in multiple places, allowing for faster customization, or it may be a dynamic parameter, where the user is prompted for information when rendering. Note you must explicitly set the Enabled state of the Parameter if you want it to take effect.

Implicit Style Name: param

- Key : String
- Enabled : Boolean

RawElement

RawElement extends RawModelElement (view RawModelElement)

RawElement is an abstract object that defines common presentation attributes that are shared by most of the report components.

Implicit Style Name: element

- Name : String
- Visible : Boolean
- Fill : Boolean
- Left : Integer
- Top : Integer
- Width : Integer
- Height : Integer
- BorderStyle : String
- BorderWidth : Integer
- BorderColor : String
- BorderLeft : Boolean
- BorderRight : Boolean
- BorderTop : Boolean
- BorderBottom : Boolean
- BorderRadius : Integer
- BackgroundColor : String
- Growable : Boolean
- Shrinkable : Boolean
- PaddingLeft : Integer
- PaddingRight : Integer
- PaddingTop : Integer
- PaddingBottom : Integer

RawElementHolder

RawElementHolder extends BasicRawElementHolder (view BasicRawElementHolder)

RawElementHolder is an abstract object that represents a collection of components in a rectangular area - this is usually a chunk (band) in the report.

Implicit Style Name: element-holder

- BackgroundColor : String
- Caption : String
- ForceNewPage : String
- KeepTogether : Boolean
- Visible : Boolean
- Growable : Boolean
- Shrinkable : Boolean
- Height : Integer
- PaddingBottom : Integer
- VerticalAlign : String

RawModelElement

RawModelElement extends StyledElement (view StyledElement)

A RawModelElement has the basic script management mechanism to control component rendering. All components and chunks inherit this capability.

Implicit Style Name: model-element

- LockHandles : Boolean
- RenderIf : Script (view Script)
- OnRenderBegin : Script (view Script)
- OnRenderEnd : Script (view Script)

RawReport

RawReport extends StyledElement (view StyledElement)

This is the main RML object, that collects together the datasources, parameters styles, page setups and sections needed to render a report.

Implicit Style Name: report

- Locale : Locale
- KeepPageCount : boolean
- CacheAllProperties : boolean
- RepositoryBase : String
- GridEnabled : boolean
- GridX : int
- GridY : int
- Metadata : Metadata (view Metadata)

- Security : Security (view Security)
- FunctionDefinitions : Script (view Script)
- OnRenderBegin : Script (view Script)
- OnRenderEnd : Script (view Script)

List: PageSetup : List<PageSetup>

List: Parameter : List<Parameter>

List: DataSource : List<DataSource>

Map: Style : Map<String,Style>key is value.getName()

List: Stylesheet : List<StylesheetRef>

List: Section : List<Section>

List: SequenceStep : List<ISequenceStep>

Map: RenderDetails : Map<String,RenderDetails>key is value.getMimeType()

RectHolder

RectHolder extends BasicRawElementHolder (view BasicRawElementHolder)

A kind of RawElementHolder that ensures the contents remain within a defined rectangle.

Implicit Style Name: rect-holder

- Left : Integer
- Top : Integer
- Width : Integer
- Height : Integer

Rectangle

Rectangle extends RawElement (view RawElement)

A Rectangle represents a rectangular area on the screen, allowing both fill and border effects. Additionally, a Rectangle can have rounded corners and individual border sides turned on or off.

Implicit Style Name: rect

RenderDetails

RenderDetails are passed from the raw model to the logical model to control output. Each set of RenderDetails is identified by a mime-type and is used when rendering that kind of output. If a report has no render details associated with a particular mime-type, then the default options will be used while rendering. Whenever you use the Render Wizard and modify the render options, these will be saved back as RenderDetails within the report, so that subsequent renders will use the same settings.

Implicit Style Name: render-details

- MimeType : String

Map: Parameter : Map<String,Parameter>key is value.getKey()

RTF

RTF extends RawElement (view RawElement)

The RTF component reads RTF data from a control source and renders it into the component rectangle.

Implicit Style Name: rtf

- ShowAsImage : Boolean
- ImageResolution : Integer
- ControlSource : ControlSource (view ControlSource)

RulerMark

Each report section may define a number of ruler marks that run vertically down the page so that components snap to them when dragged. Similarly, each chunk may define a number of ruler marks that run horizontally across the chunk.

Implicit Style Name: mark

- Value : int

SVG

SVG extends RawElement (view RawElement)

Represents an SVG component. The Dynamic flag indicates whether the contents should be rebuilt for every record processed, or can be cached once and reused (obviously much faster).

Implicit Style Name: svg

- Location : String
- ImageResolution : Integer
- Dynamic : Boolean
- Document : com.elixirtech.org.jdom.Document

Script

A script represents a code fragment, usually in JavaScript, that is executed upon certain events, for example OnRenderBegin.

Implicit Style Name:

- Language : String

Section

Section extends StyledElement (view StyledElement)

A report can contains a number of sections, each with it's own datasource, page setup etc. that are sequenced together to form the final report.

Implicit Style Name: section

- Name : String
- PageSetup : String
- DataSource : String
- BorderStyle : String
- BorderColor : String
- ShowSectionHeader : boolean
- ShowSectionFooter : boolean
- ShowPageHeader : boolean
- ShowPageFooter : boolean
- KeepBlankPages : boolean
- ResetPageCount : boolean
- RenderIf : Script (view Script)
- OnRenderBegin : Script (view Script)
- OnRenderEnd : Script (view Script)
- Watermark : SectionInvocation (view SectionInvocation)
- SectionHeader : SectionHeader (view SectionHeader)
- SectionFooter : SectionFooter (view SectionFooter)
- PageHeader : PageHeader (view PageHeader)
- PageFooter : PageFooter (view PageFooter)
- Detail : Detail (view Detail)

List: Group : List<Group>

List: RulerMark : List<RulerMark>

SectionFooter

SectionFooter extends Chunk (view Chunk)

The SectionFooter is the chunk (band) of components that is shown at the end of rendering a Section.

Implicit Style Name: section-footer

- TableOfContents : boolean

SectionHeader

SectionHeader extends TOCElementHolder (view TOCElementHolder)

The SectionHeader is the chunk (band) of components that is shown at the start of rendering a Section.

Implicit Style Name: section-header

- KeepWithNext : Boolean

SectionInvocation

Each report contains one or more SectionInvocations that control how and when sections are rendered. A section may be rendered just once (a typical case), but may also be rendered multiple times, with the same or different datasources within the same report.

Implicit Style Name: section-invocation

- ReportName : String
- SectionName : String
- DataSourceName : String
- Enabled : Boolean

List: Parameter : List<Parameter>

Security

Security describes the read-only, hide-details and encryption states of the report.

Implicit Style Name: security

- Encrypted : boolean
- Password : String
- ReadOnly : boolean
- HideInternals : boolean

ShapeGroup

ShapeGroup extends RectHolder (view RectHolder)

When components are grouped together a ShapeGroup is created to represent the grouping. ShapeGroups can contain other components, including other ShapeGroups.

Implicit Style Name: shape-group

Style

A style defines a named set of properties (style items) that are applied to every component that adopts that style. This allows global changes to be made to a report just by adjusting a single value. Styles can inherit from another style, the base style, inheriting and overriding the style items it defines.

Implicit Style Name: style

- Name : String
- BaseName : String

Map: Default : Map<String,StyleItem>key is value.getName()

StyleItem

Each style consists of a number of style items that define a name/value pair representing a style choice, eg. Font.Name=Serif.

Implicit Style Name: item

- Name : String
- Value : String

StyledElement

This object forms the root of the RML inheritance hierarchy. All visual RML elements have style.

Implicit Style Name: styled-element

- StyleName : String

SubReport

SubReport extends RawElement (view RawElement)

A SubReport identifies a section, either of the same report or another report that should be rendered within this master one. All SubReports are rendered in Streamed mode, that means there is no page header or page footer in the SubReport contents.

Implicit Style Name: sub-report

- ReportName : String
- SectionName : String
- DataSourceName : String
- KeepTogether : Boolean

List: Parameter : List<Parameter>

TOCElementHolder

TOCElementHolder extends Chunk (view Chunk)

An abstract object that identifies components that can be shown in the table of contents and indicates what text should be shown and how it should be formatted.

Implicit Style Name: toc

- TableOfContents : Boolean
- Locale : Locale
- Format : Format (view Format)
- ControlSource : ControlSource (view ControlSource)

Table

Table extends RawElement (view RawElement)

A Table consists of three table sections, `TableHeader`, `TableBody` and `TableFooter`. When rendering there will be one `TableBody` for each record in the datasource, bracketed with a header and footer, if appropriate.

Implicit Style Name: `table`

- `ShowEmptyTable` : Boolean
- `KeepTogether` : Boolean
- `Cache` : String
- `DataRange` : String
- `Header` : `TableHeader` (view `TableHeader`)
- `Body` : `TableBody` (view `TableBody`)
- `Footer` : `TableFooter` (view `TableFooter`)

TableBody

`TableBody` extends `TableSection` (view `TableSection`)

A table section that represents the detail of the table - there is one table body for each record in the table datasource.

Implicit Style Name: `table-body`

TableFooter

`TableFooter` extends `TableSection` (view `TableSection`)

A table section that represents the footer of the table - it will be rendered after all records, or if no records after the header if `Table.ShowEmptyTable` is true.

Implicit Style Name: `table-footer`

TableHeader

`TableHeader` extends `TableSection` (view `TableSection`)

A table section that represents the header of the table - it will be rendered before any records, or if no records as long as `Table.ShowEmptyTable` is true.

Implicit Style Name: `table-header`

TableSection

`TableSection` extends `RectHolder` (view `RectHolder`)

This abstract object represents a chunk within a table. `TableHeader`, `TableBody` and `TableFooter` are kinds of `TableSection`.

Implicit Style Name: `table-section`

TextElement

TextElement extends RawElement (view RawElement)

A abstract object holding the common attributes for those components that display plain text - Field and Grid.

Implicit Style Name: text-element

- TextAlign : String
- FontName : String
- FontSize : Integer
- FontBold : Boolean
- FontItalic : Boolean
- FontUnderline : Boolean
- FontStrikethrough : Boolean
- FirstLineIndent : Integer
- FontColor : String
- VerticalAlign : String
- URL : String
- URDescription : String
- URLTarget : String
- KeepTogether : Boolean

VBox

VBox extends Box (view Box)

The vertical version of Box, that lays cells out top to bottom. All cells are set to the width of the widest cell. The cell height varies with weight. If the weight of a cell is changed dynamically (eg. through scripts) then the layoutCells() function of VBox should be called to ensure the cell positions are correctly updated.

Implicit Style Name: vbox