SAP BusinessObjects Business Intelligence platform
Document Version: 4.2 – 2015-11-12

SAP Crystal Reports JavaScript API Developer Guide

Content

1 Document History

Version	Date	Description
SAP BusinessObjects Business Intelligence platform 4.2	November 2015	Updated the guide with branding changes.

2 About the SAP Crystal Reports JavaScript API

With the SAP Crystal Reports JavaScript API, you can display Crystal reports content in an embedded web application without any client-side component installation. The JavaScript API lets you customize the report viewer, and add interactivity to your Crystal reports content. You can develop your web application in any language that uses JavaScript, because reports are generated in DHTML.

API features include:

- Event listeners for mouse clicks and mouse over events.
- Action listeners for printing and exporting events.
- Methods to customize color and font.
- Methods to show and hide viewer components.

The JavaScript API supports most OpenDocument parameters. If you previously used OpenDocument URLs to link to reports stored in the BI Platform repository, you can transition to the JavaScript API without losing functionality.

3 Using the SAP Crystal Reports JavaScript API

The SAP Crystal Reports JavaScript API is included as part of the SAP BusinessObjects Business Intelligence Platform installation. The API can be accessed by referencing the `ViewerSeed.js` file located on the BIP server.

General workflow, shown in detail in this guide:

1. Find the location of the `ViewerSeed.js` file. In a typical BIP installation, the file is available at the following location: `http://localhost:8080/clientapi/CR/ViewerSeed.js`
2. Retrieve a logon token to use for BI platform authentication.
3. Add a reference to the `ViewerSeed.js` file to your web application.
4. Add the CR Embedded viewer to your web application.
5. Set the report source.

> **Note**
>
> If you are developing your application on your local computer instead of on deployed to a web server, the Adobe Flash Player may prevent the Flex prompting dialog from appearing. You can resolve this issue by deploying your application to a web server or designating your application content as trusted in the Flash Player Global Security Settings.

3.1 Create a report viewer and open a report

1. Set up `head`, `style` and `body` opening and closing tags.
2. Add a script tag within the `head` tags and use it to reference `ViewerSeed.js`.

```
<script src="http://localhost:8080/clientapi/CR/ViewerSeed.js">
```

3. Create a token variable, which will hold the logon token and set it to `null`.

```
var token = null;
```

4. Create a function called `init` that sets the value of `token` to the logon token and creates a new JavaScript viewer instance.

   Refer to the Business Intelligence Platform RESTful Web Services Developer Guide for information on generating a logon token.

```
init(){
  token = "logonToken";
  SAP.CR.Viewer.create("viewerName","container",onViewerInit);
}
```

5. Add a function called `onViewerInit`that sets the report source.

```
function onViewerInit(viewerInstance){
```

```
            viewerInstance.setReportSource('reportID', token);
}
```

The `reportID` is the numerical code assigned to a report.

You can discover the report ID programmatically or through the Central Management Console (CMC). When you right-click a report in the CMC, the report ID is be listed with the report properties. For more information see the SAP Crystal Reports RESTful web services API.

6. Within the `body` tag set onload event handler equal to the `init` function.

```
<body onload="init()">
```

This calls the initialization function once the page is fully loaded, which causes any text or buttons coded into the body to load before the report viewer.

7. Within the `style` tag create a new class that defines the position and size of the JavaScript viewer. In the code below the viewer is given absolute positioning. Its width and height will be 75% of the size of the width and height of the page the viewer will be opened in. The position of the viewer in the page is specified by the `left` and `top` properties.

```
.viewerStyle
{
  position : absolute;
  left : 12.5%;
  top : 20.5%;
  width : 75%;
  height : 75%;
}
```

8. Within the body tags add code to set the style of the viewer.

```
<div id="container" class="viewerStyle"></div>
```

Example

Create a report viewer and open a report

The following example opens a report with report ID 1234 in a report viewer given the name crystalViewer. The value set to the `token` variable in the `init` function is an example.The value you use must be generated by you. The viewer is given absolute positioning. Its width and height will be 75% of the size of the width and height of the page the viewer will be opened in. The position of the viewer in the page is specified by the `left` and `top` properties. The `onViewerFailure` function is optional.

```
<head>

  <script src="http://computername/clientapi/CR/ViewerSeed.js">
  <script>
    var token = null;
    function init(){
      token = "COMMANDCOM-LCM:6400@{3&2=5328,U3&p=40676
        .8926203819,Y7&4F=12,U3&63=secEnterprise
        ,0P&66=60,03&68=secEnterprise:Administrator
        ,0P&qe=100,U3&vz=IVD21LbMCB0eRiI4at
        z9sNL18Ux5anRBdYB9fFv5NrY,UP}";
      SAP.CR.Viewer.create("crystalViewer", 'viewerContainer1', onViewerInit,
onViewerFailure);
    }
    function onViewerInit(){
      viewerInstance.setReportSource('1234', token);
    }
    function onViewerFailure(instance, error){
```

```
      alert(error);
    }

  </script>
</head>
<style>

  .viewerStyle
  {
    position : absolute;
    left : 12.5%;
    top : 20.5%;
    width : 75%;
    height : 75%;
  }
</style>
<body>

  <div id="viewerContainer1" class = "viewerStyle"></div>

</body>
```

3.2 Show and hide viewer controls

The JavaScript API allows you to show or hide the following browser controls in the JavaScript viewer:

- The left panel (used to find, set parameters and view the *Group Tree.*)
- The toolbar.
- The status bar.
- The page navigation bar.

This is done by setting the control to `true` or `false`. By default all browser controls are set to `true`.

Example

The following code causes all control bars and panels to be displayed, except for the status bar.

```
var viewerInstance = SAP.CR.Viewer.getInstance("crystalViewer");
viewerInstance.setDisplayToolbar(true);
viewerInstance.setDisplayLeftPanel(true);
viewerInstance.setDisplayStatusbar(false);
viewerInstance.setDisplayBreadcrumb(true);
```

3.3 Change the look and feel the JavaScript viewer

The JavaScript API allows you to customize the color and font used in the JavaScript viewer.

Example

The following example changes the color of the viewer to a light purple gradient and the font to Times. The first parameter in the `setThemeColor` method is the hexidecimal representation of a color. Use of `true` instead of `false` as the boolean value of the second parameter gives the viewer a solid color instead of a gradient.

```
SAP.CR.Viewer.ThemeManager.setThemeColor("#CDB7F9", false);
SAP.CR.Viewer.ThemeManager.setThemeFont("times");
```

3.4 Event listeners

The JavaScript API supports canvas and action event listeners. Event listeners detect when a certain event has occured, and then trigger a response. Canvas events are mouse events (clicks and mouse-overs) that occur on the report canvas. Action listeners deal with action events, such as clicking a button or selecting an item from a list.

Example

The following code causes report objects to get highlighted in green on a mouseover event. When the curser leaves the report object the background color property is cleared. This means that the default background color of the object is not restored.

```
var canvasListener = new SAP.CR.Viewer.CanvasListener();
var viewerInstance = SAP.CR.Viewer.getInstance("crystalViewer");

canvasListener.onEvent(SAP.CR.Viewer.CanvasEvents.REPORT_ELEMENT_MOUSE_ENTER,funct
ion(args){
  args.target.style.backgroundColor="green";
});
canvasListener.onEvent(SAP.CR.Viewer.CanvasEvents.REPORT_ELEMENT_MOUSE_LEAVE,funct
ion(args){
  args.target.style.backgroundColor=" ";
});
viewerInstance.addCanvasListener(canvasListener);
```

Related Information

SAP.CR.Viewer.CanvasListener [page 12]
SAP.CR.Viewer.ActionListener [page 10]

4 API reference

This section provides a reference of the classes and methods in the JavaScript API.

4.1 SAP.CR.Viewer.ActionListener

The `ActionListener` class allows you to add custom functionality for viewer action events, for most items in the toolbar.

Syntax

```
var actionListener = new SAP.CR.Viewer.ActionListener();
```

4.1.1 onEvent

The `onEvent` method registers an action listener for an event. Multiple listeners can be registered for the same event.

Syntax

```
var actionListener = new SAP.CR.Viewer.ActionListener();
actionListener.onEvent(eventName,listener);
```

Parameters

- `eventName` - The name of the event.
  Possible values:

Value	Description
`SAP.CR.Viewer.ActionEvents.EXPORT`	User clicks the export button.
`SAP.CR.Viewer.ActionEvents.PRINT`	User clicks the print button.
`SAP.CR.Viewer.ActionEvents.DRILL`	User drills down to the report from the group tree, breadcrumb, content or a chart.
`SAP.CR.Viewer.ActionEvents.GROUP_TREE_NAVIGATE`	User clicks on a group tree node that changes the page being viewed.
`SAP.CR.Viewer.ActionEvents.PROMPT`	Report prompts for database or parameters.
`SAP.CR.Viewer.ActionEvents.ERROR`	An error occurs in the viewer.

- `listener` - [function(arg)] Custom function that you define, that determines what happens when the event occurs. The parameter `arg` can be given any name except for JavaScript keyword names, and is created by the handler of the report viewer and passed to the function.

4.1.2 removeEvent

The `removeEvent` method removes listeners from an `ActionListener` object. The method can be used to remove a specific listener, with an associated event name or to remove all listeners with an associated event name.

Syntax

```
var actionListener = new SAP.CR.Viewer.ActionListener();
actionListener.removeEvent(eventName,listener);
```

Parameters

- `eventName` - The name of the event to be removed.
  Possible values:

Value	Description
`SAP.CR.Viewer.ActionEvents.EXPORT`	User clicks the export button.
`SAP.CR.Viewer.ActionEvents.PRINT`	User clicks the print button.
`SAP.CR.Viewer.ActionEvents.DRILL`	User drills down to the report from the group tree, breadcrumb, content or a chart.
`SAP.CR.Viewer.ActionEvents.GROUP_TREE_NAVIGATE`	User clicks on a group tree node that changes the page being viewed.
`SAP.CR.Viewer.ActionEvents.PROMPT`	Report prompts for database or parameters.
`SAP.CR.Viewer.ActionEvents.ERROR`	An error occurs in the viewer.

- `listener` - [function(arg)] The specific listener to be removed. A custom function defined by the user when the event listener was created.

4.2 SAP.CR.Viewer.CanvasListener

The `CanvasListener` class allows you to add custom functionality for mouse events that occur on the report page.

Syntax

```
var canvasListener = new SAP.CR.Viewer.CanvasListener();
```

4.2.1 onEvent

The `onEvent` method registers a canvas listener for an event. Multiple listeners can be registered for the same event.

Syntax

```
var canvasListener = new SAP.CR.Viewer.CanvasListener();
canvasListener.onEvent(eventName,listener);
```

Parameters

- `eventName` - The name of the event.
  Possible values:

Value	Description
`SAP.CR.Viewer.CanvasEvents .REPORT_ELEMENT_MOUSE_ENTER`	Curser enters a report element.
`SAP.CR.Viewer.CanvasEvents .REPORT_ELEMENT_MOUSE_LEAVE`	Curser leaves a report element.
`SAP.CR.Viewer.CanvasEvents .REPORT_ELEMENT_CLICK`	User clicks a report element.
`SAP.CR.Viewer.CanvasEvents .REPORT_ELEMENT_RIGHT_CLICK`	User right-clicks a report element.
`SAP.CR.Viewer.CanvasEvents .REPORT_CANVAS_MOUSE_ENTER`	Curser enters the report canvas.
`SAP.CR.Viewer.CanvasEvents .REPORT_CANVAS_MOUSE_LEAVE`	Curser leaves the report canvas.
`SAP.CR.Viewer.CanvasEvents .REPORT_CANVAS_CLICK`	User clicks the report canvas.
`SAP.CR.Viewer.CanvasEvents .REPORT_CANVAS_RIGHT_CLICK`	User right-clicks the report canvas.

- `listener` - [function(arg)] Custom function that you define, that determines what happens when the event occurs. The parameter `arg` can be given any name except for JavaScript keyword names, and is created by the handler of the report viewer and passed to the function.

4.2.2 removeEvent

The `removeEvent` method removes listeners from a CanvasListener object. The method can be used to remove a specific listener, with an associated event name or to remove all listeners with an associated event name.

Syntax

```
var canvasListener = new SAP.CR.Viewer.CanvasListener();
canvasListener.removeEvent(eventName,listener);
```

Parameters

- `eventName` - The name of the event to be removed.
  Possible values:

Value	Description
`SAP.CR.Viewer.CanvasEvents .REPORT_ELEMENT_MOUSE_ENTER`	Curser enters a report element.
`SAP.CR.Viewer.CanvasEvents .REPORT_ELEMENT_MOUSE_LEAVE`	Curser leaves a report element.
`SAP.CR.Viewer.CanvasEvents .REPORT_ELEMENT_CLICK`	User clicks a report element.
`SAP.CR.Viewer.CanvasEvents .REPORT_ELEMENT_RIGHT_CLICK`	User right-clicks a report element.
`SAP.CR.Viewer.CanvasEvents .REPORT_CANVAS_MOUSE_ENTER`	Curser enters the report canvas.
`SAP.CR.Viewer.CanvasEvents .REPORT_CANVAS_MOUSE_LEAVE`	Curser leaves the report canvas.
`SAP.CR.Viewer.CanvasEvents .REPORT_CANVAS_CLICK`	User clicks the report canvas.

Value	Description
`SAP.CR.Viewer.CanvasEvents.REPORT_CANVAS_RIGHT_CLICK`	User right-clicks the report canvas.

- `listener` - [function(arg)] The specific listener to be removed. A custom function defined by the user when the event listener was created.

4.3 SAP.CR.Parameter

The `SAP.CR.Parameter` class allows you to create parameters and add them to a viewer instance. The class can also be used to set parameter values in a sub-report.

Syntax

```
var param = new SAP.CR.Parameter(parameterName,parameterType);
```

Parameters

- `parameterName` - [String] The name of the parameter in the report. This value is often pre-defined by the report.
- `parameterType` - The type of the parameter.
  Possible values:

Value	Description
`SAP.CR.Parameter.DataTypes.BOOLEAN`	Boolean value (`true` or `false`.)
`SAP.CR.Parameter.DataTypes.CURRENCY`	Numerical value.
`SAP.CR.Parameter.DataTypes.DATE_TIME`	Date and time specified using the JavaScript `Date` function.
`SAP.CR.Parameter.DataTypes.DATE`	Date specified using the JavaScript `Date` function.

Value	Description
`SAP.CR.Parameter.DataTypes.TIME`	Time specified using the JavaScript `Date` function.
`SAP.CR.Parameter.DataTypes.NUMBER`	Numerical value.
`SAP.CR.Parameter.DataTypes.STRING`	String value.

4.3.1 addValue

Sets the value of a parameter.

Syntax

```
var param = new SAP.CR.Viewer.Parameter(parameterName,parameterType);
param.addValue(val);
```

Parameters

- `val` - Parameter value. The type depends on `parameterType`. Can be a discreet or range value.

4.3.2 setReportName

Sets the parameters of a sub report. If you do not use the `setReportName` method the parameters are automatically set to the main report.

Syntax

```
var param = new SAP.CR.Viewer.Parameter(parameterName,parameterType);
param.setReportName(subReportName);
```

Parameters

- `subReportName` - [String] Name of the sub report you are setting the parameters to.

Example

The following example creates a new parameter of type `SAP.CR.DataTypes.NUMBER`, and adds it to the sub report with name `subReport1`.

```
var instance = SAP.CR.Viewer.getInstance("crystalViewer");
var param = new
SAP.CR.Parameter("subNumberParameter",SAP.CR.Parameter.DataTypes.NUMBER);
param.setReportName("subReport1");
param.addValue(10);
instance.setParameters([param]);
```

4.4 SAP.CR.Parameter.RangeValue

The `SAP.CR.Parameter.Range` class is used to specify a range of values for a parameter. The `SAP.CR.Parameter.Range` object is added to an `SAP.CR.Parameter` object using the `addValue` method.

Syntax

```
var range = new SAP.CR.Parameter.RangeValue();
```

4.4.1 setBeginValue

Sets the first of the range of values.

Syntax

```
var range = new SAP.CR.Parameter.RangeValue();
range.setBeginValue(val);
```

Parameters

- `val` - The lower bound of the range of values.
  Possible types for val:

Value	Description
`SAP.CR.Parameter .DataTypes.BOOLEAN`	Boolean value (`true` or `false`.)
`SAP.CR.Parameter .DataTypes.CURRENCY`	Numerical value.
`SAP.CR.Parameter .DataTypes.DATE_TIME`	Date and time specified using the JavaScript `Date` function.
`SAP.CR.Parameter .DataTypes.DATE`	Date specified using the JavaScript `Date` function.
`SAP.CR.Parameter .DataTypes.TIME`	Time specified using the JavaScript `Date` function.
`SAP.CR.Parameter .DataTypes.NUMBER`	Numerical value.
`SAP.CR.Parameter .DataTypes.STRING`	String value.

  The type must be the same as that of the parameter the range value will be added to.

4.4.2 setEndValue

Sets the last of the range of values.

Syntax

```
var range = new SAP.CR.Parameter.RangeValue();
range.setEndValue(val);
```

Parameters

- `val` - The upper bound of the range of values.
  Possible types for val:

Value	Description
`SAP.CR.Parameter .DataTypes.BOOLEAN`	Boolean value (`true` or `false`.)
`SAP.CR.Parameter .DataTypes.CURRENCY`	Numerical value.
`SAP.CR.Parameter .DataTypes.DATE_TIME`	Date and time specified using the JavaScript `Date` function.
`SAP.CR.Parameter .DataTypes.DATE`	Date specified using the JavaScript `Date` function.
`SAP.CR.Parameter .DataTypes.TIME`	Time specified using the JavaScript `Date` function.
`SAP.CR.Parameter .DataTypes.NUMBER`	Numerical value.
`SAP.CR.Parameter .DataTypes.STRING`	String value.

  The type must be the same as that of the parameter the range value will be added to.

4.4.3 setLowerBound

Sets the type of the lower bound of the range parameter.

Syntax

```
var range = new SAP.CR.Parameter.RangeValue();
range.setLowerBound(boundType);
```

Parameters

- `boundType` - Determines the type of the lower bound.
  Possible values:

Value	Description
`SAP.CR.Parameter.Parameter .RangeBoundTypes.INCLUSIVE`	Range of values includes the begin value.
`SAP.CR.Parameter.Parameter .RangeBoundTypes.EXCLUSIVE`	Range of values does not include the begin value.
`SAP.CR.Parameter.Parameter .RangeBoundTypes.UNBOUNDED`	No lower bound.

4.4.4 setUpperBound

Sets the type of the upper bound of the range parameter.

Syntax

```
var range = new SAP.CR.Parameter.RangeValue();
range.setUpperBound(boundType);
```

Parameters

- `boundType` - Determines the type of the upper bound.
  Possible values:

Value	Description
`SAP.CR.Parameter.Parameter .RangeBoundTypes.INCLUSIVE`	Range of values includes the end value.
`SAP.CR.Parameter.Parameter .RangeBoundTypes.EXCLUSIVE`	Range of values does not include the end value.
`SAP.CR.Parameter.Parameter`	No upper bound.

Value	Description
`.RangeBoundTypes.UNBOUNDED`	

4.5 SAP.CR.Viewer

The SAP.CR.Viewer class contains methods that allows you to create an instance of the report viewer, and to customize and add functionality to it.

4.5.1 addActionListener

Adds an action event listener to a viewer instance.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.addActionListener(aListener);
```

Parameters

- `aListener` - Name of the action listener you are adding. Must be an instance of SAP.CR.ActionListener.

4.5.2 addCanvasListener

Adds a canvas event listener to a viewer instance.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.addCanvasListener(cListener);
```

Parameters

- `cListener` - Name of the canvas listener you are adding. Must be an instance of SAP.CR.CanvasListener.

4.5.3 batchExecute

The `batchExecute` method ensures that API calls are executed synchronously.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.batchExecute(function());
```

Parameters

- `function` - [function()] Function defined by you that contains the API calls to be executed in the order they are to be executed.

Example

The following example drills down on a group path and then sets the page number to 2. If the `batchExecute` method was not used, two asynchronous calls would be sent to the server at the same time. This would cause only one of the two API calls to occur.

```
var viewerInstance = SAP.CR.Viewer.getInstance("crystalViewer");
viewerInstance.batchExecute(function(){
  viewerInstance.drillDown(["0"]);
  viewerInstance.setPageNumber(2);
});
```

4.5.4 create

Creates a CR Embedded viewer instance.

Syntax

```
SAP.CR.Viewer.create(viewerName,containerID,initCB,failCB)
```

Parameters

- `containerID` - [String] ID of the element that will hold the viewer.
- `viewerName` - [String] Name of the viewer.
- `initCB` - [function(instance)] Function that initializes the viewer instance. It is not defined within the method call. The parameter `instance` is the viewer instance.
- `failCB` - [function(instance,error)] Function that is executed if something fails. It is not defined within the method call. This is an optional parameter. The parameter `instance` is the viewer instance.

4.5.5 drilldown

Drills down to a group.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.drilldown([groupPath]);
```

Parameters

- `groupPath` - An array of integers, which displays the path you have drilled down on.

4.5.6 getInstance

Gets a viewer instance based on the viewer name.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
```

Parameters

- viewerName - [String] Name of the viewer you are getting.

4.5.7 refresh

Refreshes the report displayed in the JavaScript viewer. The method refreshes the data in the report being displayed and if necessary prompts for parameters or logon information.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.refresh();
```

4.5.8 removeActionListener

Removes an action event listener from a viewer instance.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.removeActionListener(aListener);
```

Parameters

- aListener - Name of the action listener you are removing.

4.5.9 removeCanvasListener

Removes a canvas event listener from a viewer instance.

Syntax

```
var instance = SAP.CR.Viewer.getInstance(viewerName);
instance.removeCanvasListener(cListener);
```

Parameters

- cListener - Name of the canvas listener you are removing.

4.5.10 setDisplayBreadcrumb

Show or hide the breadcrumb.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.setDisplayBreadcrumb(isBreadcrumb);
```

Parameters

- `isBreadcrumb` - [Boolean] True to show breadcrumb or false to hide breadcrumb.

4.5.11 setDisplayLeftPanel

Show or hide the left panel, which can be used as the find panel, to view the *Group Tree*, or to view and edit parameters.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.setDisplayLeftPanel(isLeftPanel);
```

Parameters

- `isLeftPanel` - [Boolean] Set to `true` to show panel or `false` to hide panel.

4.5.12 setDisplayStatusBar

Show or hide the status bar along the bottom of the report viewer.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.setDisplayStatusBar(isStatusBar);
```

Parameters

- `isStatusBar` - [Boolean] Set to `true` to show status bar or `false` to hide status bar.

4.5.13 setDisplayToolbar

Show or hide the toolbar.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.setDisplayToolbar(isToolbar);
```

Parameters

- `isToolbar` - [Boolean] True to show toolbar or false to hide toolbar.

4.5.14 setHasLogo

Shows or hides the logo in the top, right corner of the report viewer. By default the logo is shown.

> **Note**
>
> Must be called before the viewer has initialized, or an exception is thrown.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerInstance);
viewerInstance.setHasLogo(isLogo);
```

Parameters

- `isLogo` - [Boolean] Set to `true` to show the logo or to `false` to hide it.

4.5.15 setHasRefreshButton

Shows or hides the refresh button in the report viewer's toolbar. By default the refresh button is hidden.

> **Note**
>
> Must be called before the viewer has initialized, or an exception is thrown.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.setHasRefreshButton(isRefreshButton);
```

Parameters

- `isRefreshButton` - [Boolean] Set to `true` to show the refresh button or to `false` to hide it.

4.5.16 setLogo

Sets the image to be used as a logo in the top, right corner. Allows you to link a url to the logo and set a tooltip for when you mouseover the logo.

> **Note**
>
> Must be called before the viewer has initialized, or an exception is thrown.

> **Note**
>
> The image should be at most 100x125 pixels in size.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.setLogo(url, link, tooltip);
```

Parameters

- `url` - [String] Relative or absolute path to the image to be used as the logo.
- `link` - [String] The url that will be opened when the user clicks the logo.
- `tooltip` - [String] The tooltip that appears when you mouseover the logo. If it contains non-English characters it must be encoded.

4.5.17 setPageNumber

Sets the page number of the report page being viewed.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.setPageNumber(pageNumber);
```

Parameters

- `pageNumber` - [number] A positive number. If larger than the number of pages in the report the page number is set to the last page in the report.

4.5.18 setParameters

Sets report parameters. Multiple parameters can be set in one call of the method.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.setParameters([param]);
```

Parameters

- `param` - An array of the parameter(s) being set to the viewer.

4.5.19 setPrintMode

Sets the print mode of the report viewer to either ACTIVEX or PDF.

> **Note**
>
> Must be called before the viewer has initialized, or an exception is thrown.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.setReportMode(mode);
```

Paramters

- `mode` - The print mode.
  Possible values:

Value	Description
`SAP.CR.PrintMode.ACTIVEX`	Print using the ActiveX printer. Only works in Internet Explorer.
`SAP.CR.PrintMode.PDF`	Print to PDF.

4.5.20 setPromptOnRefresh

Sets whether or not the viewer prompts for new parameter values after being refreshed. By default the viewer prompts for new parameter values.

> **Note**
>
> Must be called before the viewer has initialized, or an exception is thrown.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerIntance.setPromptOnRefresh(isPrompt);
```

Parameters

- `isPrompt` - [Boolean] Set to `true` to prompt for new parameters or to `false` to use existing parameter values.

4.5.21 setReportMode

Sets the color of the background behind the report page and the alignment of the report pages in the report viewer.

> **Note**
>
> Must be called before the viewer has initialized, or an exception is thrown.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInsatnce(viewerName);
viewerInstance.setReportMode(mode);
```

Parameters

- `mode` - Sets the color and alignment of the report page in the report viewer.
  Possible values:

Value	Description
`SAP.CR.Viewer .ReportMode.PRINT`	Sets the background of the report viewer to grey.
`SAP.CR.Viewer .ReportMode.WEB`	Sets the background of the report viewer to white. Report pages are not center aligned in the viewer.

4.5.22 setReportSource

Sets the report source based on a numerical object ID and an enterprise session. If needed you can also set the document locale.

Syntax

```
var viewerInstance = SAP.CR.Viewer.getInstance(viewerName);
viewerInstance.setReportSource(infoObjectID,token,locale);
```

Parameters

- `infoObjectID` - [String] ID or report in info view/CMC.
- `token` - [String] Logon token. The token expires after a few minutes of inactivity and needs to be regenerated. Refer to the Business Intelligence Platform RESTful Web Services Developer Guide for how to generate a logon token.
- `locale` - [String] The document locale.

4.6 SAP.CR.Viewer.ThemeManager

The `ThemeManager` class allows you to set the color and font style used in the viewer.

4.6.1 setThemeColor

Sets the color of the viewer.

Syntax

```
SAP.CR.Viewer.ThemeManager.setThemeColor(hexColor,isSolid);
```

Parameters

- `hexColor` - [String] The hexadecimal representation of the color being used. Value always starts with the `#` symbol.
- `isSolid` - [Boolean] Set to `true` for a solid color or set to `false` for a gradient color.

4.6.2 setThemeFont

Sets the family font name used in the JavaScript viewer UI. This does not effect the font used in the report being viewed.

Syntax

```
SAP.CR.Viewer.ThemeManager.setThemeFont(font);
```

Parameters

- `font` - [String] The font to be used in the viewer. Any browser supported fonts can be used.

Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

Accessibility

The information contained in the SAP documentation represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of wilful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP's gross negligence or willful misconduct. All links are categorized for transparency (see: http://help.sap.com/disclaimer).

www.sap.com/contactsap