# BIRT Chart Interactivity

Functional Specifications

Draft 8: November 1, 2005

## Abstract

*This document describes the functional specifications of the Chart interactivity features for BIRT 2.0*

## Document Revisions

| Version | Date | Primary Author(s) | Description of Changes |
|---|---|---|---|
| Draft 1 | July 21, 2005 | David Michonneau | Initial Draft |
| Draft 2 | July 26, 2005 | David Michonneau | Revised TriggerConditions. Added a new refresh entry in the interactivity menu and actions. Added a new paragraph about the event flow |
| Draft 3 | July 28, 2005 | David Michonneau | Added mockups for default interactivity |
| Draft 4 | August 5, 2005 | David Michonneau | Marked Rotation as POST 2.0 feature. Added chart dimension in menu. More details on JavaScript API. Removed Builder section. |
| Draft 5 | August 24, 2005 | David Michonneau | Added:<br>- Use Cases<br>- Drilldown<br>- Filtering<br>- Interactivity Configuration<br><br>Removed:<br>- 3D rotation<br>- Subtype/Dimension change<br><br>Improved:<br>- Zoom usability<br>- Toolbar |
| Draft 6 | September 22, 2005 | David Michonneau | Dropped toolbar, zoom, filtering, refresh features. Added event handler for SWT/Swing |
| Draft 7 | October 4, 2005 | David Michonneau | Moved drilldown to 2.1. Added enable flag for interactivity api. Updated support matrix. |
| Draft 8 | November 1, 2005 | David Michonneau | More details on SWT/Swing event handlers and SVG Invoke_Script |

# Contents

# 1. Introduction

The chart interactivity allows the users to perform different actions on the chart they are viewing.

Each action is mapped to a user gesture, this mapping is called a trigger. These UI gestures can be standard mouse or keyboard events. The Actions can modify the visuals of the chart in different ways, but they can also perform some change at the viewer level, such as redirect to a new page for instance.

Interactivity features depend on the output type, but also on the environment (report/html/birt viewer). The goal is to achieve the best interactivity using SVG inside the HTML BIRT viewer. Other outputs will benefit some interactivity features as well.

# 2. Use Cases

These use cases are extracted from the Chart interactivity design document of IBM and feedback received on the chart-dev mailing list.

## 2.1 Highlight

A pie chart is displayed in the eclipse workbench. Therefore the chart is rendered in an SWT device. The user has the ability to select a pie slice. When the user select a pie slice the pie slice changes color indicating that it has been selected. Furthermore, property information associated with the pie slice is displayed in the property view.

## 2.2 Visibility Toggle

A line chart consisting of two datasets is displayed in an html page using the SVG renderer. The user has the ability to select a legend item to toggle the visibility of the associated dataset in the plot area.

## 2.3 URL redirection

A pie chart is displayed in an html page using the SVG renderer. When the user selects a pie section the browser is redirected to a URL.

## 2.4 ToolTip

A pie chart is displayed in the eclipse workbench. When the user hovers over a pie section tooltip text is displayed.

## 2.5 DrillDown

### 2.5.1 Example scenario #1

I am viewing a bar chart of world-wide sales by year -- with a bar for each year showing me total sales. I think click on bat for year "2005" and the chart changes to show me the

sales for each quarter in 2005. Effectively, this "drill down" has applied a filter to only look at "2005", and has changed the series from year to quarter. If I wanted, could also drill back out to the year level.

#### 2.5.2 Example scenario #2

I am looking at a pie chart that shows total sales for each product line in each slice of the pie. I click on the pie slice and we drill into a pie chart that shows the total sales numbers for all the individual products in that category.

### 2.6 Custom Event Handlers

A line chart is generated with 10 datasets each of these datasets have 100 points. Visually the chart may be cluttered with data. The ability to filter out datasets during runtime can simply the visualization. One solution is to provide an event handler of a legend item. The event handler can be associated with the legend item of a data set to control the visibility of the data set within the plot area. The event is triggered by a "mouse down" event on the legend item.

## 3. Triggers

### 3.1 Overview

Triggers define a mapping between a trigger condition (a UI gesture) and an action. They can be associated with any Chart Block or Serie (see Block.getTriggers() and Series.getTriggers()).

### 3.2 Trigger Conditions

This table lists all the possible trigger conditions. There are similar to most of the HTML40 intrisic events except that they apply to chart elements and not html elements.

Note that some events such as doubleclick will not work in all renderers/browsers/os. In such case they will simply be ignored.

| Name | Description |
|---|---|
| onclick | Occurs when the pointing device button is clicked over an element |
| ondblclick | Occurs when the pointing device button is double clicked over an element. |
| onmousedown | Occurs when the pointing device button is pressed over an element. |
| onmouseup | Occurs when the pointing device button is released over an element. |
| onmouseover | Occurs when the pointing device is moved onto an element. |
| onmousemove | Occurs when the pointing device is moved while it is over an element. |

| Name | Description |
|---|---|
| onmouseout | Occurs when the pointing device is moved away from an element. |
| onfocus | Occurs when an element receives focus either by the pointing device or by tabbing navigation |
| onblur | Occurs when an element loses focus either by the pointing device or by tabbing navigation |
| onkeydown | Occurs when a key is pressed down on an element |
| onkeypress | Occurs when a key is pressed on an element |
| onkeyup | Occurs when a key is up on an element |
| onload | Occurs when the chart is loaded in the viewer |

Each trigger is associated with any visual component of the chart, called hotspot: that can be a legend item, the chart title, a chart serie, etc… Each hotspot will react to some predefined triggers with one or several actions. Note that this action is not limited to the hotspot it originated.

## 3.3 Trigger Flow

The trigger flow will follow the W3C DOM specifications event model: http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/events.html#Events-flow

That is, trigger conditions will be first captured down the target element and then will bubble back up. Any block having registered the same trigger during those two sequences will have the correspond action executed. The TriggerFlow class will define whether it reacts to the capturing or bubbling sequence, the default being the bubbling sequence. It will also allow the propagation sequence to stop in the bubbling phase.

### 3.3.1 API

A new TriggerFlow class will define three constants:

Capture: the trigger's action will be triggered in the capturing phase

Bubble (default): the trigger's action will be triggered in the bubbling phase

BubbleAndStop: the trigger's action will be triggered in the bubbling phase, and will stop the bubbling flow.

# 4. Actions

Actions are executed in answer to some trigger events.

## 4.1 Description

Here are the possible actions:

#### 4.1.1 Url Redirect

Only works in an html environment. This redirects the browser to a new url. A possible extension is to redirect the browser to a different page of the report or another report.

#### 4.1.2 Show ToolTip

This shows a tooltip on a chart element. It is normally associated with a mouse hover event.

#### 4.1.3 ToggleVisibility

Toggles the visibility of the serie which is associated with the object the event originated from

#### 4.1.4 Highlight (new)

This highlights a chart element. It is normally associated with a mouse click event.

#### 4.1.5 Invoke Script

This invokes a script inside the viewer. The scripts are written in a language specific to the rendering environment and output type.

In SVG, the generated JavaScript method signature will have the 'evt' and 'source' parameter. The source parameter would be an id that identifies the source object. This enables the user to access those parameters in the script for instance identifying the event properties (as which key or mouse button was pressed for instance)

Here is an example:

```
StringBuffer myCallback = new StringBuffer("alert(evt);") ;
ActionImpl.create(ActionType.INVOKE_SCRIPT,
                  ScriptValueImpl.create( myCallback.toString()));
```

The resulting svg file would look like this:

```
<script>
        function callback1992356(evt, source){
          alert(evt);
   }
</script>
```

Notice that a hashcode uniquely identifies the callback method in the svg file.

#### 4.1.6 Event Handler

In case of SWT and Swing it will possible to hook an event handler in java. For this purpose a new interface is available:

```
interface ICallBackAction
{
        void execute(Object event, Object source);
```

}

The event class will depend on the renderer. The source refers to the chart component that triggered the event.

This interface is hooked in the Action like other Action types. The difference is that it is not serialized in the chart model, this is a purely dynamic Action.

It will be used this way:

ICallBackAction myCallBack = new MyCallBackClass();

ActionImpl.create(ActionType.CALLBACK, CallBackValueImpl.create( myCallBack );

A new CallBackValue interface will be derived from ActionValue for this purpose.

## 4.2 Support matrix

Not all actions are supported for all outputs and environments. Here is a table summarizing the supported actions on an output basis for each version of BIRT ( actions are upward compatible)

| Output Type/Action | SVG | SWT | Swing | Static image in PDF | ImageMap in HTML |
|---|---|---|---|---|---|
| Url Redirect | 1.0.1 | 2.0 | 1.0.1 | - | 2.0 |
| Show Tool tip | 1.0.1 | 2.0 | 1.0.1 | - | 2.0 |
| ToggleVisibility | 2.0 | 2.0 | 1.0.1 | - | - |
| Highlight | 2.0 | 2.0 | 2.0 | - | - |
| Invoke Script | 2.0 | - | - | - | 2.0 |
| Event Handler | - | 2.0 | 2.0 | - | - |

# 5. Scripting

When the built-in interactivity features defined by the API are not enough, the user can write his own interactivity script, which is dependent on the output type. This is not to be confused with Chart Scripting which is done at the rendering time.

This release will provide scripting support for SVG (in any svg viewer) and static images (inside a browser)

## 5.1 SVG Scripting

SVG output supports the ECMAScript language, also known as JavaScript.

An API will be provided to allow the user to manipulate the SVG DOM and access the chart engine on the server.

The capabilities of SVG Scripts allow it to communicate with the chart engine at a specific url through web services. It can also modify the SVG output

### 5.1.1 SVG JavaScript API

To be defined in the design document.

## 5.2 Static Images Scripting

It's possible to write scripts for static images, using JavaScript. An API will be provided to communicate with the chart engine or access other report elements. These scripts can only be used in a javascript-enabled environment.

### 5.2.1 Static images Script API

function refresh(Chart) : this will cause the chart element to be regenerated by the server and refreshed inside the browser, using the provided chart model as a parameter.

function Chart getModel(): returns the current chart model

function refresh() : refresh the chart using the current chart model. This is useful for getting live data charts.

## 6. Chart Interactivity API

Several classes in the Chart API define the interactivity properties. These properties are then used by each device renderer. Here are the key classes and interfaces to define interactivity at the API level:

### 6.1 InteractionEvent

The InteractionEvent defines the triggers and action mapping on given hotspots:

public final class **InteractionEvent**
extends PrimitiveRenderEvent
**Author:**
Actuate Corporation
**See Also:**
Serialized Form

---

# Field Summary

| **Fields inherited from class org.eclipse.birt.chart.event.PrimitiveRenderEvent** |
|---|
| DRAW, FILL, iObjIndex |

# Constructor Summary

| |
|---|
| **InteractionEvent**(java.lang.Object source) |

# Method Summary

| | |
|---|---|
| void | **addTrigger**(Trigger t) |
| Action | **getAction**(TriggerCondition tc) |

| | |
|---|---|
| PrimitiveRenderEvent | **getHotSpot**() |
| Trigger[] | **getTriggers**() |
| void | **reuse**(java.lang.Object oNewSource) |
| void | **setHotSpot**(PrimitiveRenderEvent pre) |

## 6.2 Trigger

public interface **Trigger**

extends org.eclipse.emf.ecore.EObject

A representation of the model object '***Trigger***'. This type defines a Trigger. A trigger defines interactivity for a chart component.

The following features are supported:

- *Condition*
- *Action*

**See Also:**
DataPackage.getTrigger()

---

# Method Summary

| | |
|---|---|
| Action | **getAction**()<br>Returns the value of the '***Action***' containment reference. |
| TriggerCondition | **getCondition**()<br>Returns the value of the '***Condition***' attribute. |
| boolean | **isSetCondition**()<br>Returns whether the value of the '*Condition*' attribute is set. |
| void | **setAction**(Action value)<br>Sets the value of the '*Action*' containment reference. |
| void | **setCondition**(TriggerCondition value)<br>Sets the value of the '*Condition*' attribute. |
| Void | **unsetCondition**()<br>Unsets the value of the '*Condition*' attribute. |
| void | **setFlowReaction**(TriggerFlow flow)<br>Set the trigger flow reaction |
| TriggerFlow | **getFlowReaction**()<br>Returns the trigger flow reaction |

### 6.3 Action

public interface **Action**
extends org.eclipse.emf.ecore.EObject

A representation of the model object '***Action***'. This type defines an Action. An action is a property defining interactivity for an element. It is associated in a trigger with a trigger condition that defines when the action is to be processed.

The following features are supported:

- *Type*
- *Value*

**See Also:**
DataPackage.getAction()

---

| Method Summary | |
|---|---|
| ActionType | **getType**()<br>Returns the value of the '***Type***' attribute. |
| ActionValue | **getValue**()<br>Returns the value of the '***Value***' containment reference. |
| boolean | **isSetType**()<br>Returns whether the value of the '*Type*' attribute is set. |
| void | **setType**(ActionType value)<br>Sets the value of the '*Type*' attribute. |
| void | **setValue**(ActionValue value)<br>Sets the value of the '*Value*' containment reference. |
| void | **unsetType**()<br>Unsets the value of the '*Type*' attribute. |

## 7. BIRT HTML Viewer Interactivity

### 7.1 Browser Requirements

Full interactivity will only be available when the user is using a SVG and JavaScript enabled browser. The Chart full interactivity will be available on IE6+ with Adobe SVG Plugin or Firefox 1.1.

### 7.2 Automatic SVG support detection

The report will automatically show SVG content if the browser supports it, otherwise it will use a static image of the chart, with limited interactivity.

# 8. Built-in Interactivity

BIRT will provide default built-in interactivity for the Chart, in addition to what the user defines. This section describes what is this default interactivity. Note that this interactivity behavior is only available in the SVG output.

## 8.1 Drill Down / Drill Up (v2.1)

### 8.1.1 Feature Description

This feature allows the user to click on any orthogonal value of the chart to drill down through the data, and see an updated chart showing more granular data for this value. What it does is:

- Apply a data filter on the chart dataset on the field represented by the clicked value.
- Change the X serie data definition and grouping for that value
- Change the chart type/subtype (optional)
- Change the chart title (optional)

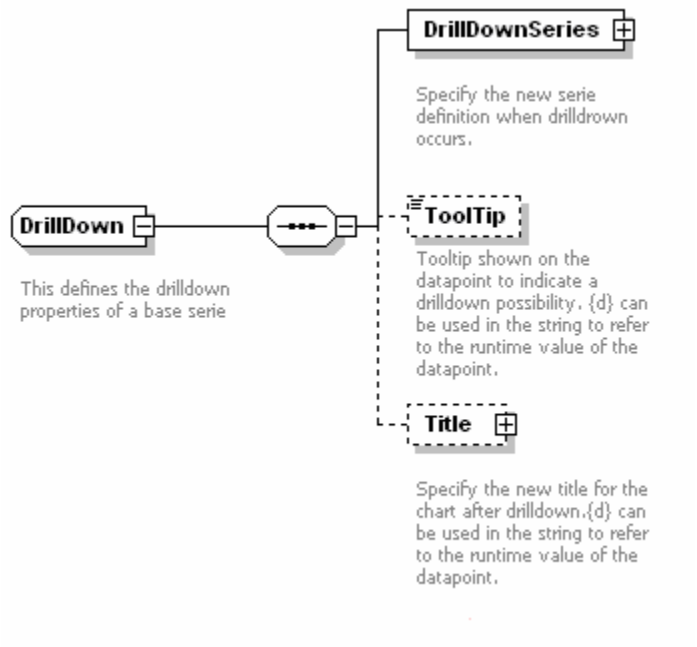
It also allows drilling up to the previous chart, by clicking on a left arrow, at the bottom right of the chart:

### 8.1.2 Levels

Any number of levels is supported by the engine and model, although the chart builder might allow only one or two levels to keep the UI simple.

### 8.1.3 Model API Change

```
<xsd:complexType name="SeriesGrouping">
        …
        <xsd:element name="DrillDown" type="DrillDown" minOccurs="0" maxOccurs="1">
                <xsd:annotation>
                        <xsd:documentation xml:lang="en">
                                Specify the DrillDown properties related to this grouping
                        </xsd:documentation>
                </xsd:annotation>
        </xsd:element>
</xsdcomplexType>
```




```
<xsd:complexType name="DrillDown">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            This defines the drilldown properties of a base serie
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="DrillDownSeries" type="data:SeriesDefinition">
                <xsd:annotation>
                    <xsd:documentation xml:lang="en">
                    Specify the new serie definition when drilldrown occurs.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="ToolTip" type="xsd:string" minOccurs="0">
                <xsd:annotation>
                    <xsd:documentation>Tooltip shown on the datapoint to indicate a drilldown possibility. {d} can be used in
the string to refer to the runtime value of the datapoint.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="Title" type="component:Label" minOccurs="0">
                <xsd:annotation>
                    <xsd:documentation xml:lang="en">
                    Specify the new title for the chart after drilldown.{d} can be used in the string to refer to the runtime value
of the datapoint.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
```

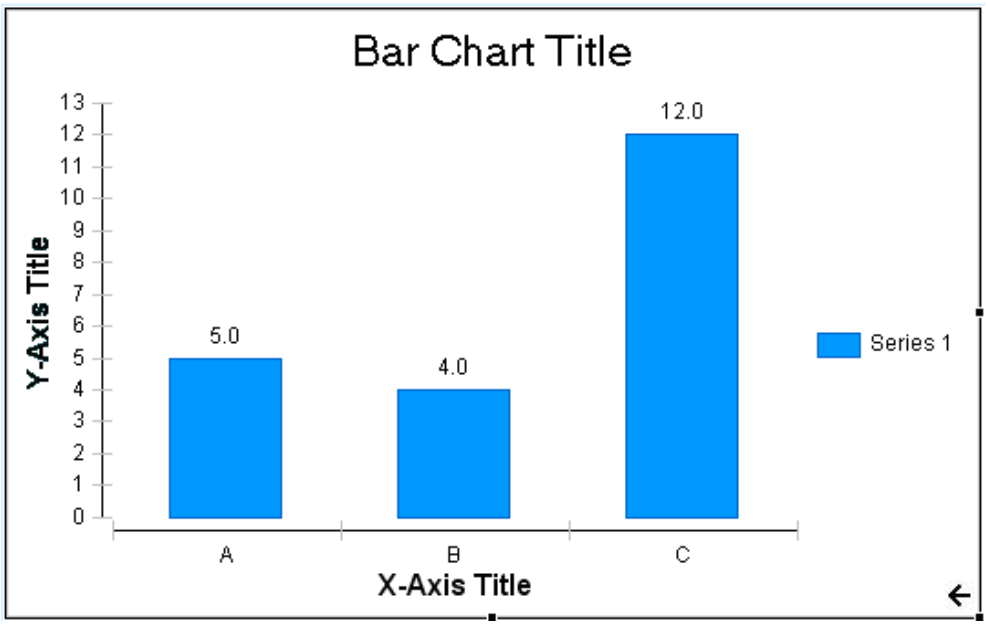
### 8.1.4 Runtime UI

Drilldown/Drillup will be available in SVG, Swing, and SWT. If drilldown is available on a datapoint, a tooltip will appear when hovering the mouse. The user must click on the datapoint he is interested in to drilldown. If the mouseclick event has been overridden with another action, no drilldown will occur.

When the user clicks to drilldown, the chart is regenerated using a new SerieDefinition for the base serie, in order to refine the X data. This could be more granular grouping or

use of a different data field. The original grouping and data definition of the base serie will be overridden by this new one. A filter will also be applied to the new chart with the value of the X value of the datapoint. The title of the chart can possibly change during the drilldown.

A drill-up arrow will appear at the right bottom of the chart, so that the user can drill back up to the previous chart. This icon has a “Drill Up” tooltip.




### 8.1.5 Examples

## 8.2 Legend Interactivity

Legend interactivity responds to mouse left clicks. It can either toggle the visibility of the serie it maps to, or highlight it. This is set at design time by the user (see Configuration API)

## 8.3 Configuration API

The Chart class will have a new optional Interactivity attribute. This Interactivity holds various configuration properties of the interactivity features.

```
<xs:complexType name="Interactivity">
        <xs:sequence>
            <xs:element name=”Enable” type=”xs:boolean” minOccurs=”0”>
                <xs:annotation>
                    <xs:documentation>Enable/Disable all interactive features, true by default</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="LegendBehavior" type="LegendType" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Three possible behaviours: None, Toggle the Serie visibility, Highlight the
serie</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
```

```
</xs:complexType>

<xs:complexType name="LegendType">
    <xs:simpleContent>
        <xs:restriction base="xs:string">
            <xs:enumeration value="ToggleSerieVisibility"/>
            <xs:enumeration value="HighlightSerie"/>
            <xs:enumeration value="None"/>
        </xs:restriction>
    </xs:simpleContent>
</xs:complexType>
```