Actuate One™

One Design
One Server
One User Experience

Using Actuate JavaScript API

Document No. 130131-2-791301 January 23, 2013

Contents

About Using Actuate JavaScript API

Using Actuate JavaScript API is a guide to designing custom reporting web applications with the Actuate JavaScript API.

Using Actuate JavaScript API includes the following chapters:

- *About Using Actuate JavaScript API.* This chapter provides an overview of this guide.
- *Chapter 1. Creating a custom web page using the Actuate JavaScript API.* This chapter describes the Actuate JavaScript API requirements and common implementations.
- *Chapter 2. Creating dynamic report content using the Actuate JavaScript API.* This chapter describes using Actuate JavaScript API code in a BIRT report.
- *Chapter 3. Working with BIRT Data Analyzer and cross tabs.* This chapter describes the BIRT Data Analyzer and the use of cross tabs.
- *Chapter 4. Actuate JavaScript API classes.* This chapter lists all of the standard Actuate JavaScript API classes and their methods.
- *Chapter 5. BIRT Data Analyzer API classes.* This chapter lists all of the cross tab classes and their methods.

Chapter

1

Creating a custom web page using the Actuate JavaScript API

This chapter contains the following topics:

- About the Actuate JavaScript API
- Accessing the Actuate JavaScript API
- Establishing an HTTP session with an Actuate web application
- About Actuate JavaScript API security integration
- Viewing reports
- Navigating repository content using ReportExplorer
- Using dashboards and gadgets
- Using and submitting report parameters
- Retrieving report content as data

About the Actuate JavaScript API

The Actuate JavaScript API enables the creation of custom web pages that use Actuate BIRT report elements. The Actuate JavaScript API handles connections, security, and content. The Actuate JavaScript API classes functionally embed BIRT reports or BIRT report elements into web pages, handle scripted events within BIRT reports or BIRT report elements, package report data for use in web applications, and operate the BIRT Viewer and Data Analyzer.

To use the Actuate JavaScript API, connect to Actuate Information Console or Deployment Kit for BIRT Reports.

The Actuate JavaScript API uses the Prototype JavaScript Framework. The following directory contains the Actuate JavaScript API source files:

```
<Context Root>\iportal\jsapi
```

The base class in the Actuate JavaScript API is actuate. The actuate class is the entry point for all of the Actuate JavaScript API classes. The actuate class establishes connections to the Actuate web application services. The Actuate JavaScript API uses HTTP requests to retrieve reports and report data from an Actuate web service. The subclasses provide functionality that determines the usage of the reports and report data.

Many functions in the Actuate JavaScript API use a callback function. A callback function is a custom function written into the web page that is called immediately after the function that calls it is finished. A callback function does not execute before the required data or connection has been retrieved from the server.

Many of the callback functions in the Actuate JavaScript API use a passback variable. A passback variable contains data that is passed back to the page by the calling function. A callback function that uses an input parameter as a passback variable must declare that input parameter.

Accessing the Actuate JavaScript API

To use the Actuate JavaScript API from a web page, add a script tag that loads the Actuate JavaScript API class libraries from an Actuate application.

Start with a web page that contains standard HTML elements, as shown in the following code:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
   <head>
   <meta http-equiv="content-type" content="text/html;
   charset=utf-8" />
   </head>
   <body>
      <div id="viewer1">
         <script type="text/javascript" language="JavaScript"
            src="http://localhost:8700/iportal/jsapi"></script>
         <script type="text/javascript" language="JavaScript">
         … <!--functionality goes here-->
         </script>
      </div>
   </body>
</html>
```

The <script> element nested in the <div> element imports the Actuate JavaScript API libraries into the web page's context. For example:

```
<script type="text/javascript" src="http://localhost:8700
   /iportal/jsapi">
</script>
```

where

- localhost:8700 is the host name and TCP port for an available Actuate application host.
- /iportal is the context root for the Actuate web service.
- /jsapi is the default location of the Actuate JavaScript API libraries.

Use additional script tags to call JavaScript functions for the page. Use the actuate.load( ) function to enable the components of the Actuate JavaScript API.

The scripts in this section are encapsulated in <div> tags for portability. Encapsulated Actuate JavaScript API functions can be used in any web page.

About the DOCTYPE tag

To render the page in standards compliance mode, specify strict.dtd in the DOCTYPE tag at the top of the page. Standards compliance mode makes the page layout and behaviors significantly more consistent. Pages without this definition render inconsistently.

About UTF8 character encoding

Use a <meta> tag to direct the browser to use UTF8 encoding for rendering and sending data. UTF8 encoding prevents the loss of data when using internationalized strings.

Establishing an HTTP session with an Actuate web application

The actuate class is the general controller for the HTTP session. Call actuate.initialize( ) to establish a connection to an Actuate application. Load the elements that are selected by actuate.load( ) before accessing reports or applications. Initialization establishes a session with an Actuate service. To initialize the actuate object, call the actuate.initialize( ) initialization function. To use actuate.initialize( ), provide connection parameters as shown in the following code:

```
actuate.initialize("http://localhost:8700/iportal", null, null,
   null, runReport, null);
```

where

- http://localhost:8700/iportal is a URL for the Actuate report application service. This URL must correspond to an Actuate Deployment Kit for BIRT Report application or Information Console application.
- null specifies the default settings for RequestOptions that are provided by the connected Actuate web application. RequestOptions sets custom or additional URL parameters for the request. To use custom or additional URL parameters, construct an actuate.RequestOptions object, assign the specific values to the object, and put the object into the URL parameter.
- The third and fourth parameters are reserved. Leave these parameters as null.
- runReport is the callback function called after the initialization finishes. Specify the callback function on the same page as the initialize function. The callback function cannot take a passback variable.
- null specifies the optional errorCallback parameter. The errorCallback parameter specifies a function to call when an error occurs.

The initialization procedure in this section is the first step in using Actuate JavaScript API objects. Nest the initialization code in the second <script> element in the <div> element of the page.

The runReport( ) function is used as a callback function that executes immediately after actuate.initialize( ) completes. The page must contain runReport( ).

About Actuate JavaScript API security integration

The web service that provides reports also establishes security for a reporting web application. The actuate.initialize( ) function prompts users for authentication information if the web service requires authentication. The

Actuate JavaScript API uses a secure session when a secure session already exists. Remove authentication information from the session by using actuate.logout( ).

To integrate an Actuate JavaScript API web page with an Actuate reporting web service, identify the web service from the following list:

- Deployment Kit using file-system repositories: Actuate Java Components provide web services that are secured by the application server that runs those services. These applications do not perform an authentication step initially, which enables the Actuate JavaScript API to integrate smoothly. See *Actuate BIRT Java Components Developer Guide* for information about customizing security for Actuate Deployment Kit.
- Deployment Kit using an Encyclopedia volume repository: Encyclopedia volumes are managed by Actuate BIRT iHub. To connect to a Deployment Kit that accesses an Encyclopedia volume, an Actuate JavaScript API web page prompts the user for a user name and password if a secure session has not been established. See *Actuate BIRT Java Components Developer Guide* for information about customizing security for Actuate Deployment Kit.
- Information Console: Actuate Information Console connects to an Encyclopedia volume and requires authentication. To connect to an Information Console, an Actuate JavaScript API web page prompts the user for a user name and password if a secure session has not been established. Information Console provides a login page to establish the secure session. See *Information Console Developer Guide* for information about customizing security for Actuate Information Console.

Establishing a secure connection to more than one web service

The actuate.initialize( ) function establishes a session with one Actuate web application service, requesting authentication from the user when the web service requires authentication. Use the actuate.authenticate( ) function for additional secure sessions. Call actuate.authenticate( ) to establish secure sessions with additional web services. Call actuate.initialize( ) before calling actuate.authenticate( ).

Use authenticate( ) as shown in the following code:

```
actuate.authenticate(serviceurl,
                     null,
                     userID,
                     userpassword,
                     null,
                     callback,
                     errorcallback);
```

where

- serviceurl is a URL for the Actuate web application service in use. This URL must correspond to an Actuate Deployment Kit for BIRT Reports or Information Console application.
- null specifies the default settings for the RequestOptions object that is provided by the connected Actuate web application. RequestOptions sets custom or additional URL parameters for the request. To use custom or additional URL parameters, construct an actuate.RequestOptions object, assign the specific values to the object, and put the object into the custom or additional URL parameter.
- userID is the userid for authentication when loading Actuate JavaScript API resources. To force a user login, set this parameter to null.
- userpassword is the password for the userid parameter to complete authentication when loading Actuate JavaScript API resources. Use null to force the user to log in.
- null specifies no additional user credentials. This parameter holds information that supports external user credential verification mechanisms, such as LDAP. Add any required credential information with this parameter where additional security mechanisms exist for the application server upon which the web service is deployed.
- callback is a function to call after the authentication completes.
- errorcallback is a function to call when an exception occurs.

After authenticate( ) finishes, access resources from the Actuate web application service at the URL in serviceurl.

Application servers share session authentication information to enable a user to log in to one application context root and have authentication for another. For example, for Apache Tomcat, setting the crossContext parameter to "true" in the server.xml Context entries allows domains to share session information. The entries to share the authentication information from the web application with an Actuate Java Component look like the following example:

```
<Context path="/MyApplication" crossContext="true" />
<Context path="/ActuateJavaComponent" crossContext="true" />
```

Using a login servlet to connect to an Actuate web application

Actuate web applications provide a login servlet, loginservlet, that establishes a secure session with an Actuate web application service. Use the following code to use a form that calls loginservlet explicitly from a login page:

```
<form name="Login"
action="https://myApp/iPortal/loginservlet?" function="post">
   <input type="text" name="userID" />
   <input type="text" name="password" />
   …
</form>
```

This code sets username and password variables in the session. When initialize( ) runs, the Actuate JavaScript API looks up the session map in the current HTTP session, using the service URL as the key. The Actuate JavaScript API finds the session established by login servlet and accepts the authentication for that service URL.

The login servlet authenticates the connection to an Actuate web service. Do not call the actuate.authenticate( ) function to authenticate the connection when using loginservlet.

Using a custom servlet to connect to an Actuate web application

Actuate web applications provide single-sign-on functionality to authenticate users using a custom security adapter. See *Actuate BIRT Java Components Developer Guide* or *Information Console Developer Guide* for details on creating and using a custom security adapter matching a specific deployment scenario.

Unloading authentication information from the session

The Actuate JavaScript API keeps authentication information encrypted in the session. To remove this information from the session, use actuate.logout( ). Use logout( ) as shown in the following code:

```
actuate.logout(serviceurl,
               null,
               callback,
               errorcallback);
```

where

- serviceurl is a URL for the Actuate web application service to log out from. This URL must correspond to an Actuate Deployment Kit for BIRT Reports or Information Console application.
- null specifies the default settings for RequestOptions that are provided by the connected Actuate web application. RequestOptions sets custom or additional URL parameters for the request. To use custom or additional URL parameters, construct an actuate.RequestOptions object, assign the specific values to the object, and put the object into the custom or additional URL parameter.

- callback is a function to call after logout( ) completes.
- errorcallback is a function to call when an exception occurs.

After logout( ) finishes, the authentication for the serviceurl is removed. Authenticate again to establish a secure connection.

Viewing reports

The actuate.Viewer class loads and displays reports and report content. Load actuate.Viewer with actuate.load( ) before calling actuate.initialize( ), as shown in the following code:

```
actuate.load("viewer");
```

Load support for dialog boxes from the Actuate JavaScript API using the actuate.load function, as shown in the following code:

```
actuate.load("dialog");
```

Load the viewer and dialog components to use the viewer on the page. Call actuate.Viewer functions to prepare a report, then call the viewer's submit function to display the report in the assigned <div> element.

The actuate.Viewer class is a container for Actuate reports. Create an instance of actuate.Viewer using JavaScript, as shown in the following code:

```
var myViewer = new actuate.Viewer( "viewer1" );
```

The "viewer1" parameter is the name value for the <div> element which holds the report content. The page body must contain a <div> element with the id viewer1 as shown in the following code:

```
<div id="viewer1"></div>
```

Use setReportName( ) to set the report to display in the viewer, as shown in the following code:

```
myViewer.setReportName("/public/customerlist.rptdocument");
```

SetReportName accepts a single parameter, which is the path and name of a report file in the repository. In this example, "/public/customerlist.rptdesign" indicates the Customer List report design in the /public directory.

Call viewer.submit( ) to make the viewer display the report, as shown in the following code:

```
myViewer.submit( );
```

The submit( ) function submits all the asynchronous operations that previous viewer functions prepare and triggers an AJAX request for the report. The Actuate web application returns the report and the page displays the report in the assigned <div> element.

This is an example of calling viewer( ) in a callback function to display a report:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
   <meta http-equiv="content-type" content="text
      /html;charset=utf-8" />
   <title>Viewer Page</title>
</head>
<body onload="init( )">
<div id="viewerpane">
   <script type="text/javascript" language="JavaScript"
      src="http://localhost:8700/iportal/jsapi"></script>

   <script type="text/javascript" language="JavaScript">

   function init( ){
      actuate.load("viewer");
      actuate.initialize( "http://localhost:8700/iportal", null,
            null, null, runReport);
   }  function runReport( ) {
      var viewer = new actuate.Viewer("viewerpane");
      viewer.setReportName("/Public/BIRT and BIRT Studio
              Examples/Top 5 Sales Performers.rptdesign");
      viewer.submit(callback);
   }
   </script>
</div>
</body>
</html>
```

The viewer component displays an entire report. If the report is larger than the size of the viewer, the viewer provides scroll bars to navigate the report. To display a specific element of a report instead of the whole report, use viewer.setReportletBookmark( ) prior to calling submit( ), as shown in the following code:

```
function init( ) {
   actuate.load("viewer");
   actuate.initialize( "http://localhost:8700/iportal", null,
         null, null, runReport);
   function runReport( ) {
   var viewer = new actuate.Viewer("viewerpane");
   viewer.setReportName("/Public/BIRT and BIRT Studio
           Examples/Top 5 Sales Performers.rptdesign");
   viewer.setReportletBookmark("FirstTable");
   viewer.submit(callback);
```

When the FirstTable bookmark is assigned to any table, this code displays that table.

Any changes to the report display must take place after viewer.submit( ) completes. Embed presentation code in a callback class to ensure proper execution.

Controlling viewer user interface features

Control the viewer controls and interface features with the actuate.viewer.UIOptions class. Create an instance of this class using JavaScript, as shown in the following code:

```
var uioptions = new actuate.viewer.UIOptions( );
```

Set the user interface options with the enable functions in the actuate.viewer.UIOptions class. For example, a toolbar appears in the viewer by default, as shown in Figure 1-1.

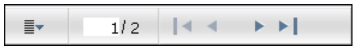


Figure 1-1 The default toolbar for the JavaScript API viewer

To disable this toolbar, use the following code:

```
uioptions.enableToolBar(false);
```

All of the enable functions take a Boolean value as an argument. To configure the viewer to use these options, use setUIOptions( ) as shown in the following code:

```
viewer.setUIOptions(uioptions);
```

The setUIOptions( ) function accepts one parameter: an actuate.viewer.UIOptions object. The viewer's submit( ) function commits the user interface changes to the viewer when the function sends the object to the HTML container. Set the UI options using setUIOptions( ) before calling submit( ).

Accessing report content

Use the actuate.report subclasses to access report content that is displayed in the viewer. For example, use the actuate.report.Table subclass to manipulate a specific table on a report. To manipulate a specific text element in a report, use the actuate.Viewer.Text subclass. Use viewer.getCurrentPageContent( ) to access specific subclasses of actuate.report as shown in the following code:

```
var myTable= myViewer.getCurrentPageContent( ).
   getTableByBookmark("mytable");
```

Identify report elements by their bookmarks. Set bookmarks in the report design. The viewer subclasses access specific report elements and can change how they are displayed.

To hide a particular data column in the table, use code similar to the following function as the callback function after submitting the viewer:

```
function hideColumn( ){
var myTable=
   myViewer.getCurrentPageContent().getTableByBookmark("mytable");
if ( myTable) {
   myTable.hideColumn("PRODUCTCODE");
   myTable.submit( );
   }
}
```

Hiding the column PRODUCTCODE suppresses the display of the column from the report while keeping the column in the report. Elements that use the PRODUCTCODE column from mytable retain normal access to PRODUCTCODE information and continue to process operations that use PRODUCTCODE information.

Accessing HTML5 Chart features

HTML5 charts are accessed from the viewer using actuate.viewer.getCurrentPageContent( ).getChartByBookmark( ) like other report charts. To access HTML5 chart features, use the actuate.report.HTML5Chart.ClientChart object to handle the chart. For example, to access the HTML5 chart with the HTML5ChartBookmark, use the following code:

```
var bchart = this.getViewer().getCurrentPageContent().
   getChartByBookmark("HTML5ChartBookmark");
var clientChart = bchart.getClientChart();
```

ClientChart provides access to ClientOptions, which can change chart features. For example, to change an HTML5 chart title to Annual Report, use the following code:

```
clientChart.getClientOptions().setTitle('Annual Report');;
clientChart.redraw();
```

Using a filter

Apply a data filter to data or elements in a report, such as a charts or tables, to extract specific subsets of data. For example, the callback function to view only the rows in a table with the CITY value of NYC, uses code similar to the following function:

```
function filterCity(pagecontents){
var myTable = pagecontents.getTableByBookmark("bookmark");

var filters = new Array( );
var city_filter = new actuate.data.Filter("CITY",
   actuate.data.Filter.EQ, "NYC");
filters.push(city_filter);

myTable.setFilters(filters);
myTable.submit(nextStepCallback);
}
```

In this example, the operator constant actuate.data.filter.EQ indicates an equals (=) operator.

Using a sorter

A data sorter can sort rows in a report table or cross tab based on a specific data column. For example, to sort the rows in a table in descending order by quantity ordered, use code similar to the following function as the callback function after submitting the viewer:

```
function sortTable( ){
var btable = this.getViewer( ).getCurrentPageContent( ).
   getTableByBookmark("TableBookmark");

var sorter = new actuate.data.Sorter("QUANTITYORDERED", false);
var sorters = new Array( );
sorters.push(sorter);

btable.setSorters(sorters);
btable.submit( );
}
```

The first line of sortTable( ) uses the this keyword to access the container that contains this code. Use the this keyword when embedding code in a report or report element. The this keyword doesn't provide reliable access to the current viewer when called directly from a web page.

Navigating repository content using ReportExplorer

Use the actuate.ReportExplorer class to navigate and view the contents of a Encyclopedia volume in a generic graphical user interface. Load the actuate.ReportExplorer class with actuate.load( ), as shown in the following code:

```
actuate.load("reportexplorer");
```

Call actuate.ReportExplorer functions to identify the root directory to display then call the ReportExplorer's submit function to display the content in the assigned <div> element.

The ReportExplorer class requires the use of a pre-existing actuate.RequestOptions object loaded with initialize. To use the default RequestOptions, use the RequestOptions constructor and provide the object as a parameter to the initialize call, as shown in the following code:

```
requestOpts = new actuate.RequestOptions( );
actuate.initialize( "http://localhost:8900/iportal", requestOpts,
   null, null, runReportExplorer);
```

Displaying ReportExplorer

The actuate.ReportExplorer class is a GUI that displays repository contents. Create an instance of the actuate.ReportExplorer class using JavaScript, as shown in the following code:

```
var explorer = new actuate.ReportExplorer("explorerpane");
```

The "explorerpane" parameter is the name value for the <div> element which holds the report explorer content. The page body must contain a <div> element with the id explorerpane as shown in the following code:

```
<div id="explorerpane"></div>
```

Use setFolderName( ) to set the directory to display in the explorer, as shown in the following code:

```
explorer.setFolderName("/public");
```

SetFolderName( ) accepts a single parameter, which is the path and name of a directory in the repository. In this example, "/public" indicates the /public directory.

ReportExplorer requires a results definition in order to retrieve data from the repository. The setResultDef( ) accepts an array of strings to define the results definition, as shown in the following code:

```
var resultDef = "Name|FileType|Version|VersionName|Description";
explorer.setResultDef( resultDef.split("|") );
```

The valid string values for the results definition array are "Name", "FileType", "Version", "VersionName", "Description", "Timestamp", "Size", and "PageCount", which correspond to file attributes loaded by ReportExplorer as it displays repository contents.

Call reportexplorer.submit( ) to make the page display the report explorer, as shown in the following code:

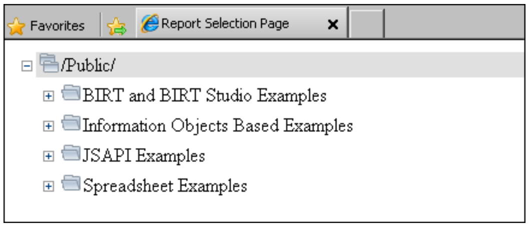
```
explorer.submit( );
```

The submit( ) function submits all the asynchronous operations that previous ReportExplorer functions prepare and triggers an AJAX request for the file information. The Actuate web application returns the list according to the results definition and the page displays the report explorer in the assigned <div> element.

This is a complete example of constructing actuate.ReportExplorer( ) in a callback function to display repository contents:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
   <meta http-equiv="content-type" content="text
      /html;charset=utf-8" />
   <title>Report Explorer Page</title>
</head>
<body onload="init( )">
<div id="explorerpane">
   <script type="text/javascript" language="JavaScript"
      src="http://localhost:8700/iportal/jsapi"></script>

   <script type="text/javascript" language="JavaScript">

   function init( ) {
      actuate.load("reportexplorer");
      requestOpts = new actuate.RequestOptions( );
      actuate.initialize( "http://localhost:8900/iportal",
         requestOpts, null, null, runReportExplorer);
   }

   function runReportExplorer( ) {
      var explorer = new actuate.ReportExplorer("explorerpane");
      explorer.setFolderName( "/Public" );
      var resultDef =
         "Name|FileType|Version|VersionName|Description";
      explorer.setResultDef( resultDef.split("|") );
      explorer.submit( );
   }
   </script>
</div>
</body>
</html>
```

The report explorer component displays the contents of the set folder, as shown in Figure 1-2.



/Public/
BIRT and BIRT Studio Examples
Information Objects Based Examples
JSAPI Examples
Spreadsheet Examples

Figure 1-2 Report Explorer page

Use the mouse or arrow keys to navigate the repository tree and expand folders to view their contents.

Opening files from ReportExplorer

The ReportExplorer class generates an actuate.reportexplorer.eventconstants.ON_SELECTION_CHANGED event when the user selects a folder or file in the Report Explorer User Interface. To access the file information in this event, implement an event handler like the one shown in the following code:

```
var file;
...
explorer.registerEventHandler(
   actuate.reportexplorer.EventConstants.ON_SELECTION_CHANGED,
   selectionChanged );
...
function selectionChanged( selectedItem, pathName ){
   file = pathName;
}
```

The event passes the path and name of the file in the second parameter of the handler, pathName. To access the file, the event handler stores the path in a global variable, file.

In this implementation, the file path is updated each time a file selected. To open the file currently selected, implement a button on the page that runs a separate function that opens the file. The following code example shows a button that calls the custom displayReport( ) function, which attempts to open the file using an actuate.viewer object:

```
<input type="button" style="width: 150pt;" value="View Report"
   onclick="javascript:displayReport( )"/>
...
function displayReport( ){
   var viewer = new actuate.Viewer("explorerpane");
   try {
      viewer.setReportName(file);
      viewer.submit( );
   } catch (e) {
      alert("Selected file is not viewable: " + file);
      runReportExplorer( );
   }
}
```

The try-catch block returns to the report explorer if Viewer is unable to open the file.

This is a complete example of a ReportExplorer page that opens a file in the BIRT Viewer when the user activates the View Report button:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
   <meta http-equiv="content-type" content="text
      /html;charset=utf-8" />
   <title>Report Explorer Page</title>
</head>
<body onload="init( )">
<input type="button" style="width: 150pt;" value="View Report"
   onclick="javascript:displayReport( )"/>
<hr />
<div id="explorerpane">
   <script type="text/javascript" language="JavaScript"
      src="http://localhost:8700/iportal/jsapi"></script>

   <script type="text/javascript" language="JavaScript">

   var file = "unknown";

   function init( ) {
      actuate.load("reportexplorer");
      actuate.load("viewer");
      actuate.load("dialog");
      requestOpts = new actuate.RequestOptions( );
      actuate.initialize( "http://localhost:8900/iportal",
         requestOpts, null, null, runReportExplorer);
   }

   function runReportExplorer( ) {
      var explorer = new actuate.ReportExplorer("explorerpane");
      explorer.registerEventHandler( actuate.reportexplorer.
         EventConstants.ON_SELECTION_CHANGED, selectionChanged );
      explorer.setFolderName( "/Public" );
      var resultDef =
         "Name|FileType|Version|VersionName|Description";
      explorer.setResultDef( resultDef.split("|") );
      explorer.submit( );
   }

   function selectionChanged( selectedItem, pathName ){
      file = pathName;
   }

   function displayReport( ){
      var y = document.getElementById('explorerpane'), child;
      while(child=y.firstChild){
         y.removeChild(child);
      }
      var viewer = new actuate.Viewer("explorerpane");
      try {
         viewer.setReportName(file);
         viewer.submit( );
      } catch (e) {
         alert("Selected file is not viewable: " + file);
         runReportExplorer( );
      }
   }
   </script>
</div>
</body>
</html>
```

Using dashboards and gadgets

The actuate.Dashboard class loads and displays dashboards and provides access to the gadgets contained in dashboards. To use actuate.Dashboard, load the class with actuate.load( ), as shown in the following code:

```
actuate.load("dashboard");
```

Load support for dialog boxes from the Actuate JavaScript API using actuate.load( ), as shown in the following code:

```
actuate.load("dialog");
```

Load the dashboard and dialog components to use the dashboard on the page. Call the actuate.Dashboard functions to prepare a dashboard and call the dashboard's submit( ) function to display the contents in the assigned <div> element.

The actuate.Dashboard class is a container for a dashboard file. Create an instance of the class using JavaScript, as shown in the following code:

```
dashboard = new actuate.Dashboard("containerID");
```

The value of "containerID" is the name value for the <div> element that displays the dashboard content. The page body must contain a <div> element with the containerID id, as shown in the following code:

```
<div id="containerID"></div>
```

To set the dashboard file to display, use setDashboardName( ) as shown in the following code:

```
dashboard.setDashboardName("/sharedtab.dashboard");
```

The setReportName( ) function accepts the path and name of a report file in the repository as the only parameter. In this example, "/public /customerlist.rptdesign" indicates the Customer List report design in the /public directory.

To display the dashboard, call dashboard.submit( ) as shown in the following code:

```
dashboard.submit(submitCallback);
```

The submit( ) function submits all of the asynchronous operations that previous viewer functions prepare and triggers an AJAX request for the dashboard. The Actuate web application returns the dashboard and the page displays the dashboard in the assigned <div> element. The submitCallback( ) callback function triggers after the submit operation completes.

This is a complete example that displays a dashboard:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
   <meta http-equiv="content-type" content="text
      /html;charset=utf-8" />
   <title>Dashboard Page</title>
</head>
<body onload="init( )">
<div id="dashboardpane">
   <script type="text/javascript" language="JavaScript"
      src="http://localhost:8700/iportal/jsapi"></script>

   <script type="text/javascript" language="JavaScript">
   function init( ){
      actuate.load("dashboard");
      actuate.initialize( "http://localhost:8900/iportal", null,
            null, null, runDashboard);
   }  function runDashboard( ){
      var dash = new actuate.Dashboard("dashboardpane");
      dash.setDashboardName("/Dashboard/Contents
               /Documents.DASHBOARD");
      dash.setIsDesigner(false);
      dash.submit( );
   }
   </script>
</div>
</body>
</html>
```

Using and submitting report parameters

Use the actuate.Viewer class to run report design and executable files. When a report design or executable runs, actuate.Viewer accepts parameters that modify the report output.

The actuate.Parameter class handles parameters and parameter values. The actuate.Parameter class enables a web page to display and gather parameters from users before processing and downloading a report to the client. Load the actuate.Parameter class with actuate.load( ), as shown in the following code:

```
actuate.load("parameter");
```

Load the parameter component to use it later in the page. Call actuate.Parameters functions to prepare a parameters page, display the parameters in the assigned <div> element, and assign the parameters to the viewer object for processing.

Using a parameter component

The actuate.Parameter class is a container for Actuate report parameters. Create an instance of the actuate.Parameter class using JavaScript, as shown in the following code:

```
var myParameters = new actuate.Parameter( "param1" );
```

The value of the "param1" parameter is the name value for the <div> element that holds the report parameters display. The page body must contain a <div> element with the param1 id, as shown in the following code:

```
<div id="param1"></div>
```

Use setReportName( ) to set the report from which to retrieve parameters, as shown in the following code:

```
myParameters.setReportName("/public/customerlist.rptdesign");
```

The setReportName( ) function takes the path and name of a report file in the repository as the only parameter. In this example, "/public /customerlist.rptdesign" indicates the Customer List report design in the /public directory.

To download the parameters and display them in a form on the page, call parameter.submit( ), as shown in the following code:

```
myParameters.submit(processParameters);
```

The submit( ) function submits all of the asynchronous operations prepared by the calls to parameter functions. The submit function also triggers an AJAX request to download the report parameters to the client. The Actuate web application sends the requested report parameters and the page displays them as a form in the assigned <div> element. The submit( ) function takes a callback function as a parameter, shown above as processParameters.

The following code example calls parameter in the callback function for actuate.initialize( ) to display a parameter:

```
<div id="param1">
   <script type="text/javascript" language="JavaScript"
      src="http://localhost:8700/iportal/jsapi"></script>

   <script type="text/javascript" language="JavaScript">
   function init( ){
      actuate.load("viewer");
      actuate.load("parameter");
      actuate.initialize( "http://localhost:8900/iportal", null,
         null, null, displayParams);
   }
   function displayParams( ) {
      param = new actuate.Parameter("param1");
      param.setReportName("/Public/BIRT and BIRT Studio
         Examples/Customer Order History.rptdesign");
      param.submit(function ( ) { this.run.style.visibility=
         'visible';});
   }function processParameters( ) {
      …
   }
</script></div>
```

The parameter component displays all of the parameters of the report in a form. When the parameters page is larger than the size of the viewer, the viewer provides scroll bars to navigate the parameters page.

To retrieve the parameters, use actuate.Parameter.downloadParameterValues( ). This function takes a callback function as an input parameter. The callback function processes the parameter values, as shown in the following code:

```
function processParameters( ) {
   myParameters.downloadParameterValues(runReport);
}
```

The downloadParameterValues( ) function requires the callback function to accept an array of parameter name and value pairs. The API formats this array properly for the actuate.Viewer class.

Accessing parameter values from the viewer

The actuate.Viewer.setParameterValues( ) function adds the parameters set by the user to the viewer component. The setParameterValues( ) function takes as an input parameter an object composed of variables whose names correspond to parameter names. The downloadParameterValues( ) function returns a properly formatted object for use with actuate.Viewer.setParameterValues( ). The following code example shows how to call downloadParameterValues( ) and move the parameter name and value array into the viewer with actuate.Viewer.setParameterValues( ):

```
function runReport(ParameterValues){
   var viewer = new actuate.Viewer("viewerpane");
   viewer.setReportName("/Public/BIRT and BIRT Studio
      Examples/Customer Order History.rptdesign");
   viewer.setParameterValues(ParameterValues);
   viewer.submit( );
}
```

When the viewer calls submit( ), the client transfers the parameters to the server with the other asynchronous operations for the viewer.

The following code example shows a custom web page that displays parameters and then shows the report in a viewer using those parameters:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
   <meta http-equiv="content-type" content="text/
      html;charset=utf-8" />
   <title>Viewer With Parameters Page</title>
</head>
<body onload="init( )">
   <div id="parampane">
   <script type="text/javascript" language="JavaScript"
      src="http://localhost:8700/iportal/jsapi"></script>
   <script type="text/javascript" language="JavaScript">
   function init( ){
      actuate.load("viewer");
      actuate.load("parameter");
      actuate.initialize( "http://localhost:8700/iportal", null,
         null, null, displayParams);
   }
   function displayParams( ) {
      param = new actuate.Parameter("parampane");
      param.setReportName("/Public/BIRT and BIRT Studio
         Examples/Customer Order History.rptdesign");
      param.submit(
            function ( ) {this.run.style.visibility = 'visible';});
   }
   function processParameters( ) {
      param.downloadParameterValues(runReport);
   }
   </script>
   </div>
   <hr><br />
   <input type="button" class="btn" name="run"
      value="Run Report" onclick="processParameters( )"
      style="visibility: hidden">

   <div id="viewerpane">
   <script type="text/javascript" language="JavaScript"
   src="http://localhost:8700/iportal/jsapi"></script>
   <script type="text/javascript" language="JavaScript">
   function runReport(paramvalues) {
      var viewer = new actuate.Viewer("viewerpane");
      viewer.setReportName("/Public/BIRT and BIRT Studio
         Examples/Customer Order History.rptdesign");
      viewer.setParameterValues(paramvalues);
      viewer.submit( );
   }
   </script>
   </div>
</body>
</html>
```

The code in the example uses the administrator user credentials and the default report installed with a standard installation of Information Console. The default report is at the following path:

```
/Public/BIRT and BIRT Studio Examples/Customer Order
   History.rptdesign
```

Retrieving report content as data

To retrieve report content as data, use the actuate.DataService class from the Actuate JavaScript API. The DataService is packaged with the actuate.Viewer class. Load the actuate.DataService class with actuate.load( ), as shown in the following code:

```
actuate.load("viewer");
```

Load support for dialog boxes from the Actuate JavaScript API with actuate.load( ), as shown in the following code:

```
actuate.load("dialog");
```

Load the viewer and dialog components to use data services on the page. Call the functions in the actuate.DataService class to prepare report data, then call downloadResultSet( ) from the DataService class to obtain the report data.

Using a data service component

The actuate.DataService class is a container for Actuate report data. Create an instance of the class with JavaScript, as shown in the following code:

```
var dataservice = new actuate.DataService( );
```

Without parameters, the actuate.DataService class uses the Actuate web application service called in actuate.intialize.

To gather data from a report, define a request and send the request to the Actuate web application service for the data. The actuate.data.Request object defines a request. To construct the Request object, use the actuate.data.Request constructor, as shown below:

```
var request = new actuate.data.Request(bookmark, start, end);
```

where

- bookmark is a bookmark that identifies an Actuate report element. The actuate.data.Request object uses the bookmark to identify the report element from which to request information. If bookmark is null, the actuate.data.Request object uses the first bookmark in the report.
- start is the numerical index of the first row to request. The smallest valid value is 1.
- end is the numerical index of the last row to request. A value of 0 indicates all available rows.

To download the data, use dataservice.downloadResultSet( ), as shown in the following code:

```
dataservice.downloadResultSet(filedatasource, request,
   displayData, processError);
```

where

- filedatasource is the path and name of a report file in the repository. For example, "/public/customerlist.rptdesign" indicates the Customer List report design in the /public directory. The dataservice.downloadResultSet( ) function uses the Actuate web application service set with actuate.Initialize( ) by default.
- request is an actuate.data.Request object that contains the details that are sent to the server in order to obtain specific report data.
- displayData is a callback function to perform an action with the downloaded data. This callback function takes an actuate.data.ResultSet object as an input parameter.
- processError is a callback function to use when an exception occurs. This callback function takes an actuate.Exception object as an input parameter.

JSAPI DataService cannot download ResultSets from BIRT report elements with an automatically generated bookmark. When designing a report, report developers can explicitly specify bookmarks for report elements. If a bookmark is not specified, the report generates a generic bookmark name automatically when it executes. The JSAPI DataService class cannot retrieve a result set from these generic bookmarks. To use the JSAPI DataService on a bookmark, the report developer must specify a name value for the bookmark.

To provide a quick alert displaying the column headers for the retrieved data set, use code similar to the following:

```
alert("Column Headers: " + myResultSet.getColumnNames());
```

where myResultSet is the ResultSet object retrieved by downloadResultSet.

Using a result set component

The actuate.data.ResultSet class is the container for the report data obtained with actuate.dataservice.downloadResultSet( ). Because a ResultSet object is not a display element, an application can process or display the data in an arbitrary fashion.

The ResultSet class organizes report data into columns and rows, and maintains an internal address for the current row. To increment through the rows, use the ResultSet's next( ) function as shown in the following code:

```
function displayData(rs)
{
…
   while (rs.next( ))
…
```

In this example, rs is the ResultSet object passed to the displayData callback function. To read the contents of the ResultSet object, a while loop increments through the rows of data with rs.next( ).

Because a web page that loads a DataService object also loads initiates the viewer, the target for displaying a result set must be a separate page or application.

Chapter

2

Creating dynamic report content using the Actuate JavaScript API

This chapter contains the following topics:

- About Actuate JavaScript API scripting in a BIRT report design
- Using the Actuate JavaScript API in an HTML button
- Using the Actuate JavaScript API in chart interactive features
- Using the Actuate JavaScript API in chart themes
- Using the Actuate JavaScript API in the BIRT script editor

About Actuate JavaScript API scripting in a BIRT report design

The scripting features of the BIRT designers support using the JSAPI for the following operations:

- Using the Actuate JavaScript API in an HTML button
- Using the Actuate JavaScript API in chart interactive features
- Using the Actuate JavaScript API in the BIRT script editor

Most Actuate JavaScript API functions run when an event occurs. The report element defines the events that it supports. For example, the onRender event occurs when the report renders in the viewer or on a page.

A BIRT report or Reportlet renders in the following ways:

- In BIRT Viewer or Interactive Viewer
- In BIRT Studio
- In Actuate BIRT Designer
- In an Actuate JavaScript API viewer object on a mashup page

All of these products load the actuate.Viewer and actuate.Dialog classes when they render a report, except for the preview functionality in BIRT Designer. Use the View Report in Web Viewer function to view and test Actuate JavaScript API scripts with BIRT Designer, as shown in Figure 2-1.

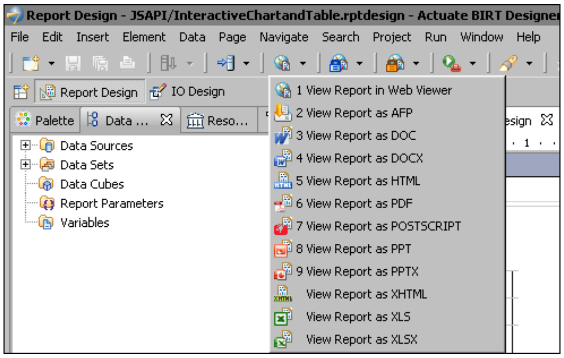
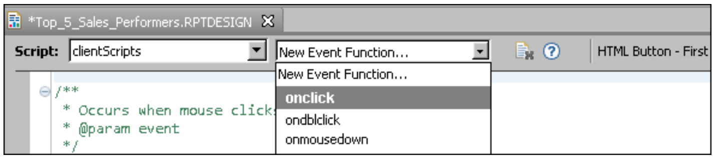


Figure 2-1 Accessing Web Viewer in Actuate BIRT Designer

Most of the classes and functions in the actuate.Viewer class can be used in a BIRT report without loading or initializing the actuate.Viewer class and dialog support. Most of the viewers also load the actuate.Parameters and actuate.DataService classes by default. Define the classes loaded for Actuate JavaScript API mashup page explicitly. Load the DataService, Parameters, and Viewer classes before the API initializes the connection to the reporting web service.

Using the Actuate JavaScript API in an HTML button

The HTML button element can execute client-side JavaScript code based on button events. Access the HTML button in the BIRT designer by selecting a button element, choosing the script tag, and selecting the event from the event drop-down list, as shown in Figure 2-2.



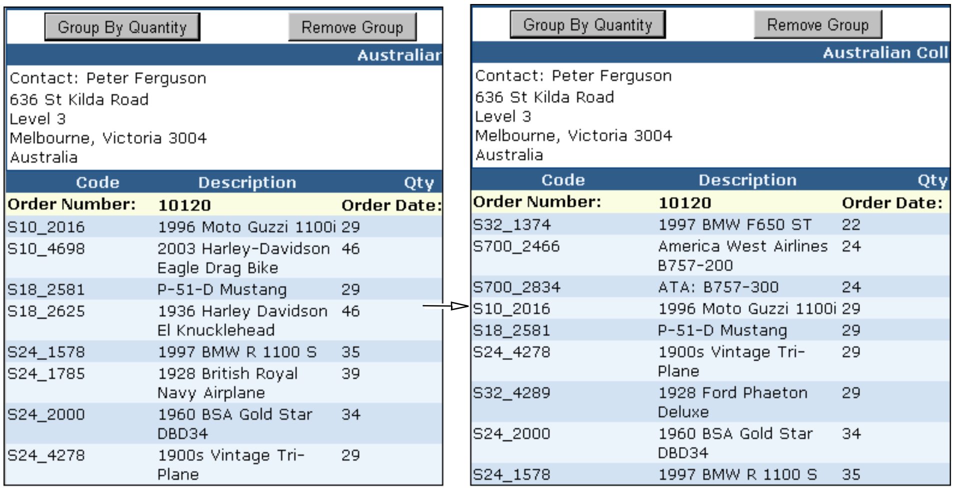
```
/**
 * Occurs when mouse click
 * @param event
 */
```

Figure 2-2 Choosing HTML button event handlers

Use event functions to add JavaScript functionality to HTML buttons. For example, a button that swaps columns of data, filters data, sorts data, hides information, or groups the rows of a table by some data column can be created with event functions. The following script groups the rows of a table by the quantity of items in each order when the HTML button is clicked:

```
this.onclick = function(event){
   var btable = this.getViewer( ).getCurrentPageContent( ).
      getTableByBookmark("TableBookmark");
   btable.groupBy("QUANTITYORDERED");
   btable.submit( );
}
```

When the HTML button triggers the example event above, the table grouping changes and the display refreshes, as shown in Figure 2-3.



Contact: Peter Ferguson
636 St Kilda Road
Level 3
Melbourne, Victoria 3004
Australia

Code	Description	Qty
Order Number:	10120	Order Date:
S10_2016	1996 Moto Guzzi 1100i	29
S10_4698	2003 Harley-Davidson Eagle Drag Bike	46
S18_2581	P-51-D Mustang	29
S18_2625	1936 Harley Davidson El Knucklehead	46
S24_1578	1997 BMW R 1100 S	35
S24_1785	1928 British Royal Navy Airplane	39
S24_2000	1960 BSA Gold Star DBD34	34
S24_4278	1900s Vintage Tri-Plane	29

Contact: Peter Ferguson
636 St Kilda Road
Level 3
Melbourne, Victoria 3004
Australia

Code	Description	Qty
Order Number:	10120	Order Date:
S32_1374	1997 BMW F650 ST	22
S700_2466	America West Airlines B757-200	24
S700_2834	ATA: B757-300	24
S10_2016	1996 Moto Guzzi 1100i	29
S18_2581	P-51-D Mustang	29
S24_4278	1900s Vintage Tri-Plane	29
S32_4289	1928 Ford Phaeton Deluxe	29
S24_2000	1960 BSA Gold Star DBD34	34
S24_1578	1997 BMW R 1100 S	35

Figure 2-3 Using a GroupBy HTMLButton control

HTML buttons can be arranged into sets of controls for the user to use once a report runs. For example, when these buttons are used in the header for a table, the header can provide controls similar to those in the header shown in Figure 2-4.

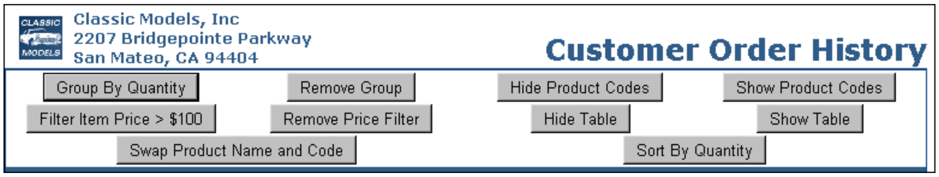


Figure 2-4 HTML button header

Using the Actuate JavaScript API in chart interactive features

BIRT reports support adding interactive features to a chart to enhance the behavior of a chart in the viewer. The interactive chart features are available through the chart builder. Implement Actuate JavaScript API functions within interactive features.

An interactive chart feature supports a response to an event, such as the report user choosing an item or moving the mouse pointer over an item. The response can trigger an action, such as opening a web page, drilling to a detail report, or changing the appearance of the chart. For example, use a tooltip to display the series total when a user places the mouse over a bar in a bar chart, as shown in Figure 2-5.

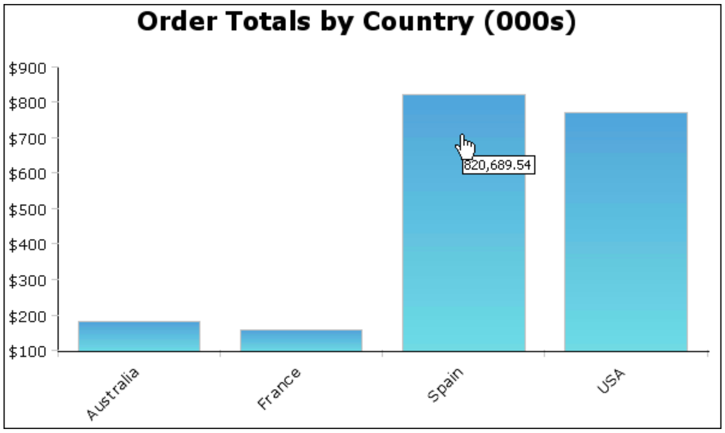


Figure 2-5 Chart showing a tooltip

Interactive features can be added to a value series, the chart area, a legend, marker lines, the *x*- and *y*-axis, or a title. Figure 2-6 identifies these elements.

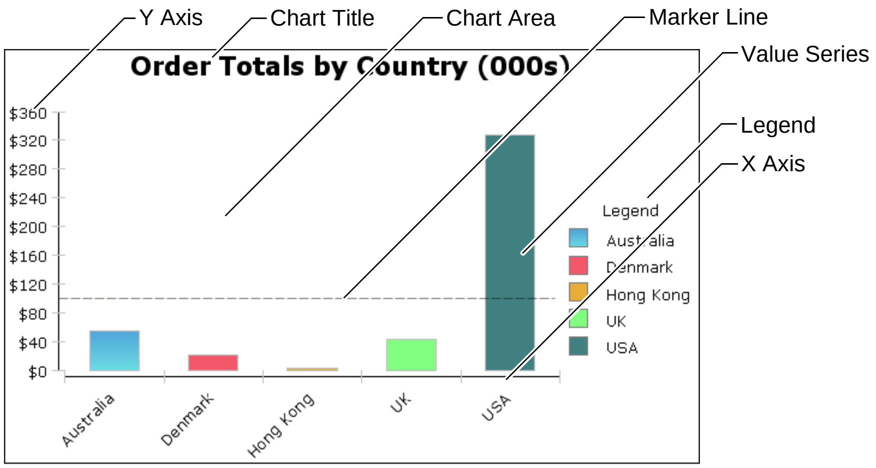


Figure 2-6 Elements selectable for chart interactivity

To add an interactive feature to a chart, either choose Format Chart in the chart builder and select a chart element to make interactive, or choose Script in the chart builder and select the chart element to make interactive. Figure 2-7 shows the location of the Interactivity button for a value series.

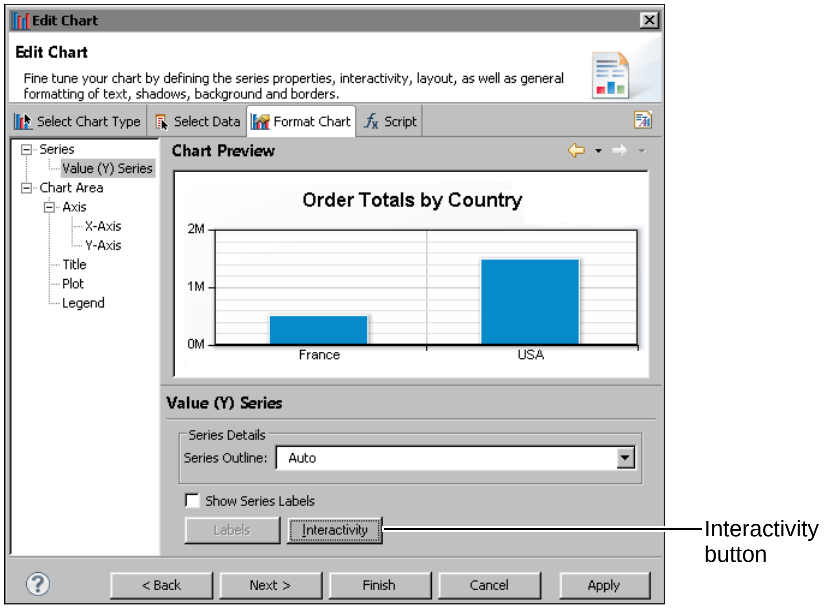
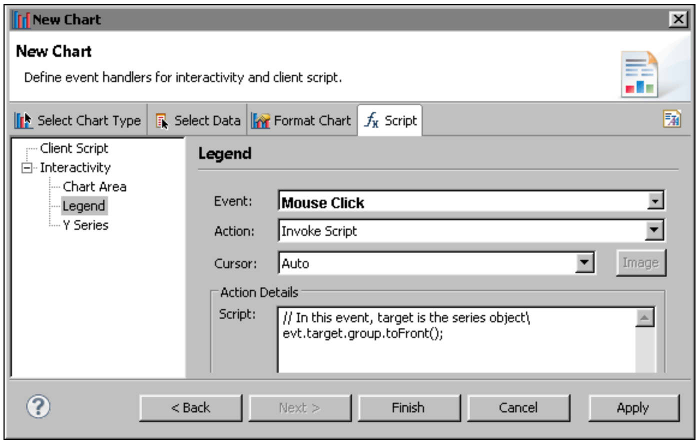


Figure 2-7 Accessing interactivity for a value series

Figure 2-8 shows the elements accessible using the script feature.



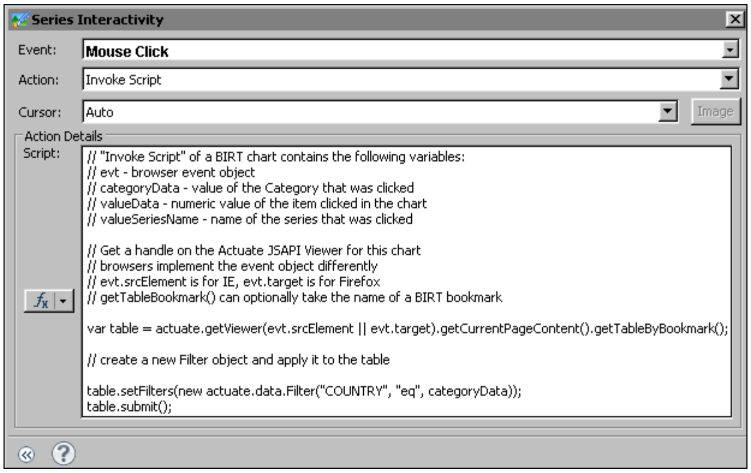
Legend

Event: Mouse Click
Action: Invoke Script
Cursor: Auto
Action Details
Script:

```
// In this event, target is the series object\
evt.target.group.toFront();
```

Figure 2-8 Accessing interactivity for a legend

The location of the Interactivity button varies by chart element. Click the Interactivity button to display the interactivity editor. Figure 2-9 shows the interactivity editor.



Event: Mouse Click
Action: Invoke Script
Cursor: Auto
Action Details
Script:

```
// "Invoke Script" of a BIRT chart contains the following variables:
// evt - browser event object
// categoryData - value of the Category that was clicked
// valueData - numeric value of the item clicked in the chart
// valueSeriesName - name of the series that was clicked

// Get a handle on the Actuate JSAPI Viewer for this chart
// browsers implement the event object differently
// evt.srcElement is for IE, evt.target is for Firefox
// getTableBookmark() can optionally take the name of a BIRT bookmark

var table = actuate.getViewer(evt.srcElement || evt.target).getCurrentPageContent().getTableByBookmark();

// create a new Filter object and apply it to the table

table.setFilters(new actuate.data.Filter("COUNTRY", "eq", categoryData));
table.submit();
```

Figure 2-9 Interactivity editor

The Action Details window displays a script that runs when the user clicks an item in the series. The script adds a filter to the table that displays below the chart. The filter restricts the data by the selected element. The code performs the following three tasks to handle this interactivity:

- Obtains the bookmark for the table when the event occurs:

```
var table = actuate.getViewer(evt.srcElement ||
   evt.target).getCurrentPageContent( ).getTableByBookmark( );
```

The event is taken from the Invoke Script action of a BIRT chart. Set the Invoke Script action in the second field of the interactivity editor. The Invoke Script action contains the following variables:

- evt: browser event object
- categoryData: value of the selected category
- valueData: numeric value of the selected item
- valueSeriesName: name of the selected series

The code above uses getViewer and the evt object to obtain a handle for the viewer when an event occurs. The Firefox and Internet Explorer browsers implement the event differently. For Firefox, evt.target contains the name of the viewer object. For Internet Explorer, evt.srcElement contains the name of the viewer object.

The getCurrentPageContent.getTableByBookmark( ) function retrieves the Table object for the first table in the viewer. To target a different table, use a specific table bookmark as the input parameter for getTableByBookmark( ).

- Performs an operation on the target:

```
table.setFilters(new actuate.data.Filter("COUNTRY", "eq",
   categoryData));
```

This code example creates a new filter using the actuate.data.Filter constructor. The constructor takes three arguments:

- column name: The column name is the name of the series. In this case, the *y*-axis is a list of countries, so a mouse click filters the table according to the COUNTRY column.
- operator: eq is the reserved operator for equal to.
- value: the value of the categoryData object generated by the event, which is a country. The filter returns rows with a COUNTRY value that matches the value selected by the user.

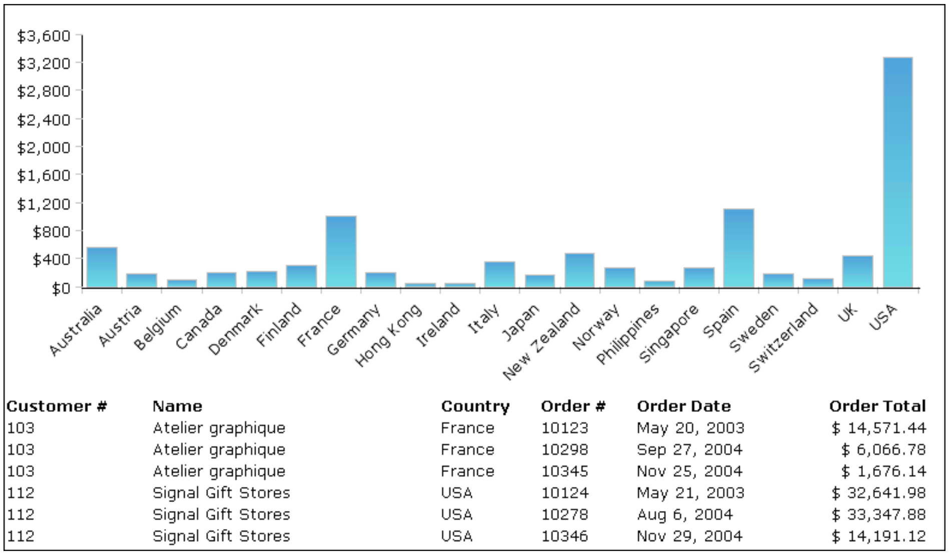
- Submits the action for processing:

```
table.submit( );
```

The Actuate JavaScript API processes operations asynchronously. Actions are performed when submit( ) is called.

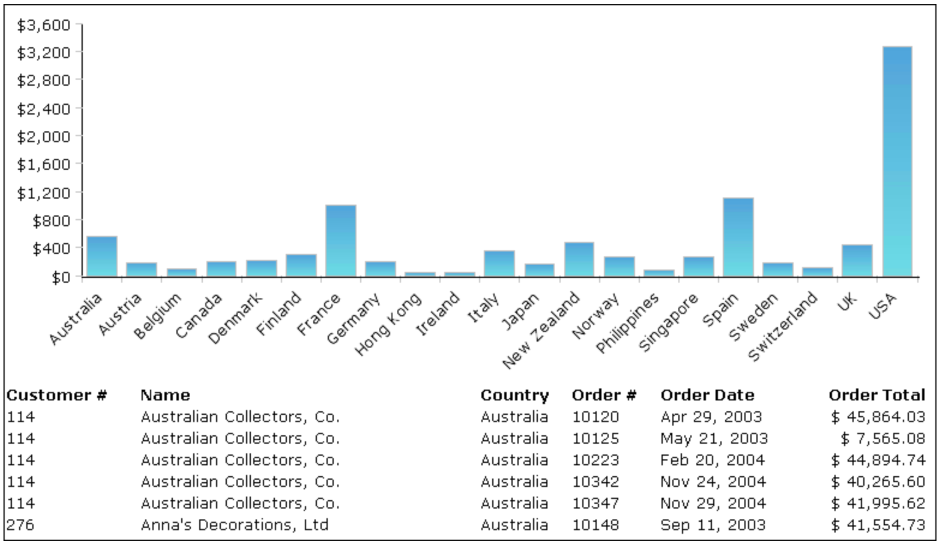
Figure 2-10 shows the chart before interaction.

When the user selects the bar for Australia in the value series, the table is filtered for Australia, as shown in Figure 2-11.



Customer #	Name	Country	Order #	Order Date	Order Total
103	Atelier graphique	France	10123	May 20, 2003	$ 14,571.44
103	Atelier graphique	France	10298	Sep 27, 2004	$ 6,066.78
103	Atelier graphique	France	10345	Nov 25, 2004	$ 1,676.14
112	Signal Gift Stores	USA	10124	May 21, 2003	$ 32,641.98
112	Signal Gift Stores	USA	10278	Aug 6, 2004	$ 33,347.88
112	Signal Gift Stores	USA	10346	Nov 29, 2004	$ 14,191.12

Figure 2-10 An interactive chart and table before any user action



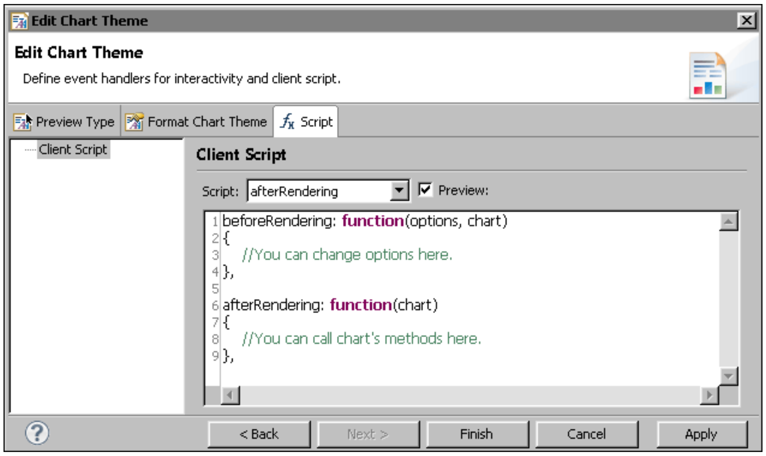
Customer #	Name	Country	Order #	Order Date	Order Total
114	Australian Collectors, Co.	Australia	10120	Apr 29, 2003	$ 45,864.03
114	Australian Collectors, Co.	Australia	10125	May 21, 2003	$ 7,565.08
114	Australian Collectors, Co.	Australia	10223	Feb 20, 2004	$ 44,894.74
114	Australian Collectors, Co.	Australia	10342	Nov 24, 2004	$ 40,265.60
114	Australian Collectors, Co.	Australia	10347	Nov 29, 2004	$ 41,995.62
276	Anna's Decorations, Ltd	Australia	10148	Sep 11, 2003	$ 41,554.73

Figure 2-11 An interactive chart and table after the user selects Australia

Using the Actuate JavaScript API in chart themes

BIRT reports support adding themes to a chart to apply common elements to similar charts. Access chart themes by exporting and then editing a theme or by creating a new theme. Implement Actuate JavaScript API functions within specific theme elements or in the script feature of the theme.

A chart theme supports executing a script before or after certain events, such as before rendering the chart. For example, you can add scripts for beforeGeneration, beforeRendering, beforeDrawAxis, beforeDrawSeries, beforeDrawDataPoint, and afterRendering when editing a chart theme, as shown in Figure 2-12.



Script: afterRendering ☑ Preview:

```
beforeRendering: function(options, chart)
{
   //You can change options here.
},

afterRendering: function(chart)
{
   //You can call chart's methods here.
},
```

Figure 2-12 Adding script elements in edit chart theme

In an HTML5 chart, you can use the actuate.report.HTML5Chart classes to alter the report display. For example, to render every data point in the series that is greater than avgValue in a green color, use code similar to the following:

```
beforeDrawSeries: function(series, seriesOptions, tempChart,
   seriesIndex){
   for ( var i = 0; i < series.data.length; i++ ){
      // Find out if this data point is above average
      if ( series.data[i].y <= aveValue ){
         // The data point is above average.  Color it green
         var pointOptions = seriesOptions.data[i];
         pointOptions.color = 'green';
   }
}
```

Using the Actuate JavaScript API in the BIRT script editor

Use the Script tab to open the BIRT script editor in Actuate BIRT Designer Professional, as shown in Figure 2-13.

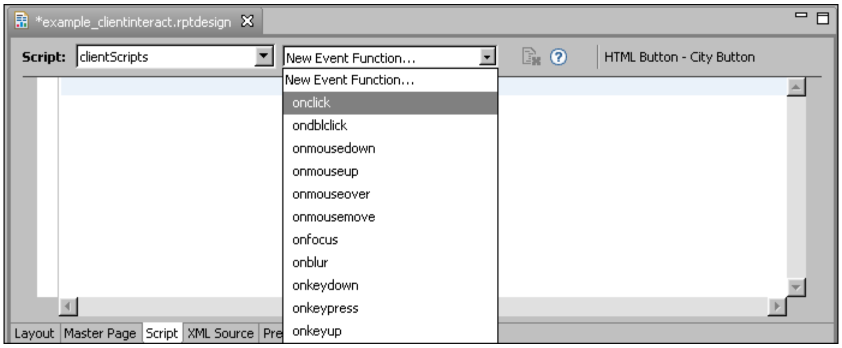


Figure 2-13 The BIRT Report Designer Professional script editor

Assign a Actuate JavaScript API script to run when a particular event occurs, such as onRender.

Chapter

3

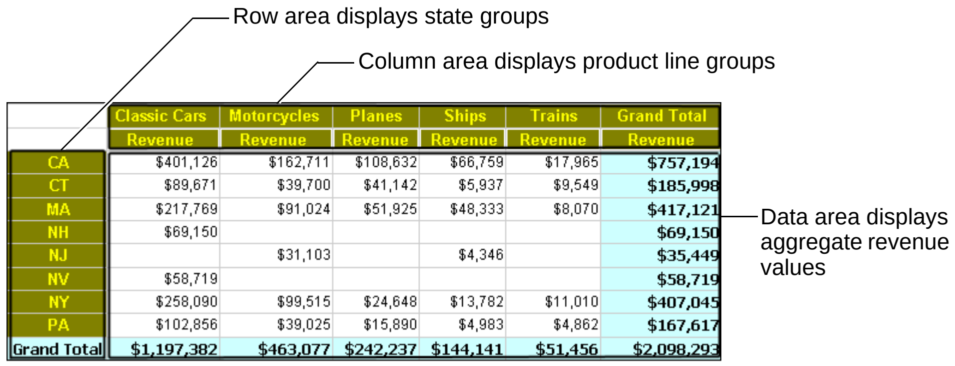
Working with BIRT Data Analyzer and cross tabs

This chapter contains the following topics:

About cross tabs

A cross tab, or cross tabulation, displays data in a row-and-column matrix similar to a spreadsheet. A cross tab is ideal for concisely summarizing data. A cross tab displays aggregate values such as averages, counts, or sums in the cross tab's cells.

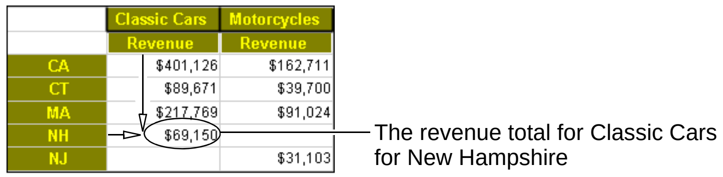
Figure 3-1 shows a cross tab that organizes state groups in the row area and product line groups in the column area. Aggregate revenue values appear in the cells of the data area.


Row area displays state groups
Column area displays product line groups
Data area displays aggregate revenue values

	Classic Cars	Motorcycles	Planes	Ships	Trains	Grand Total
	Revenue	Revenue	Revenue	Revenue	Revenue	Revenue
CA	$401,126	$162,711	$108,632	$66,759	$17,965	**$757,194**
CT	$89,671	$39,700	$41,142	$5,937	$9,549	**$185,998**
MA	$217,769	$91,024	$51,925	$48,333	$8,070	**$417,121**
NH	$69,150					**$69,150**
NJ		$31,103		$4,346		**$35,449**
NV	$58,719					**$58,719**
NY	$258,090	$99,515	$24,648	$13,782	$11,010	**$407,045**
PA	$102,856	$39,025	$15,890	$4,983	$4,862	**$167,617**
Grand Total	**$1,197,382**	**$463,077**	**$242,237**	**$144,141**	**$51,456**	**$2,098,293**

Figure 3-1 Viewing a cross tab

A cell displays a revenue value by product line and by state, as shown in Figure 3-2.



	Classic Cars	Motorcycles
	Revenue	Revenue
CA	$401,126	$162,711
CT	$89,671	$39,700
MA	$217,769	$91,024
NH	$69,150	
NJ		$31,103

Figure 3-2 A cell displaying a revenue total

A cross tab uses data from at least three fields. The cross tab in Figure 3-1 uses the following data fields:

- One field provides the values for column headings in the cross tab. The cross tab displays one column for each unique value in the field. In Figure 3-1, the cross tab displays five unique values from the productline field: Classic Cars, Motorcycles, Planes, Ships, and Trains.
- One field provides the values for row headings in the cross tab. The cross tab displays one row for each unique value in the field. In Figure 3-1, the cross tab displays eight unique values from the state field: CA, CT, MA, NH, NJ, NV, NY, and PA.

- BIRT Data Analyzer aggregates one field's values, and displays these values in the cross tab cells. In this example, each cell displays a revenue total by product line and state. Data Analyzer calculates the revenue total using the SUM function on the values in the extendedprice field.

About cubes

A cube is a multidimensional data structure that is optimized for analysis. A cube supports applications that perform complex analyses without performing additional queries on the underlying data source. A cube organizes data into the following categories:

- Measures

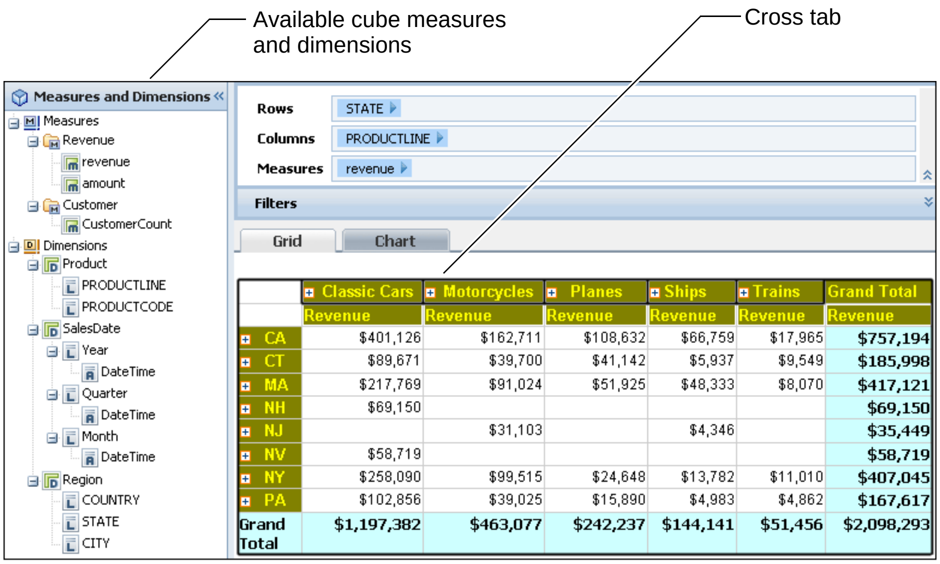
  Measures are aggregate, or summary, values, such as sales revenues or units of products.

- Dimensions

  Dimensions are groups, such as customers, product lines, or time periods, which aggregate measures. For example, a sales revenue cube contains data that enables viewing sales volume and revenues, both of which are measures, by customers, product lines, and time periods, all of which are dimensions.

Dimensions can contain levels, which organize data into hierarchies. For example, a region dimension can contain a hierarchy of the country, state, and city levels. A time dimension can contain a hierarchy of the year, quarter, month, and day levels. Cubes frequently include time dimensions because displaying measures by time dimensions is useful in data analysis. The time dimension in a cube is a special dimension that supports storing data in developer-defined time periods.

Use Actuate BIRT Designer Professional to create a cube using data from one or more data sources, then create a cross tab that uses the cube data and specifies the cross tab appearance. The initial cross tab that appears in the Data Analyzer typically displays a portion of the available cube data in a simple layout. Figure 3-3 shows a cross tab and all of the cube measures and dimensions that are available for analysis.



Measures
- Revenue
  - revenue
  - amount
- Customer
  - CustomerCount

Dimensions
- Product
  - PRODUCTLINE
  - PRODUCTCODE
- SalesDate
  - Year
    - DateTime
  - Quarter
    - DateTime
  - Month
    - DateTime
- Region
  - COUNTRY
  - STATE
  - CITY

Rows: STATE
Columns: PRODUCTLINE
Measures: revenue

	Classic Cars	Motorcycles	Planes	Ships	Trains	Grand Total
	Revenue	Revenue	Revenue	Revenue	Revenue	Revenue
CA	$401,126	$162,711	$108,632	$66,759	$17,965	$757,194
CT	$89,671	$39,700	$41,142	$5,937	$9,549	$185,998
MA	$217,769	$91,024	$51,925	$48,333	$8,070	$417,121
NH	$69,150					$69,150
NJ		$31,103		$4,346		$35,449
NV	$58,719					$58,719
NY	$258,090	$99,515	$24,648	$13,782	$11,010	$407,045
PA	$102,856	$39,025	$15,890	$4,983	$4,862	$167,617
Grand Total	$1,197,382	$463,077	$242,237	$144,141	$51,456	$2,098,293

Figure 3-3 Data Analyzer displaying a cross tab and available measures and dimensions

See *BIRT: A Field Guide* for more information about data cubes and cross tabs.

Handling Data Analyzer viewer events

The Data Analyzer viewer triggers events to indicate changes in status. These events include notifications of data changes or errors. Use the registerEventHandler function found in XTabAnalyzer to handle events, as shown in the following code:

```
ctViewer.registerEventHandler(actuate.xtabanalyzer.EventConstants.
                                                  ON_EXCEPTION,errorHandler);
```

This code registers the event handler errorHandler to be called when an ON_EXCEPTION event occurs.

The XTabAnalyzer class supports the following events:

- ON_CONTENT_CHANGED
- ON_CONTENT_SELECTED
- ON_EXCEPTION
- ON_SESSION_TIMEOUT

To remove an event handler, call removeEventHandler( ).

```
ctViewer.removeEventHandler(actuate.xtabanalyzer.EventConstants.
                             ON_EXCEPTION,errorHandler);
```

The actuate.xtabanalyzer.Exception class handles exceptions. For more information about events, see the section describing the actuate.xtabanalyzer.EventsConstants class.

Working with dimensions, measures, and levels

The actuate.xtabanalyzer.Crosstab class represents the cross tab element. Use this cross tab class when working with Data Analyzer and the XTabAnalyzer viewer. Use the functions in the actuate.xtabanalyzer.Dimension class to add, remove, or modify dimensions. Use the functions in the actuate.xtabanalyzer.Measure class to add, remove, or modify measures. Use the functions in the actuate.xtabanalyzer.Level class to add, remove, or modify levels. These classes contain functions that support the creation and modification of the dimensions, measures, and levels in the cross tab. These functions work with information from a data cube that is created with BIRT Designer Professional.

Adding a dimension with levels

To add a dimension to the cross tab, use Crosstab.addDimension( ) to add an actuate.xtabanalyzer.Dimension object to the cross tab. The following code requires that the dimensions and levels already exist within a data cube:

```
var crosstab = new actuate.xtabanalyzer.Crosstab( );
var dimension = new actuate.xtabanalyzer.Dimension( );

// Set dimension to be in the zero location.
dimension.setIndex(0);
dimension.setAxisType(actuate.xtabanalyzer.Dimension.
                       COLUMN_AXIS_TYPE);
dimension.setDimensionName("dates");
var level = new actuate.xtabanalyzer.Level( );
level.setLevelName("year");
dimension.addLevel(level);
var level = new actuate.xtabanalyzer.Level( );
level.setLevelName("quarter");
dimension.addLevel(level);
var level = new actuate.xtabanalyzer.Level( );
level.setLevelName("month");
dimension.addLevel(level);
crosstab.addDimension(dimension);
crosstab.submit( );
```

Removing a dimension

To remove a dimension from a cross tab, use Crosstab.removeDimension( ). In this example, levelNames is an array of strings containing the names of the levels to remove:

```
crosstab.removeDimension("dates",null,levelNames);
crosstab.submit( );
```

Adding and removing measures

To add a measure to the cross tab, use Crosstab.addMeasure( ). The addMeasure( ) function accepts an actuate.xtabanalyzer.Measure object as a parameter. This example creates a new measure and adds it to a cross tab:

```
var measure = new actuate.xtabanalyzer.Measure( );

measure.setIndex(1);
measure.setMeasureName("Quarter Rate");
measure.setExpression("[revenue]/[revenue_SalesDate/year_Product
                        /PRODUCTLINE]");
crosstab.addMeasure(measure);
crosstab.submit( );
```

The measure.setExpression( ) function dynamically sets the measure to display the revenue received for sales data, organized by year and product line. In this example, the expression is in easyscript. Easyscript is described in *Using Actuate BIRT Designer Professional*. The expression in the example is the database field that contains the sales revenue value. Data Analyzer aggregates the sales revenue value for each year for each product line. The [revenue_SalesDate/year_Product /PRODUCTLINE] string specifies that the expression applies to the revenue by salesdate and then by year for the product line.

The Actuate JavaScript API combined with standard JavaScript functionality enables the creation of web pages that allow for interactive manipulation of cross tabs. In this example, the measure name and the measure expression are retrieved from HTML elements with the names of measureName and measureExpression. As coded, these elements can be an item such as a text entry field. The values of any used elements then go into the new measure for the cross tab.

```
var measureName  = document.getElementById("measureName").value;
var measureExpression  =
   document.getElementById("measureExpression").value;

var measure = new actuate.xtabanalyzer.Measure( );
measure.setIndex(1);
measure.setMeasureName(measureName);
measure.setExpression(measureExpression);

crosstab.addMeasure(measure);
crosstab.submit( );
```

The web page must contain elements with the IDs of measureName and measureExpression. Use the following HTML code to create these elements:

```
<INPUT TYPE="text" SIZE="60" ID="measureName" VALUE="Quarter
   Rate">
<INPUT type="text" SIZE="60" ID="measureExpression"
   VALUE="[revenue]/[revenue_SalesDate/year_Product/PRODUCTLINE]">
```

Use removeMeasure( ) to remove a measure. Pass the name of the measure to remove to removeMeasure( ).

```
crosstab.removeMeasure("Quarter Rate");
crosstab.submit( );
```

Changing measures and dimensions

Edit measures with Crosstab.editMeasure( ). In this example, the measureName measure named measureName takes on a new value:

```
var measure = new actuate.xtabanalyzer.Measure( );

measure.setMeasureName("measureName");
measure.setExpression("measureExpression");
crosstab.editMeasure(measure);
crosstab.submit( );
```

Use Crosstab.changeMeasureDirection( ) to change the measure direction. Pivot the cross tab with Crosstab.pivot( ).

Use Crosstab.reorderDimension( ) to change the order or axis type of a dimension within a cross tab. This example moves the index of a dimension within a cross tab from 1 to 5. The dimension's axis type changes from a row axis to a column axis.

```
var dimIdx = 1;
var newDimIdx = 5
var axis = actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE;
var newAxis = actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE;
crosstab.reorderDimension(dimIdx,axis,newDimIdx,newAxis );
crosstab.submit( );
```

The measure placement order can be altered using Crosstab.reorderMeasure( ). In this example, a measure's index changes from position 1 in the cross tab to position 5:

```
crosstab.reorderMeasure(1,5);
crosstab.submit( );
```

Measures and dimensions can also be changed with the functions in the measure and dimension classes. In this example, a dimension axis changes from column to row:

```
var currentAxis = dimension.getAxisType( )
if (currentAxis ==
   actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE){
   dimension.setNewAxisType(
      actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE);
}
```

Working with totals

Each dimension within a cross tab and each level within a multilevel dimension can have a total associated with that dimension or level. A row or column with a single dimension can only have a grand total. Each level in a multilevel dimension can have a subtotal. Subtotals are only available for multilevel dimensions.

A total requires a measure and an aggregation function. To add a grand total to a measure, use the actuate.xtabanalyzer.GrandTotal class. Subtotals are added with the actuate.xtabanalyzer.SubTotal class. Both classes use the actuate.xtabanalyzer.Total class. The Total class supports creating aggregated values on a measure, calculated on either a row or a column. This example creates a total and places the SUM aggregation function on the measure located at measure index 0:

```
var grandTotal = new actuate.xtabanalyzer.GrandTotal( );
grandTotal.setAxisType(actuate.xtabanalyzer.Dimension.
                        ROW_AXIS_TYPE );
// Create a total object containing a measure and aggregation.
var total = new actuate.xtabanalyzer.Total( );
total.setMeasureIndex(0);
total.setAggregationFunction("SUM");
total.setEnabled(true);

// Add the total to the cross tab.
grandTotal.addTotal(total);
crosstab.setTotals(grandTotal);
crosstab.submit( );
```

The actuate.xtabanalyzer.Total class uses a measure index and an aggregation function to create a Total object that is added to a SubTotal or GrandTotal object for placement within the cross tab. A total must be enabled for that total to be active on the cross tab.

To remove a total from a cross tab, use setEnabled( ) and pass false as a parameter, as shown in the following code:

```
total.setEnabled(false);
grandTotal.addTotal(total);
crosstab.setTotals(grandTotal);
crosstab.submit( );
```

Sorting and filtering cross tab data

Data within levels can be filtered and sorted. To sort data within a level, use the actuate.xtabanalyzer.Sorter class. Add an instance of the Sorter class to the cross tab with Crosstab.setSorters( ).

```
var sorter = new actuate.xtabanalyzer.Sorter("sortLevelName");
sorter.setAscending(false);

// Add the sort to the cross tab.
crosstab.setSorters(sorter);
crosstab.submit( );
```

Use the actuate.xtabanalyzer.Filter class to filter data within a level. A filter requires an operator and values to filter. Use Crosstab.setFilters( ) to place the filter within the cross tab.

```
var filter = new actuate.xtabanalyzer.Filter
                ("levelName","BETWEEN");

// Filter between the values of 1000 and 2000.
var filterValue = "1000;2000";
filter.setValues(filterValue.split(";"));
crosstab.setFilters(filter);
crosstab.submit( );
```

To remove a filter from a level, use actuate.xtabanalyzer.Crosstab.clearFilters( ).

```
crosstab.clearFilters("levelName");
crosstab.submit( );
```

Drilling down within a cross tab

Drilling supports the ability to expand or collapse a member value within a specific level. Construct a XTabAnalyzer.Driller object as shown in the following code:

```
var driller = new actuate.xtabanalyzer.Driller( );
```

To drill up or down, use actuate.xtabanalyzer.Crosstab.drill( ) with the actuate.xtabanalyzer.Driller and actuate.xtabanalyzer.MemberValue classes. In this example, a cross tab has a dimension named Region with three levels:

Country, State, and City. The actuate.xtabanalyzer.Driller object updates the cross tab to display the requested information, as shown in the following code:

```
driller.setAxisType(
   actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE);
var levelName1 = "Region/Country";
var levelValue1 = "Australia";
var levelName2 = "Region/State";
var levelValue2 = "NSW";

// Create member value objects, and place them in the driller.
var memberValue1 = new
   actuate.xtabanalyzer.MemberValue(levelName1);
memberValue1.setValue(levelValue1);
var memberValue2 = new
   actuate.xtabanalyzer.MemberValue(levelName2);
memberValue2.setValue(levelValue2);
memberValue1.addMember(memberValue2);
driller.addMember(memberValue1);

crosstab.drill(driller);
crosstab.submit( );
```

To reset the drill, use a Driller object with no level names or member values.

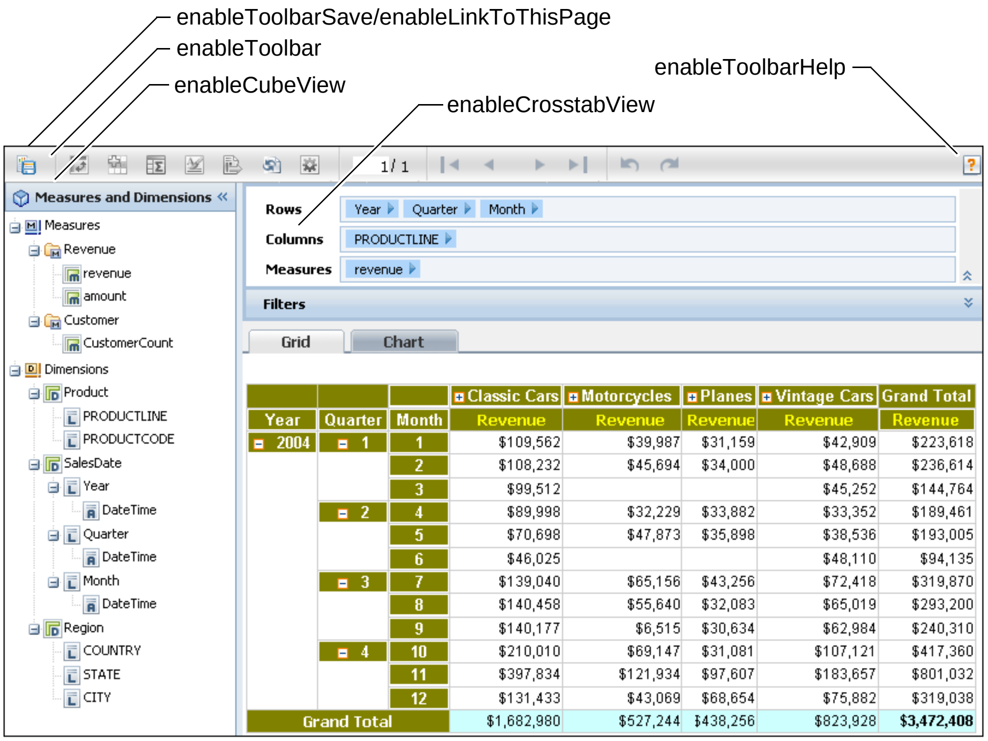
```
var driller = new actuate.xtabanalyzer.Driller( );
driller.setAxisType(actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE);
crosstab.drill(driller);
crosstab.submit( );
```

Controlling the Data Analyzer viewer user interface

Show or hide Data Analyzer viewer features with the actuate.xtabanalyzer.UIOptions class. The UIOptions class includes functions that support the ability to hide or show different features of the viewer. Figure 3-4 shows what functions affect the viewer display.

Pass true or false values to the UIOptions functions to display or hide the portion of the viewer that is associated with that particular function, as shown in the following code:

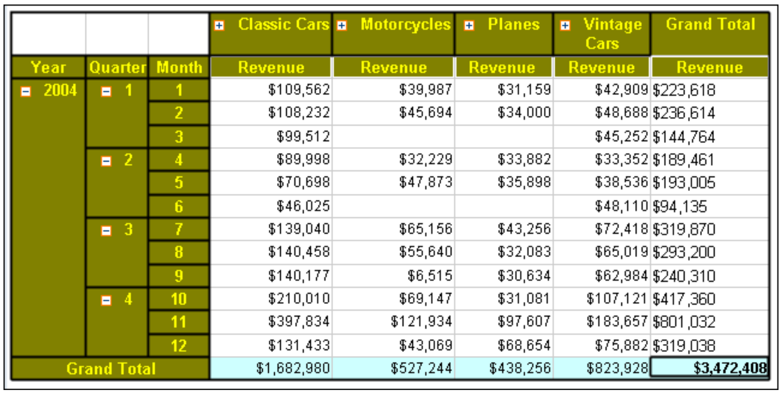
```
var uiOptions = new actuate.xtabanalyzer.UIOptions( );
uiOptions.enableToolbar(false);
uiOptions.enableCubeView(false);
uiOptions.enableCrosstabView(false);

// ctViewer is an instance of the XTabAnalyzer class.
ctViewer.setUIOptions( uiOptions );
```



			Classic Cars	Motorcycles	Planes	Vintage Cars	Grand Total
Year	Quarter	Month	Revenue	Revenue	Revenue	Revenue	Revenue
2004	1	1	$109,562	$39,987	$31,159	$42,909	$223,618
		2	$108,232	$45,694	$34,000	$48,688	$236,614
		3	$99,512			$45,252	$144,764
	2	4	$89,998	$32,229	$33,882	$33,352	$189,461
		5	$70,698	$47,873	$35,898	$38,536	$193,005
		6	$46,025			$48,110	$94,135
	3	7	$139,040	$65,156	$43,256	$72,418	$319,870
		8	$140,458	$55,640	$32,083	$65,019	$293,200
		9	$140,177	$6,515	$30,634	$62,984	$240,310
	4	10	$210,010	$69,147	$31,081	$107,121	$417,360
		11	$397,834	$121,934	$97,607	$183,657	$801,032
		12	$131,433	$43,069	$68,654	$75,882	$319,038
Grand Total			$1,682,980	$527,244	$438,256	$823,928	$3,472,408

Figure 3-4 Data Analyzer viewer showing areas altered with UIOptions

This code produces a viewer similar to Figure 3-5.



			Classic Cars	Motorcycles	Planes	Vintage Cars	Grand Total
Year	Quarter	Month	Revenue	Revenue	Revenue	Revenue	Revenue
2004	1	1	$109,562	$39,987	$31,159	$42,909	$223,618
		2	$108,232	$45,694	$34,000	$48,688	$236,614
		3	$99,512			$45,252	$144,764
	2	4	$89,998	$32,229	$33,882	$33,352	$189,461
		5	$70,698	$47,873	$35,898	$38,536	$193,005
		6	$46,025			$48,110	$94,135
	3	7	$139,040	$65,156	$43,256	$72,418	$319,870
		8	$140,458	$55,640	$32,083	$65,019	$293,200
		9	$140,177	$6,515	$30,634	$62,984	$240,310
	4	10	$210,010	$69,147	$31,081	$107,121	$417,360
		11	$397,834	$121,934	$97,607	$183,657	$801,032
		12	$131,433	$43,069	$68,654	$75,882	$319,038
Grand Total			$1,682,980	$527,244	$438,256	$823,928	$3,472,408

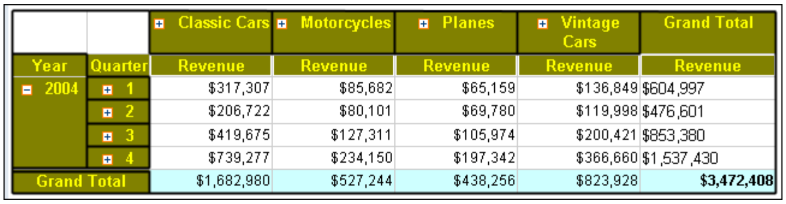
Figure 3-5 Data Analyzer viewer with settable UIOptions off

In addition to the UIOption class, some details shown within the viewer can be hidden with Crosstab.showDetail( ) and Crosstab.hideDetail( ).

For example, the cross tab in Figure 3-5 has a SalesDate dimension consisting of three levels: year, quarter, and month. The following code hides the detail from the quarter level of the dimension. In this example, crosstab is an actuate.xtabanalyzer.Crosstab object:

```
crosstab.hideDetail("SalesDate/quarter");
crosstab.submit( );
```

The code in this example modifies the cross tab so it longer shows the month detail level, as shown in Figure 3-6.



		Classic Cars	Motorcycles	Planes	Vintage Cars	Grand Total
Year	Quarter	Revenue	Revenue	Revenue	Revenue	Revenue
2004	1	$317,307	$85,682	$65,159	$136,849	$604,997
	2	$206,722	$80,101	$69,780	$119,998	$476,601
	3	$419,675	$127,311	$105,974	$200,421	$853,380
	4	$739,277	$234,150	$197,342	$366,660	$1,537,430
Grand Total		$1,682,980	$527,244	$438,256	$823,928	**$3,472,408**

Figure 3-6 Cross tab with level detail hidden

To display the detail again, use showDetail( ).

```
var axisType = actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE;
crosstab.showDetail(axisType,"SalesDate/quarter");
crosstab.submit( );
```

Chapter

4

Actuate JavaScript API classes

This chapter contains the following topics:

- Actuate JavaScript API overview
- Actuate JavaScript API classes quick reference
- Actuate JavaScript API reference

Actuate JavaScript API overview

The Actuate JavaScript API is a set of JavaScript classes used to create custom web content that contains Actuate BIRT reports and report elements.

An HTML-based JSDoc JavaScript API class reference is provided for Information Console and Actuate Java Components in the following file:

```
<Context Root>\help\jsapi\index.html
```

About the actuate namespace

All of the Actuate JavaScript API classes are in the actuate namespace. To use the viewer element, call the actuate.Viewer class.

In addition, the Actuate JavaScript API has a static class:

```
actuate
```

This class handles connections to Actuate web applications and is the only static class in the Actuate JavaScript API.

Using the Actuate library

The Actuate JavaScript library is available from any Information Console installation or Actuate Deployment Kit for BIRT reports. The URL for the library is:

```
http://localhost:8700/iportal/jsapi
```

where

- localhost:8700 is the host name and TCP port for an available Actuate web application host.
- /iportal is the context root for the web application.
- /jsapi is the default location of the Actuate JavaScript API libraries.

A script tag loads the Actuate JavaScript API library, as shown in the following code:

```
<script type="text/javascript" src="http://localhost:8700
   /iportal/jsapi">
</script>
```

To call JavaScript functions, use additional script tags after the script tag that adds these libraries for the page.

Actuate JavaScript API classes quick reference

Table 4-1 lists the Actuate JavaScript API classes.

Table 4-1 Actuate JavaScript API classes

JavaScript class	Description
actuate	Entry point to the Actuate JavaScript API library
actuate.AuthenticationException	Exception caused by failed authentication
actuate.ConnectionException	Exception caused by a failed connection
actuate.Dashboard	Dashboard class
actuate.dashboard.DashboardDefinition	Dashboard wrapper class
actuate.dashboard.EventConstants	Global constants for the dashboard events class
actuate.dashboard.GadgetScript	Dashboard gadget script class
actuate.dashboard.Tab	Dashboard tab class
Actuate.data	Container for actuate data class
actuate.data.Filter	Conditions to filter data
actuate.data.ReportContent	Represents downloaded content
actuate.data.Request	Represents and processes a request for report data
actuate.data.ResultSet	Results retrieved from a report document in response to a request
actuate.data.Sorter	Sort conditions to sort data
actuate.DataService	Data services to retrieve data from a report document
actuate.Exception	Exception object passed to a callback function or exception handler
actuate.Parameter	Parameters from a report
actuate.parameter.Constants	Global navigation and layout constants used for the Parameter class

(continues)

Table 4-1 Actuate JavaScript API classes (continued)

JavaScript class	Description
actuate.parameter.ConvertUtility	Converts parameters into specific and generic formats
actuate.parameter.EventConstants	Defines the events for parameters this API library supports
actuate.parameter.NameValuePair	Display name and the associated value
actuate.parameter.ParameterData	a high-level wrapper for an actuate.parameter .ParameterDefinition object
actuate.parameter.ParameterDefinition	Qualities, options, name, and format for a parameter as the parameter displays and accepts values
actuate.parameter.ParameterValue	The parameter's value as processed by a report
actuate.report.Chart	A report chart
actuate.report.DataItem	A report data item
actuate.report.FlashObject	A report Flash object
actuate.report.Gadget	A report gadget
actuate.report.HTML5Chart.ClientChart	An HTML5 enabled chart
actuate.report.HTML5Chart.ClientOption	Options for an HTML5 enabled chart
actuate.report.HTML5Chart.ClientPoint	A data point for an HTML5 enabled chart
actuate.report.HTML5Chart.ClientSeries	A data series for an HTML5 enabled chart
actuate.report.HTML5Chart.Highcharts	A Highchart object
actuate.report.HTML5Chart.Renderer	A Highchart renderer object
actuate.report.Label	A report label element
actuate.report.Table	A report table element
actuate.report.TextItem	A report text element
actuate.ReportExplorer	The report explorer general container
actuate.reportexplorer.Constants	Global constants used for ReportExplorer class

Table 4-1 Actuate JavaScript API classes (continued)

JavaScript class	Description
actuate.reportexplorer.EventConstants	Global EventConstants used for ReportExplorer class
actuate.reportexplorer.File	A file listed in the ReportExplorer and the file's properties
actuate.reportexplorer.FileCondition	A JavaScript version of com.actuate.schemas.FileCondition
actuate.reportexplorer.FileSearch	A JavaScript version of com.actuate.schemas.FileSearch
actuate.reportexplorer.FolderItems	A JavaScript version of com.actuate .schemas.GetFolderItemsResponse
actuate.reportexplorer.PrivilegeFilter	A JavaScript version of com.actuate .schemas.PrivilegeFilter
actuate.RequestOptions	URL parameters for requests to an iHub volume
actuate.Viewer	A report viewer component that can be embedded in an HTML page
actuate.viewer.BrowserPanel	A non-scrolling panel display
actuate.viewer.EventConstants	Defines the events for the viewer this API library supports
actuate.viewer.PageContent	Content shown on the viewer
actuate.viewer.ParameterValue	Parameter values in the viewer
actuate.viewer.RenderOptions	Options for downloading reports
actuate.viewer.ScrollPanel	A scrolling panel display
actuate.viewer.SelectedContent	Selected report element
actuate.viewer.UIConfig	Enables UI elements of the scrolling panel display
actuate.viewer.UIOptions	Enables UI elements of the viewer
actuate.viewer.ViewerException	Exception constants supported for the viewer

Actuate JavaScript API reference

This section provides an alphabetical listing of the JavaScript API classes.

Class actuate

Description The entry point to the Actuate JavaScript API library. The actuate class uses load( ) to generate data, viewer, cross tab, parameter, explorer, and other components. The actuate class uses initialize( ) and authenticate( ) to connect to an Actuate web application service.

Use actuate.load( ) before calling actuate.initialize( ). The actuate.initialize( ) function loads all of the components added with load( ).

The initialize( ) function connects to an initial Actuate web application service. To connect to additional services simultaneously, use authenticate( ). Call initialize( ) before calling authenticate( ).

Constructor

The static actuate class loads when the a <script> element loads the Actuate JavaScript API.

Function summary

Table 4-2 lists actuate functions.

Table 4-2 actuate functions

Function	Description
authenticate( )	Connects to an Actuate web application service and authenticates
getDefaultIportalUrl( )	Returns the default service URL
getDefaultRequestOptions( )	Returns the default request options
getVersion( )	Returns the Actuate web application version
getViewer( )	Returns a viewer instance containing the given bookmark element
initialize( )	Connects to an initial Actuate web application service, loads an initial component, and invokes a callback function
isConnected( )	Reports whether a given Actuate web application is connected
isInitialized( )	Returns whether a library is initialized
load( )	Loads the library for an additional component
logout( )	Logs a user out of an Actuate web application service

actuate.authenticate

Syntax void authenticate(string iPortalURL, actuate.RequestOptions requestOptions, string userid, string password, function callback, string credentials, function errorCallback)

Connects to the Actuate web application service that is addressed by iPortalURL and authenticates the connection.

Parameters

iPortalURL
The iPortalURL parameter is a required string parameter that specifies the target Actuate web application URL.

requestOptions
The requestOptions parameter is an optional actuate.RequestOptions object. The requestOptions parameter defines the URL parameters to send with the authentication request, such as the iHub URL, Encyclopedia volume, or repository type. Functions in the RequestOptions class enable the addition of custom parameters to the URL. When requestOptions is null, authenticate( ) uses the default parameter values for the target Actuate web application URL. These default parameter values are defined in the Actuate web application's web.xml file.

userid
The userid parameter is an optional string parameter that contains the login user id when the login user id is not provided in the session.

password
The password parameter is an optional string parameter that contains the login password when the login password is not provided in the session.

credentials
The credentials parameter is an optional string parameter. This parameter holds information that supports checking user credentials with an externalized system such as LDAP. The credentials parameter supports additional credential information for any additional security systems in place on the application server where the web service is deployed.

callback
The callback parameter is an optional function to call after initialization. The actuate.authenticate( ) function passes the following variables to the callback function:

- iportalURL: The iportal URL passed in from the iPortalURL parameter
- userid: The authenticated user ID
- iserverURL: The BIRT iHub URL
- volume: The Encyclopedia volume name

errorCallback
The errorCallback parameter is an optional function that specifies a function to call when an error occurs. The possible errors are actuate.ConnectionException, actuate.AuthenticationException, and actuate.Exception. The callback function must take an exception as an argument.

Example To connect to an additional Actuate web service called digits, use code similar to the following:

```
actuate.authenticate("http://digits:8700/iportal", null, myname,
   mypassword, null, null, null);
```

actuate.getDefaultIportalUrl

Syntax String getDefaultIportalUrl( )

Returns the default service URL.

Returns String. The default service URL.

Example This example calls actuate.getDefaultIportalUrl( ) to return the default service URL:

```
alert ("The default service URL is " + getDefaultIportalUrl( ));
```

actuate.getDefaultRequestOptions

Syntax actuate.RequestOptions getDefaultRequestOptions( )

Returns the default request options.

Returns actuate.RequestOptions object that contains the default request options.

Example This example calls actuate.getDefaultRequestOptions( ) to return the default iHub URL:

```
alert ("The default iHub URL is " +
   actuate.getDefaultRequestOptions( ).getServerUrl( ));
```

actuate.getVersion

Syntax string getVersion( )

Returns the Actuate web application version.

Returns String. The string contains the Actuate web application version in the format "#version# (Build #buildnumber#)".

Example The following sample code displays the version in an alert box:

```
alert("Version: " + actuate.getVersion( ));
```

actuate.getViewer

Syntax actuate.Viewer getViewer(string bookmark)

actuate.Viewer getViewer(htmlelement viewer)

Returns a viewer instance containing the given bookmark element. Load the viewer module before calling actuate.getViewer( ).

Parameters **bookmark**
This string parameter contains the name of the bookmark to retrieve or the name of an HTML <div> element.

viewer
This parameter is the DOM htmlelement object for the HTML <div> element that contains a viewer.

Returns An actuate.Viewer object that contains a viewer. When actuate.getViewer( ) does not find a viewer, the function returns null.

Example To retrieve the viewer assigned to the first_viewer <div> element on the page, use code similar to the following:

```
currentViewer = actuate.getViewer("first_viewer");
```

actuate.initialize

Syntax void initialize(string iPortalURL, actuate.RequestOptions requestOptions, reserved, reserved, function callback, function errorCallback)

Connects to an initial Actuate web application service, loads all of the components added with load( ), and invokes a callback function.

Authentication is optional in initialize( ).

When using more than one service in one mashup page, use actuate.authenticate( ) to connect to additional services.

Parameters **iPortalURL**
String. The target Actuate web application URL.

requestOptions
actuate.RequestOptions object. Optional. requestOptions defines URL parameters to send in the authentication request, such as the iHub URL, Encyclopedia volume, or repository type. It can also add custom parameters to the URL. If requestOptions is null, initialize( ) uses the default parameter values for the target Actuate web application URL. These default parameter values are defined in Actuate web application's web.xml file.

reserved
Set to null.

reserved
Set to null.

callback
Function. The callback function called after the initialization is done. The following variables are passed to the callback function:

- iportalUrl: The iportal URL passed in from the iPortalURL parameter
- userId: The authenticated user ID
- iserverUrl: The BIRT iHub URL
- volume: The Encyclopedia volume name

errorCallback
Function. The function to call when an error occurs. The possible errors are actuate.ConnectionException, actuate.AuthenticationException, and actuate.Exception. errorCallback must take an exception as an argument.

Example To initialize the client connection to a web service on myhost and then run the init( ) function, use the following code:

```
actuate.initialize("http://myhost:8700/iportal", null, null, null,
   init, null);
```

actuate.isConnected

Syntax boolean isConnected(string iportalUrl, actuate.RequestOptions requestOptions)

Returns whether a given Actuate web application URL is connected.

Parameters **iPortalURL**
String. The target Actuate web application URL.

requestOptions
actuate.RequestOptions object. Optional. requestOptions defines URL parameters to send with the authentication request, such as the iHub URL, Encyclopedia volume, or repository type. It can also add custom parameters to the URL. If requestOptions is null, initialize( ) uses the default parameter values for the target Actuate web application URL. These default parameter values are defined in Actuate web application's web.xml file.

Returns Boolean. True if there is a connection to the given Actuate web application, False if there is no connection or if it is pending.

Example The following sample code connects to the digits service using authenticate if not currently connected:

```
if (!actuate.isConnected("http://digits:8700/iportal", null)){
   actuate.authenticate("http://digits:8700/iportal", null,
   myname, mypassword, null, null, null);
}
```

actuate.isInitialized

Syntax boolean isInitialized( )

Returns whether the library is already initialized.

Returns Boolean. True if the library is already initialized.

Example The following sample code initializes a connection with the Actuate web service if one is not already initialized:

```
if (!actuate.isInitialized( )){
   actuate.initialize("http://myhost:8700/iportal", null, null,
   null, init, null);
}
```

actuate.load

Syntax void load(string componentName)

Specifies a component to be loaded by actuate.initialize( ). The available components are:

- dashboard: The dashboard component including the actuate.Dashboard package
- dialog: The dialog component including the actuate.Dialog class
- parameter: The parameter page component including the actuate.Parameter package
- reportexplorer: The report explorer component including the actuate.ReportExplorer package
- viewer: The viewer component including the actuate.Viewer and actuate.DataService packages
- xtabAnalyzer: The data analyzer component, including the actuate.XTabAnalyzer package

Parameters **componentName**
String. componentName is a case-sensitive parameter. Valid component names are listed above.

Example To enable a page to use viewer, dialog, and parameters, call actuate.load( ) three times, as shown in the following code:

```
actuate.load("viewer");
actuate.load("dialog");
actuate.load("parameter");
```

actuate.logout

Syntax void logout(string iPortalURL, actuate.RequestOptions requestOptions, function callback, function errorCallback)

Logs out from the given Actuate web application URL and removes authentication information from the session. If the application was previously not logged in to this Actuate web application, it generates no errors but still calls the callback function.

Parameters **iPortalURL**
String. The target Actuate web application URL.

requestOptions
actuate.RequestOptions object. Optional. requestOptions defines URL parameters to send with the authentication request, such as the iHub URL, Encyclopedia volume, or repository type. It can also add custom parameters to the URL. If requestOptions is null, initialize( ) uses the default parameter values for the target Actuate web application URL. These default parameter values are defined in Actuate web application's web.xml file.

callback
Function. Optional. The callback function called after the logout is done.

errorCallback
Function. The function called when an error occurs. The possible errors are actuate.ConnectionException, actuate.AuthenticationException, and actuate.Exception. errorCallback must take an exception as an argument.

Example The following sample code disconnects to the digits service if currently connected:

```
if (actuate.isConnected("http://digits:8700/iportal", null)){
   actuate.logout("http://digits:8700/iportal", null, null, null);
}
```

Class actuate.AuthenticationException

Description AuthenticationException provides an object to pass to a error callback function when an authentication exception occurs. The AuthenticationException object contains references to the URL, the UserId, and the request options used in the authentication attempt.

Constructor

The AuthenticationException object is constructed when actuate.Authenticate( ) fails.

Function summary

Table 4-3 lists actuate.AuthenticationException functions.

Table 4-3 actuate.AuthenticationException functions

Function	Description
getIportalUrl( )	Returns the web service URL
getRequestOptions( )	Returns the request options
getUserId( )	Returns the user ID

actuate.AuthenticationException.getIportalUrl

Syntax string AuthenticationException.getIportalUrl( )

Returns the Deployment Kit for BIRT reports or Information Console URL.

Returns String.

Example The following sample code retrieves the URL from an exception:

```
return AuthenticationException.getIportalUrl( );
```

actuate.AuthenticationException.getRequestOptions

Syntax actuate.RequestOptions AuthenticationException.getRequestOptions( )

Returns an instance of the requestOptions that modified the URL that caused the exception, if applicable.

Returns actuate.RequestOptions object. A RequestOptions object defines URL parameters sent in the authentication request, such as the iHub URL, Encyclopedia volume, or repository type. The RequestOptions object can also add custom parameters to the URL.

Example The following sample code retrieves the RequestOptions object that caused the exception:

```
var exceptReqOpts = AuthenticationException.getRequestOptions( );
```

actuate.AuthenticationException.getUserId

Syntax string AuthenticationException.getUserId( )

Returns the UserId used in the failed authentication attempt.

Returns String.

Example The following sample code retrieves the UserId from an exception:

```
return AuthenticationException.getUserId( );
```

Class actuate.ConnectionException

Description A container for a connection exception. ConnectionException provides an object to pass to a error callback function when an exception occurs.

Constructor

The ConnectionException object is constructed when there is a connection issue. For example, actuate.ConnectionException is created when a wrong URL is given in actuate.initialize( ) or actuate.authenticate( ), or if the server was unreachable.

Function summary

Table 4-4 describes actuate.ConnectionException functions.

Table 4-4 actuate.ConnectionException function

Function	Description
getUrl( )	Returns the whole URL

actuate.ConnectionException.getUrl

Syntax string ConnectionException.getUrl( )

Returns the complete URL sent with the connection request.

Returns String. The complete URL that was sent with the connection request.

Example This example calls ConnectionException.getUrl( ) to return the complete URL from a Connection request:

```
alert ("Connection Error at " + ConnectionException.getUrl( ));
```

Class actuate.Dashboard

Description Represents a dashboard object.

Constructor

Syntax actuate.Dashboard(string container)

Constructs a dashboard object.

Parameters **container**
String. Optional. Container object or name of a container in the current document ID of container where controls are to be rendered.

Function summary

Table 4-5 describes actuate.Dashboard functions.

Table 4-5 actuate.Dashboard functions

Function	Description
downloadDashboard( )	Downloads the dashboard definitions.
embedTemplate( )	The personal dashboard uses an embedded template file.
getActiveTab( )	Returns the active tab name.
getDashboardName( )	Returns the dashboard name used by the dashboard object.
getTemplate( )	Returns the iHub volume repository path.
isAutoSaveEnabled( )	Returns whether autosave is enabled.
isSavingNeeded( )	Returns whether there are unsaved changes on the dashboard.
isUsingPersonalDashboard( )	Returns whether the dashboard is a personal dashboard.
onUnload( )	Unloads JavaScript variables that are no longer needed by the dashboard.
registerEventHandler( )	Registers an event handler.
removeEventHandler( )	Removes an event handler.
renderContent( )	Renders the dashboard.
save( )	Saves the dashboard as a .dashboard file.
saveAs( )	Saves the dashboard in non-default location.
setActiveTab( )	Sets a specific tab as the active tab.

Table 4-5 actuate.Dashboard functions

Function	Description
setAutoSaveDelay( )	Sets the time interval before executing an automatic save for a personal dashboard.
setContainer( )	Sets the container for rendering the dashboard page HTML fragment.
setDashboardName( )	Sets the dashboard name to view.
setHeight( )	Sets the dashboard height.
setService( )	Sets the connection to the Actuate web service.
setSize( )	Sets the dashboard size.
setTemplate( )	Sets the template path.
setWidth( )	Sets the dashboard width.
showGallery( )	Shows the dashboard gallery.
showTabNavigation( )	Shows the tab toolbar.
submit( )	Submits the dashboard page component request.
usePersonalDashboard( )	Forces the dashboard framework to use a personal dashboard.

actuate.Dashboard.downloadDashboard

Syntax void Dashboard.downloadDashboard(function callback)

Downloads the dashboard metadata definitions.

Parameters **callback**
Function. The callback function to use after the dashboard finishes downloading. This function must take the returned dashboard object as an input parameter.

Example This example specifies a function to call after the dashboard object finishes downloading:

```
myDashboard.downloadDashboard(runNext);
function runNext(dashobject){
   mydashboard.getDashboardName(dashobject);
}
```

actuate.Dashboard.embedTemplate

Syntax void Dashboard.embedTemplate(boolean isEmbedded)

A personal dashboard can use a shared template file or embed a template file.

Parameters **isEmbedded**
Boolean. When the isEmbedded parameter is true, the personal dashboard uses an embedded template file. The default value is false.

Example This example specifies that the personal dashboard myDashboard uses an embedded template file:

```
myDashboard.embedTemplate(true);
```

actuate.Dashboard.getActiveTab

Syntax string Dashboard.getActiveTab

Returns the name of the current active tab for the dashboard.

Returns String. The name of the current active dashboard tab.

Example This example displays the name of the active tab for the myDashboard dashboard object in an alert box:

```
alert(myDashboard.getActiveTab( ));
```

actuate.Dashboard.getDashboardName

Syntax string Dashboard.getDashboardName( )

Returns the dashboard name used by the dashboard object.

Returns String. The dashboard's name.

Example This example displays the dashboard object's dashboard name in an alert box:

```
alert(myDashboard.getDashboardName( ));
```

actuate.Dashboard.getTemplate

Syntax string Dashboard.getTemplate( )

Returns the repository path for the iHub volume.

Returns String. The repository path for the iHub volume.

Example This example displays the repository path for the iHub volume in an alert box:

```
alert(myDashboard.getTemplate( ));
```

actuate.Dashboard.isAutoSaveEnabled

Syntax boolean Dashboard.isAutoSaveEnabled( )

Returns whether the autosave feature is enabled.

Returns Boolean. True indicates that autosave is enabled.

Example This example informs the user of the status of the autosave feature:

```
if (dashboard.isAutoSavEnabled()){
  alert("Autosave is enabled.");
}else{
  alert("Autosave is disabled.");
}
```

actuate.Dashboard.isSavingNeeded

Syntax boolean Dashboard.isSavingNeeded( )

Returns whether there are unsaved changes on the dashboard.

Returns Boolean. True indicates that there are unsaved changes on the dashboard.

Example This example informs the user of unsaved changed:

```
if (dashboard.isSavingNeeded()){
  alert("The dashboard contains unsaved changes.");
}
```

actuate.Dashboard.isUsingPersonalDashboard

Syntax boolean Dashboard.isUsingPersonalDashboard( )

Returns whether this dashboard is a personal dashboard.

Returns Boolean. True indicates that this dashboard is a personal dashboard.

Example This example informs the user that they are using a personal dashboard:

```
if (dashboard.isUsingPersonalDashboard)){
  alert("This is a personal dashboard.");
}
```

actuate.Dashboard.onUnload

Syntax void Dashboard.onUnload( )

Unloads JavaScript variables that are no longer needed by the dashboard.

Example This example unloads JavaScript variables and displays the dashboard object's dashboard name in an alert box:

```
myDashboard.onUnload;
alert("JS variables unloaded for " +
        myDashboard.getDashboardName( ));
```

actuate.Dashboard.registerEventHandler

Syntax void Dashboard.registerEventHandler(string eventName, function handler)

Registers an event handler to activate for parameter eventName. This function can assign several handlers to a single event.

Parameters **eventName**
String. Event name to capture.

handler
Function. The function to execute when the event occurs. The handler must take two arguments: The dashboard instance that fired the event and an event object specific to the event type.

Example This example registers the errorHandler( ) function to respond to the ON_EXCEPTION event:

```
myDashboard.registerEventHandler(actuate.dashboard.EventConstants.
                                  ON_EXCEPTION, errorHandler);
```

actuate.Dashboard.removeEventHandler

Syntax void Dashboard.removeEventHandler(string eventName, function handler)

Removes an event handler to activate for parameter eventName.

Parameters **eventName**
String. Event name to remove from the internal list of registered events.

handler
Function. The function to disable.

Example This example removes the errorHandler( ) function from responding to the ON_EXCEPTION event:

```
myDashboard.removeEventHandler(actuate.dashboard.EventConstants.
                                  ON_EXCEPTION, errorHandler);
```

actuate.Dashboard.renderContent

Syntax void Dashboard.renderContent(object[ ] dashboardDefinitions, function callback)

Renders the dashboard definitions content to the container. The submit API calls the renderContent API internally. The renderContent( ) function assumes that the user has already a list of DashboardDefinition to process.

Parameters **dashboardDefinitions**
Array of objects. Each object is some piece of dashboard metadata and as many can be added as needed. Typically, this array contains the following metadata:

- Number of tabs in a dashboard file
- Number of sections/columns in a dashboard tab
- Number of gadgets in each section/column

- Attributes of each gadget
- Attributes of each tab
- Dependency information between gadgets to support publishing and subscribing mechanism

callback
Function. The callback function to call after renderContent( ) finishes.

Example This example renders the myDash dashboard object using the dashboardDefinition array defs and calls afterRender( ) once complete:

```
myDash.renderContent(defs, afterRender);
```

actuate.Dashboard.save

Syntax void Dashboard.save(function callback, boolean flag)

Saves the dashboard as a .dashboard file.

Parameters **callback**
Function. Optional. The function to execute after the save operation completes.

flag
Boolean. Optional. True indicates a synchronous save operation.

Example This example saves the dashboard as a .dashboard file:

```
myDash.save( );
```

actuate.Dashboard.saveAs

Syntax void Dashboard.saveAs(function callback, string saveAsPath, boolean replace, boolean flag)

Saves the dashboard as a .dashboard file to a specific path.

Parameters **callback**
Function. Optional. The function to execute after the save operation completes.

saveAsPath
String. Optional. Fully qualified path in which to save the dashboard. The default value is the path for the original dashboard file, if one exists, or the path for the user's home directory.

replace
Boolean. Optional. True indicates to replace the latest version of the file. False indicates to create a new version.

flag
Boolean. Optional. True indicates a synchronous save operation.

Example This example saves the dashboard as a .dashboard file, replacing the latest version:

```
myDash.saveAs(null, null, true, true);
```

actuate.Dashboard.setActiveTab

Syntax void Dashboard.setActiveTab(string tabName)

Sets a specified tab as the active tab. Only one tab can be active at a time.

Parameters **tabName**
String. The name of the tab to set as the active tab.

Example This example sets the Files tab as the active tab for this dashboard:

```
myDash.setActiveTab("Files");
```

actuate.Dashboard.setAutoSaveDelay

Syntax void Dashboard.setAutoSaveDelay(integer seconds)

Sets the amount of time before executing an automatic save for a personal dashboard.

Parameters **seconds**
Integer. The number of seconds to delay the automatic save.

Example This example sets the delay for the automatic save for dashboard myDash to 5 minutes:

```
myDash.setAutoSaveDelay(300);
```

actuate.Dashboard.setContainer

Syntax void Dashboard.setContainer(string containerID)

The container that will be used for rendering the dashboard page HTML fragment.

Parameters **containerID**
String. The container ID.

Example This example sets the container where the myDash dashboard object renders:

```
myDash.setContainer("leftpane");
```

actuate.Dashboard.setDashboardName

Syntax void Dashboard.setDashboardName(string dashboardName)

Sets the dashboard name to view.

Parameters **dashboardName**
String. A fully qualified repository path and file name.

Example This example sets the path for the myDash dashboard object:

```
myDash.setDashboardName("/Dashboard/Contents/Hello.DASHBOARD");
```

actuate.Dashboard.setHeight

Syntax void Dashboard.setHeight(integer height)

Sets the dashboard's startup height.

Parameters **height**
Integer. Specifies the height in pixels.

Example To set the dashboard height to 400 pixels, use code similar to the following:

```
myDashboard.setHeight(400);
```

actuate.Dashboard.setService

Syntax void Dashboard.setService(string iportalURL, actuate.RequestOptions
    requestOptions)

Sets the web service this dashboard component connects to.

Parameters **iportalURL**
String. The URL of the web service to connect to.

requestOptions
actuate.RequestOptions object. Request options, if any, to apply to the connection. See actuate.RequestOptions for details on the options that this parameter can set.

Example This example connects a dashboard component to the iPortal service and adds a custom URL parameter:

```
function setDashboardService( ){
   myDashboard.setService("http://localhost:8700/iportal",
   myRequestOptions.setCustomParameters({myParam: "myValue"});
}
```

actuate.Dashboard.setSize

Syntax void Dashboard.setSize(integer height, integer width)

Sets the dashboard's startup size.

Parameters **height**
Integer. Height in pixels.

width
Integer. Width in pixels.

Example To set the dashboard height to 400 pixels and the width to 800 pixels, use code similar to the following:

```
myDashboard.setSize(400, 800);
```

actuate.Dashboard.setTemplate

Syntax void Dashboard.setTemplate(string path)

Sets the template path. This function overwrites the template path that is used by Information Console.

Parameters **path**
String. Specifies a new template path. Use an iHub volume repository path.

Example This example sets the template path for myDashboard to /iportal/jsapi /template/path:

```
myDashboard.setTemplate("/iportal/jsapi/template/path");
```

actuate.Dashboard.setWidth

Syntax void Dashboard.setWidth(integer width)

Sets the dashboard's startup width.

Parameters **width**
Integer. Specifies the width in pixels.

Example To set the dashboard width to 800 pixels, use code similar to the following:

```
myDashboard.setWidth(800);
```

actuate.Dashboard.showGallery

Syntax void Dashboard.showGallery(boolean show)

Shows the gadget gallery.

Parameters **show**
Boolean. The gadget gallery is visible when this parameter is set to true.

Example To show the gadget gallery for the myDashboard dashboard object, use code similar to the following:

```
myDashboard.showGallery(true);
```

actuate.Dashboard.showTabNavigation

Syntax void Dashboard.showTabNavigation(boolean show)

Shows the tab toolbar.

Parameters **show**
Boolean. The tab toolbar is visible when this parameter is set to true.

Example To show the tab toolbar for the myDashboard dashboard object, use code similar to the following:

```
myDashboard.showTabNavigation(true);
```

actuate.Dashboard.submit

Syntax void Dashboard.submit(function callback)

Submits requests to the server for the dashboard. When this function is called, an AJAX request is triggered to submit all pending operations. When the server finishes the processing, it returns a response and the results are rendered on the page in the dashboard container.

Parameters **callback**
Function. The function to execute after the asynchronous call processing is done.

Example This example submits the dashboard name that was set with setDashboardName( ):

```
dash.setDashboardName("/Dashboard/Contents/Hello.DASHBOARD");
dash.submit( );
```

actuate.Dashboard.usePersonalDashboard

Syntax void Dashboard.usePersonalDashboard(boolean true|false)

Forces the dashboard framework to use the user's personal dashboard.

Parameters **true|false**
Boolean. A value of true sets the dashboard framework to ignore any value set by the setDashboardName( ) method. The dashboard framework creates a new personal dashboard file for the logged in user when no personal dashboard file is present.

Example To force the use of a personal dashboard for the myDashboard object, use code similar to the following:

```
myDashboard.usePersonalDashboard(true);
```

Class actuate.dashboard.DashboardDefinition

Description The DashboardDefinition class is a wrapper class for a dashboard file definition.

Constructor

Syntax actuate.dashboard.DashboardDefinition( )

Constructs a new DashboardDefinition object.

Function summary

Table 4-6 lists the actuate.dashboard.DashboardDefinition functions.

Table 4-6 actuate.dashboard.DashboardDefinition functions

Function	Description
getDefaultActiveTab( )	Returns the name of the default active tab for this dashboard definition
getTabs( )	Returns an array of the tabs in this dashboard definition

actuate.dashboard.DashboardDefinition .getDefaultActiveTab

Syntax string DashboardDefinition.getDefaultActivetab( )

Returns the name of the default active tab for this dashboard definition.

Returns String. The name of the default active tab.

Example This example displays the default active tab for the myDashDef DashboardDefinition object in an alert box:

```
alert(myDashboard.getDefaultActiveTab( ));
```

actuate.dashboard.DashboardDefinition.getTabs

Syntax array DashboardDefinition.getTabs( )

Returns an array of the tabs in this dashboard definition.

Returns Array. An array of actuate.dashboard.Tab objects.

Example This example assigns the array of tabs to the mytabs variable:

```
var mytabs = new Array[myDashDef.getTabs( )];
```

Class actuate.dashboard.EventConstants

Description Defines the event constants supported by the Dashboard class. Table 4-7 lists the dashboard event constants.

Table 4-7 Actuate JavaScript API dashboard event constants

Event	Description
ON_EXCEPTION	Event triggered when an exception occurs. The event handler takes an actuate.Exception object as an input parameter.
ON_SESSION_TIMEOUT	Session time-out event. Whenever a session time-out event occurs and the user tries to perform any operation on the explorer, a prompt dialog appears to ask whether the user wants to log in again. If the user clicks yes, the ON_SESSION_TIMEOUT event fires. If no handler has been registered for this event, the viewer displays a default login dialog. The event handler takes the current actuate.Dashboard object as an input parameter.

Class actuate.dashboard.GadgetScript

Description The actuate.dashboard.GadgetScript class is a container for the information passed to the onChange event function.

Constructor

Syntax

```
onChange( string event, actuate.dashboard.GadgetScript publisher, object data,
    actuate.dashboard.GadgetScript thisGadget )
```

Constructs a new GadgetScript object. This object contains the publisher and thisGadget for an onChange event signature.

Parameters

event
String. An event name.

publisher
actuate.dashboard.GadgetScript object. The publisher gadget.

data
Object. Data to pass to the subscriber.

thisGadget
actuate.dashboard.GadgetScript object. thisGadget is this script gadget.

Function summary

Table 4-8 lists the actuate.dashboard.GadgetScript functions.

Table 4-8 actuate.dashboard.GadgetScript functions

Function	**Description**
getCurrentReportParameters( )	Gets the current report parameter values for thisGadget
getGadgetName( )	Returns thisGadget's name
getGadgetTitle( )	Returns thisGadget's title
getGadgetType( )	Returns thisGadget's type
getTabName( )	Returns the name of the tab containing thisGadget
getTabTitle( )	Returns the title of the tab containing thisGadget

actuate.dashboard.GadgetScript .getCurrentReportParameters

Syntax actuate.parameter.ParameterValue[ ]
    GadgetScript.getCurrentReportParameters( )

Returns the current report parameter values for report and Reportlet gadgets.

Returns Array of actuate.parameter.ParameterValue objects. Parameter values assigned to this gadget.

actuate.dashboard.GadgetScript.getGadgetName

Syntax string GadgetScript.getGadgetName( )

Returns this gadget's name.

Returns String. The name of this gadget.

Example This example displays this gadget's name in an alert box:

```
alert(myGadgetScript.getGadgetName( ));
```

actuate.dashboard.GadgetScript.getGadgetTitle

Syntax string GadgetScript.getGadgetTitle( )

Returns this gadget's title.

Returns String. The title of this gadget.

Example This example displays this gadget's title in an alert box:

```
alert(myGadgetScript.getGadgetTitle( ));
```

actuate.dashboard.GadgetScript.getGadgetType

Syntax string GadgetScript.getGadgetType( )

Returns this gadget's type.

Returns String. This gadget's type.

Example This example displays this gadget's type in an alert box:

```
alert(myGadgetScript.getGadgetType( ));
```

actuate.dashboard.GadgetScript.getTabName

Syntax string GadgetScript.getTabName( )

Returns the name of the tab containing this gadget.

Returns String. The name of the tab containing this gadget.

Example This example displays the name of the tab containing this gadget in an alert box:

```
alert(myGadgetScript.getTabName( ));
```

actuate.dashboard.GadgetScript.getTabTitle

Syntax string GadgetScript.getTabTitle( )

Returns the title of the tab containing this gadget.

Returns String. The title of the tab containing this gadget.

Example This example displays the title of the tab containing this gadget in an alert box:

```
alert(myGadgetScript.getTabTitle( ));
```

Class actuate.dashboard.Tab

Description A wrapper class for the raw definition of a tab in a dashboard file.

Constructor

Syntax actuate.dashboard.Tab( )

Constructs a new tab object.

Function summary

Table 4-9 lists the actuate.dashboard.Tab functions.

Table 4-9 actuate.dashboard.Tab functions

Function	Description
getName( )	Returns the tab's name
getTabType( )	Returns the tab's type
getTitle( )	Returns the tab's title

actuate.dashboard.Tab.getName

Syntax string Tab.getName( )

Returns the tab's name.

Returns String. The name of the tab.

Example This example displays the tab object's name in an alert box:

```
alert(myTab.getName( ));
```

actuate.dashboard.Tab.getTabType

Syntax string Tab.getTabType( )

Returns the tab's type.

Returns String. The tab's type. The legal type values are ISystemTabHandle and ITabHandle.

Example This example displays the tab object's type in an alert box:

```
alert(myTab.getTabType( ));
```

actuate.dashboard.Tab.getTitle

Syntax string Tab.getTitle( )

Returns the tab's title.

Returns String. The title of the tab.

Example This example displays the tab object's title in an alert box:

```
alert(myTab.getTitle( ));
```

Class actuate.data.Filter

Description Specifies filter conditions to be used by other classes when processing data. A filter has three components: a column, an operator, and a value or set of values. The condition is expressed as "value1 operator value2". For some operators, like "IN", the expression will be "value1 IN value2" where value2 is an array of strings.

Format numbers and date/time values in a locale neutral format, for example "2.5" or "09/31/2008 01:02:03 AM".

Constructor

Syntax

```
actuate.data.Filter(string columnName, string operator, string[ ] value1,
    string[ ] value2)
```

Constructs a filter object.

Parameters

columnName
String. The column name.

operator
String. The operator can be any operator. Table 4-10 lists the valid filter operators and the number of arguments to pass to the constructor or setValues( ).

Table 4-10 Filter operators

Operator	Description	Number of arguments
BETWEEN	Between an inclusive range	2
BOTTOM_N	Matches the bottom n values	1
BOTTOM_PERCENT	Matches the bottom percent of the values	1
EQ	Equal	1
FALSE	Matches false Boolean values	0
GREATER_THAN	Greater than	1
GREATER_THAN_OR_EQUAL	Greater than or equal	1
IN	Matches any value in a set of values	1+
LESS_THAN	Less than	1
LESS_THAN_OR_EQUAL	Less than or equal	1
LIKE	Search for a pattern	1

(continues)

Table 4-10 Filter operators (continued)

Operator	Description	Number of arguments
MATCH	Equal	1
NOT_BETWEEN	Not between an inclusive range	2
NOT_EQ	Not equal	1
NOT_IN	Does not match any value in a set of values	1+
NOT_LIKE	Search for values that do not match a pattern	1
NOT_MATCH	Not equal	1
NOT_NULL	Is not null	0
NULL	Is null	0
TOP_N	Matches the top n values	1
TOP_PERCENT	Matches the top percent of the values	1
TRUE	Matches true Boolean values	0

value1
String or array of strings. The first value to compare to the column value for the BETWEEN or NOT_BETWEEN operators.

value2
String or array of strings. This parameter is only required for the BETWEEN or NOT_BETWEEN operators.

Example To select all of the rows matching a list of countries in their country fields, use code similar to the following:

```
var filter = new actuate.data.Filter("COUNTRY",
   actuate.data.Filter.IN,["Canada" , "USA", "UK", "Australia"]);
```

To create a filter to display only entries with a CITY value of NYC, use the following code:

```
var cityfilter = new actuate.data.Filter("CITY",
   actuate.data.Filter.EQ, "NYC");
```

Function summary

Table 4-11 lists actuate.data.Filter functions.

Table 4-11 actuate.data.Filter functions

Function	Description
getColumnName( )	Returns the column name
getOperator( )	Returns the filter operator
getValues( )	Returns the value or values of the filter
setColumnName( )	Sets the name of the column to filter
setOperator( )	Sets the operator for the filter
setValues( )	Sets string values for the filter

actuate.data.Filter.getColumnName

Syntax string Filter.getColumnName( )

Returns the column name.

Returns String. The name of the column.

Example This example retrieves the name of the column:

```
function retrieveColumnName(myFilter){
   var colname = myFilter.getColumnName( );
   return colname;
}
```

actuate.data.Filter.getOperator

Syntax string Filter.getOperator( )

Returns the filter operator.

Returns String. Table 4-10 lists the legal filter operator values.

Example This example retrieves the name of the filter operator:

```
function retrieveFilterOperator(myFilter){
   var myOp = myFilter.getOperator( );
   return myOp;
}
```

actuate.data.Filter.getValues

Syntax string Filter.getValues( )

string[ ] Filter.getValues( )

Returns the evaluated results of this filter. When the filter is constructed or set with a single argument, the returned value corresponds to the single argument. When two arguments or an array are set in the filter, the return value is an array of values.

Returns String or array of strings. Returns the value or values from the filter.

Example This example retrieves the name of the filter operator:

```
function retrieveValues(myFilter){
   var myVals = myFilter.getValues( );
   return myVals;
}
```

actuate.data.Filter.setColumnName

Syntax void Filter.setColumnName(columnName)

Sets the name of the column to filter.

Parameters **columnName**
String. The column name.

Example This example sets the name of the column to filter to Sales:

```
function setFilterSales( myfilter ){
   myfilter.setColumnName("Sales");
}
```

actuate.data.Filter.setOperator

Syntax void Filter.setOperator(string operator)

Sets filter operator. The operator determines the comparison made between the data in the column and the value or values set in the filter.

Parameters **operator**
String. The operator can be any valid operator. Table 4-10 lists the valid filter operators and the number of arguments to pass to Filter.setValues( ).

Example This example sets the filter to retrieve the bottom five values:

```
function setFilterBot5( ){
   myfilter.setOperator("BOTTOM_N");
   myfilter.setValues("5");
}
```

actuate.data.Filter.setValues

Syntax void Filter.setValues(string value)

void Filter.setValues(string value1, string value2)

void Filter.setValues(string[ ] values)

Sets string values for the filter to compare to the data in the column according to the operator. Table 4-10 lists the valid filter operators and the values they use. Takes either one or two values, or one array of values.

Parameters **value**

String. The value to compare to the column value.

value1

String. The first value to compare to the column value for the BETWEEN operator.

value2

String. The second value to compare to the column value for the BETWEEN operator.

values

Array of strings. The values to compare to the column value for the IN operator.

Example This example sets the filter to retrieve values between 10 and 35:

```
function setFilter( myfilter ){
   myfilter.setOperator("BETWEEN");
   myfilter.setValues("10","35");
}
```

Class actuate.data.ReportContent

Description The ReportContent class is a container for downloadable report content.

Constructor

Syntax actuate.data.ReportContent(data)

Constructs a ReportContent object.

Parameters **data**
String. Content text.

Function summary

Table 4-12 describes actuate.data.ReportContent functions.

Table 4-12 actuate.data.ReportContent function

Function	Description
getTextContent( )	Returns the text in the downloaded content

actuate.data.ReportContent.getTextContent

Syntax string ReportContent.getTextContent( )

Returns the text in the downloaded content.

Returns String. The text in the downloaded content.

Example To make a callback function that prints back the first line of text from some downloaded content back onto the page, use code similar to the following:

```
function callback(data1){
   var rcontent = data1.ReportContent.getTextContent( );
   var contentArray = rcontent.split("\n");
   var items = contentArray.length
   document.write("<P>\n")
   document.write(listItems.arguments[o] + "\n</P>")
}
```

Class actuate.data.Request

Description Specifies a request for retrieving data and the conditions for that request. This class provides the scope for a request by defining a target element and a range of rows. The scope of the request determines what goes into a ResultSet. Functions that use request can only retrieve ResultSets from report elements that have an explicit bookmark.

Constructor

Syntax actuate.data.Request(string bookmark, integer startRow, integer maxRow)

Constructs a request object that other classes use to retrieve data.

Parameters **bookmark**
String. A bookmark that identifies an Actuate report element. The actuate.data.Request object uses the bookmark to identify the report element to request information from. If null, Request uses the first bookmark. Functions that use request can only retrieve actuate.data.ResultSet objects from report elements that have an explicit bookmark.

startRow
Integer. The numerical index of the requested first row. The smallest value is 0.

maxRow
Integer. The numerical index of the requested last row. 0 indicates no limit.

Function summary

Table 4-13 lists actuate.data.Request functions.

Table 4-13 actuate.data.Request functions

Function	Description
getBookmark( )	Returns the bookmark name
getColumns( )	Returns the column names
getFilters( )	Returns filters defined in this data condition
getMaxRows( )	Returns the max row number
getSorters( )	Returns sorters defined in this data condition
getStartRow( )	Returns the start row number
setBookmark( )	Sets the bookmark name
setColumns( )	Sets the columns to return

(continues)

Table 4-13 actuate.data.Request functions (continued)

Function	Description
setFilters( )	Sets the filters for the returned data
setMaxRows( )	Sets the max row number
setSorters( )	Sets the sorters for the returned data
setStartRow( )	Sets the start row number

actuate.data.Request.getBookmark

Syntax string Request.getBookmarkName( )

Returns the bookmark name for this request.

Returns String. The bookmark used in the request object's constructor.

Example This example retrieves the bookmark set in the myRequest object:

```
return myRequest.getBookmarkName( );
```

actuate.data.Request.getColumns

Syntax string[ ] Request.getColumns( )

Returns a list of column names that match the request.

Returns Array of strings. The column names.

Example This example retrieves the first, third, and fifth column names from the request object myRequest:

```
function get135Columns(myRequest){
   var columns = myRequest.getColumns( );
   return columns[0];
   return columns[2];
   return columns[4];
}
```

actuate.data.Request.getFilters

Syntax actuate.data.Filter[ ] Request.getfilters( )

Returns filters set for this request.

Returns Array of actuate.data.Filter objects.

actuate.data.Request.getMaxRows

Syntax integer Request.getMaxRows( )

Returns the maximum number of rows to retrieve.

Returns Integer. The index of the last row in the request. 0 means no limit.

actuate.data.Request.getSorters

Syntax actuate.data.Sorter[ ] Request.getSorters( )

Returns sorters assigned to this request.

Returns Array of actuate.data.Sorter objects.

actuate.data.Request.getStartRow

Syntax Integer Request.getStartRow( )

Returns the index of the starting row as an integer.

Returns Integer. The startRow value. The first row in a column has an index of 0.

actuate.data.Request.setBookmark

Syntax void Request.setBookmark(string bookmark)

Sets the bookmark of the element from which to request values.

Parameters **bookmark**
String. A bookmark.

Example This example sets the bookmark for the myRequest object to the string myRequestStart:

```
function setMyRequestBookmark(myRequest){
   myRequest.setBookmark("myRequestStart");
}
```

actuate.data.Request.setColumns

Syntax void Request.setColumns(string[ ] columns)

Sets the request column names.

Parameters **columns**
An array of strings designating the columns of requested data. Use an array for this argument, even if there is only one value.

actuate.data.Request.setFilters

Syntax void Request.setFilters(actuate.data.Filter[ ] filters)

Adds filters to a request. Filters further refine the set of data provided by a request. Using setFilter removes the previous filters from the request object. All of the filters set in a request are applied when the request is used.

Parameters **filters**
An array of actuate.data.Filter objects or a single actuate.data.Filter object to refine the request. Use an array for this argument, even if there is only one value.

actuate.data.Request.setMaxRows

Syntax void Request.setMaxRows(integer maxrow)

Sets the maximum number of rows to retrieve.

Parameters **maxrow**
Integer. The numerical value of the index for the last row to request. 0 indicates no limit.

Example This example sets the index of the last row for the myRequest request object to 50:

```
myRequest.setMaxRows(50);
```

actuate.data.Request.setSorters

Syntax void Request.setSorts(actuate.data.Sorter[ ] sorters)

Adds sorters to a request to sort the set of data that a request provides. Sorting the data increases the effectiveness of requests by providing the data in a relevant order. Using setSorters removes the previous sorter objects from the request object. All of the sorters set in a request are applied when the request is used.

Sorters are applied in the order that they occur in the array. For example, if the first sorter specifies sorting on a state column and the second sorter specifies sorting on a city column, the result set is sorted by city within each state.

Parameters **sorters**
An array of actuate.data.Sorter objects or a single actuate.data.Sorter object to sort the result of the request. Use an array for this argument, even if there is only one value.

Example This example sets the alphaNumericSorterSet array in myRequest:

```
myRequest.setSorters(alphaNumericSorterSet);
```

actuate.data.Request.setStartRow

Syntax void Request.setStartRow(integer startrow)

Sets the requested first row.

Parameters **startrow**
Integer. The numerical value of the index for the first row to request. The first row in a column has an index of 0.

Example This example sets the index of the first row for the myRequest request object to 10:

```
myRequest.setStartRow(10);
```

Class actuate.data.ResultSet

Description The actuate.data.ResultSet class represents the data retrieved from a report document. The functions in the actuate.data.ResultSet class access the data by row. The actuate.data.ResultSet class keeps an internal reference to the current row and increments the current row with next( ).

Constructor

There is no public constructor for actuate.data.ResultSet. The actuate.DataService.downloadResultSet and actuate.Viewer.downloadResultSet functions instantiate the ResultSet object. Set the reference to the ResultSet object in the callback function. For example, when the result set is used as the input parameter for the callback function, result becomes the label for the ResultSet, as shown below:

```
viewer.downloadResultSet(request, parseRS)
function parseRS(resultset){
   // do something with resultset
}
```

Function summary

Table 4-14 lists actuate.data.ResultSet functions.

Table 4-14 actuate.data.ResultSet functions

Function	Description
getColumnNames( )	Returns the column names
getValue( )	Returns the data by the given column index
next( )	Increments the current row

actuate.data.ResultSet.getColumnNames

Syntax string[ ] Request.getColumnNames( )

Returns a list of column names.

Returns Array of strings. The column names.

Example This example retrieves the first, third, and fifth column names from the ResultSet object myResult:

```
function get135Columns(myResult){
   var columns = myResult.getColumns( );
   return columns[0];
   return columns[2];
   return columns[4];
}
```

actuate.data.ResultSet.getValue

Syntax string ResultSet.getValue(integer columnIndex)

Returns the value of the specified column from the current row. Specify the column by its numerical index. Use next( ) before using getValue( ) to set the cursor to the first record.

Parameters **columnIndex**
Integer. The numerical index of the column from which to retrieve data.

Returns String. The field value.

Example This example returns the value for the column with an index value of 4 from the current row in the ResultSet object myResult:

```
return myResult.getValue(4);
```

actuate.data.ResultSet.next

Syntax boolean next( )

Increments the current row for the ResultSet. When no current row is set, next( ) sets the current row to the first row in the ResultSet. When no next row exists, next( ) returns false.

Returns Boolean. True indicates a successful row increment. False indicates that there are no further rows.

Example This example returns the value for the column with an index value of 4 from all of the rows in the ResultSet object myResult:

```
function getColumn4Rows(myResult){
   var nextrow = myResult.next( );
   while (nextrow){
      return myResult.getValue(4);
      nextrow = myResult.next( );
   }
}
```

Class actuate.data.Sorter

Description Specifies the conditions for sorting data as it is returned by a request or stored temporarily in a local ResultSet object. The sort arranges rows based on the value of a specified column.

Constructor

Syntax actuate.data.Sorter(string columnName, boolean ascending)

Constructs a sorter object.

Parameters **columnName**
String. The name of the column to sort.

ascending
Boolean. True sets sorting to ascending. False sets sorting to descending.

Function summary

Table 4-15 lists actuate.data.Sorter functions.

Table 4-15 actuate.data.Sorter functions

Function	Description
getColumnName( )	Returns the column name
isAscending( )	Returns true if the current sorting is ascending
setAscending( )	Sets the sort order to ascending or descending
setColumnName( )	Sets the column to which this sorter applies

actuate.data.Sorter.getColumnName

Syntax string Sorter.getColumnName( )

Returns the name of the column to sort on.

Returns String. The column name.

Example This example displays an alert box that contains the column name currently being sorted on:

```
function showMyColumnName(mySorter){
   var sortColName = mySorter.getColumnName( );
   alert(sortColName);
}
```

actuate.data.Sorter.isAscending

Syntax boolean Sorter.isAscending( )

Returns true if the current sort order is ascending. Returns false if the current order is descending.

Returns Boolean. True indicates ascending. False indicates descending.

Example This example checks if the current sort order is ascending. When the current sort order is descending, this code sets the order to ascending:

```
function makeAscending(mySort){
   if (mySort.isAscending( )) {
      return;
   } else {
      mySort.setAscending(true);
   }
}
```

actuate.data.Sorter.setAscending

Syntax void Sorter.setAscending(boolean ascending)

Sets the sort order to ascending or descending.

Parameters **ascending**
Boolean. True sets the sort order to ascending. False sets the sort order to descending.

Example This example checks if the current sort order is descending. When the current sort order is ascending, this code sets the order to descending:

```
function makeAscending(mySort){
   if (mySort.isAscending( )) {
      return;
   } else {
      mySort.setAscending(true);
   }
}
```

actuate.data.Sorter.setColumnName

Syntax void Sorter.setColumnName(string columnName)

Sets the column to sort on.

Parameters **columnName**
String. The column name.

Example This example makes the current sorter arrange the result set ascending by the Sales column:

```
function makeAscendingOnSales(mySort){
   mySort.setColumnName("Sales");
   if (mySort.isAscending( )) {
      return;
   } else {
      mySort.setAscending(true);
   }
}
```

Class actuate.DataService

Description Connects to an Actuate web application service to retrieve data from Actuate BIRT reports as a ResultSet.

Constructor

Syntax actuate.DataService(string iportalUrl, actuate.RequestOptions requestOptions)

Constructs a DataService object.

Parameters **iportalUrl**
String. Optional. The URL of an Actuate web application service. The DataService uses the web application service set in actuate.initialize if one is not specified.

requestOptions
actuate.RequestOptions object. Optional. Specifies the request options for the iportal web service connection. The DataService uses the options set in actuate.initialize if one is not specified.

Function summary

Table 4-16 lists actuate.DataService functions.

Table 4-16 actuate.DataService functions

Function	Description
downloadResultSet( )	Retrieves data from a report in a ResultSet object

actuate.DataService.downloadResultSet

Syntax void DataService.downloadResultSet(string datasource, actuate.data.Request request, function callback, function errorCallback)

Returns data from an Actuate BIRT report document managed by an Actuate web application. The actuate.data.ResultSet object that downloadResultSet( ) returns is used by the callback function.

Parameters **datasource**
String. The repository path and name of the file from which to retrieve data.

request
actuate.data.Request object. Specifies the request for the report.

callback
Function. The callback function to use after the ResultSet finishes downloading. This function must take the returned ResultSet object as an input parameter.

errorCallback
Function. Optional. The function to call when an error occurs. The possible errors are actuate.Exception objects. The errorCallback( ) function must take an exception as an argument.

Example This example retrieves a result set as specified by the myRequest request object, and calls the makeAscendingSales function, which must take a actuate.data.ResultSet object as an input parameter:

```
var myRequest = new actuate.data.Request("Top_5_Customers", 1, 0);
var myDataService =
   new actuate.DataService( "http://localhost:8900/iportal" );
myDataService.downloadResultSet("/Public/
   BIRT and BIRT Studio Examples/Customer Dashboard.rptdocument",
   myRequest, makeAscendingSales, errorCallback);
```

Class actuate.Exception

Description A container for an uncategorized exceptions that also supports specific exceptions. Exception provides an object to pass to a callback function or event handler when an exception occurs. The Exception object contains references to the exception's origin, description, and messages.

Constructor

The Exception object is constructed when unspecified exceptions occur. The exceptions are divided into three types, which determine the contents of the Exception object. These types are:

- ERR_CLIENT: Exception type for a client-side error
- ERR_SERVER: Exception type for a server error
- ERR_USAGE: Exception type for a JSAPI usage error

Function summary

Table 4-17 lists actuate.Exception functions.

Table 4-17 actuate.Exception functions

Function	Description
getDescription( )	Returns details of the exception
getErrCode( )	Returns error code for server-side exceptions
getMessage( )	Returns a short message about the exception
getType( )	Returns the type of exception error
isExceptionType( )	Confirms exception type

actuate.Exception.getDescription

Syntax string Exception.getDescription( )

Returns exception details as provided by the Server, Client, and User objects.

Returns String. A detailed description of the error. Information is provided according to the type of exception generated, as shown below:

- Server error: The SOAP string
- Client error: For the Firefox browser, a list comprised of fileName+number+stack
- Usage error: Any values set in the object generating the exception

Example This example displays the server error description in an alert box:

```
alert("Server error: " + Exception.getDescription( ));
```

actuate.Exception.getErrCode

Syntax string Exception.getErrCode( )

Returns the error code for server exceptions.

Returns String. A server error code.

Example This example displays the server error code in an alert box:

```
alert("Server error: " + Exception.getErrCode( ));
```

actuate.Exception.getMessage

Syntax string Exception.getMessage( )

Returns a short message about the exception. This message is set for an actuate.Exception object with the actuate.Exception.initJSException( ) function.

Returns String. A server error code.

Example This example displays the error′s short message code in an alert box:

```
alert("Error Message: " + Exception.getMessage( ));
```

actuate.Exception.getType

Syntax string Exception.getType( )

Returns the type of the exception:

- ERR_CLIENT: Exception type for a client-side error
- ERR_SERVER: Exception type for a server error
- ERR_USAGE: Exception type for a Actuate JavaScript API usage error

Returns String. A server error code.

Example This example displays the error type in an alert box:

```
alert("Error type: " + Exception.getType( ));
```

actuate.Exception.isExceptionType

Syntax boolean Exception.isExceptionType(object exceptionType)

Compares the input object to the exception contained in this actuate.Exception object to the exceptionType object argument.

Parameters **exceptionType**

Object. Either an Exception object, such as an instance of actuate.Viewer.ViewerException, or the name of an Exception class as a string.

Returns Boolean. Returns true if the exception contained in this actuate.Exception object matches the exceptionType object argument.

Example To alert the user when the exception e is a usage error, use code similar to the following:

```
if (e.isExceptionType(actuate.exception.ERR_USAGE)){
   alert('Usage error occurred!');
}
```

Class actuate.Parameter

Description The actuate.Parameter class retrieves and displays Actuate BIRT report parameters in an HTML container. Users can interact with the parameters on the page and pass parameter values to an actuate.Viewer object, but not to the server directly.

The actuate.Parameter class displays the parameters by page. The actuate.parameters.navigate( ) function changes the page display or changes the current position on the page.

Constructor

Syntax actuate.Parameter(string container)

Constructs a parameter object for a page, initializing the parameter component.

Parameters **container**
String. The name of the HTML element that displays the rendered parameter component or a container object. The constructor initializes the parameter component but does not render it.

Function summary

Table 4-18 lists actuate.Parameter functions.

Table 4-18 actuate.Parameter functions

Function	Description
downloadParameters( )	Returns an array of ParameterDefinition objects
downloadParameterValues( )	Returns an array list of ParameterValue objects
getLayout( )	Returns the parameter layout
getParameterGroupNames( )	Returns the names of the groups of parameters
getReportName( )	Returns the name of the report file
getTransientDocumentName( )	Returns the name of the transient document
hideNavBar( )	Hides the navigation bar
hideParameterGroup( )	Hides report parameters by group
hideParameterName( )	Hides parameters by name
navigate( )	Navigates the parameter page
onUnload( )	Unloads unused JavaScript variables

Table 4-18 actuate.Parameter functions

Function	Description
registerEventHandler( )	Registers an event handler
removeEventHandler( )	Removes an event handler
renderContent( )	Renders the parameter content to the container
setAutoSuggestDelay( )	Sets the autosuggest delay time
setAutoSuggestFetchSize( )	Sets the fetch size of the autosuggestion list
setAutoSuggestListSize( )	Sets the size of the autosuggestion list
setContainer( )	Sets the HTML container for the parameter content
setExpandedGroups( )	Sets the groups to expand by default
setFont( )	Sets the font of the parameter page
setGroupContainer( )	Sets the HTML container for the group
setLayout( )	Sets the parameter layout type
setReadOnly( )	Sets the parameter UI to read-only
setReportName( )	Sets the remote report path and name
setService( )	Sets the Actuate web application service
setShowDisplayType( )	Sets the parameter page to display localized content
submit( )	Submits all the asynchronous operations that the user has requested on this Parameter object and renders the parameter component on the page

actuate.Parameter.downloadParameters

Syntax void Parameter.downloadParameters(function callback)

Retrieves an array of actuate.parameter.ParameterDefinition objects that contain the report parameters for the report and sends the array to the callback function, which must take the array as an input parameter.

Parameters **callback**
Function. The function to execute after the report parameters finish downloading. Parameter.downloadParameters( ) sends an array of actuate.parameter.ParameterDefinition objects to the callback function as an input argument.

Example This example retrieves a set of report parameters and sends them to a callback function.

```
function getChartParams(myParameter){
   myParameter.downloadParameters(callback( ));
}
```

actuate.Parameter.downloadParameterValues

Syntax void Parameter.downloadParameterValues(function callback)

Returns an array of the actuate.parameter.ParameterValue objects for the parameter object. If no values have been set, the parameter object downloads the default values from the server.

Parameters **callback**
Function. The function to execute after the report parameters finish downloading. Parameter.downloadParameterValues( ) sends an array of actuate.parameter.ParameterValue objects to the callback function as an input argument.

Example To download the parameter values and add them to the viewer, the callback function must use the values as an input parameter, as shown in the following code:

```
paramObj.downloadParameterValues(runNext);
function runNext(values){
   viewer.setParameterValues(values);
}
```

actuate.Parameter.getLayout

Syntax string Parameter.getLayout( )

Returns the parameter layout type.

Returns String. The parameter layout, which will match one of the layout constants in actuate.parameter.Constants:

- actuate.parameter.Constants.LAYOUT_NONE
- actuate.parameter.Constants.LAYOUT_GROUP
- actuate.parameter.Constants.LAYOUT_COLLAPSIBLE

Example This example calls getLayout( ) to display the parameter layout type in an alert box:

```
alert(paramObj.getLayout( ));
```

actuate.Parameter.getParameterGroupNames

Syntax string[ ] Parameter.getParameterGroupNames( )

Returns all the group names for the parameter page as an array of strings.

Returns Array of strings. Each string is a group name.

Example This example displays an alert box with the name of the first group for the parameter page:

```
var groupNames = paramObj.getParameterGroupNames( );
alert("First Group Name: " + groupNames[0]);
```

actuate.Parameter.getReportName

Syntax string Parameter.getReportName( )

Returns the name of the report file currently referenced by this Parameter object.

Returns String. The report file name.

Example This example displays an alert box with the name of the report file:

```
alert("Report file: " + paramObj.getReportName( ));
```

actuate.Parameter.getTransientDocumentName

Syntax string Parameter.getTransientDocumentName( )

Returns the name of the transient document generated by running the report currently referenced by this Parameter object.

Returns String.

Example This example displays an alert box with the name of the transient document:

```
alert("Transient document: " +
   paramObj.getTransientDocumentName( ));
```

actuate.Parameter.hideNavBar

Syntax void Parameter.hideNavBar( )

Hides the navigation bar for the parameter component in the LAYOUT_GROUP layout.

Example This example hides the navigation bar:

```
paramObj.hideNavBar( );
alert("Navigation bar is hidden");
```

actuate.Parameter.hideParameterGroup

Syntax void Parameter.hideParameterGroup(string[ ] groupNames)

Hides all report parameters that belongs to a group or to a list of groups.

Parameters **groupNames**
String or array of strings. Hides any groups listed.

Example This example hides the report parameters that belong to the groups that are listed in the myGroups string array:

```
var myGroups = ["Group1", "Group2", "Group3"];
paramObj.hideParameterGroup(myGroups);
alert("Groups are hidden");
```

actuate.Parameter.hideParameterName

Syntax void Parameter.hideParameterName(string[ ] parameterNames)

Hides report parameters as specified by name.

Parameters **parameterNames**
String or array of strings.

Example This example hides the parameters that are listed in the myParams string array:

```
var myParams = ["Parameter1", "Parameter2", "Parameter3"];
paramObj.hideParameterName(myParams);
alert("Parameters are hidden");
```

actuate.Parameter.navigate

Syntax void Parameter.navigate(string containerId, string navTarget)

Changes the current page of the parameter component. The navTarget determines the new location to display the parameter container.

Parameters **containerId**
String. The value of the id parameter for the HTML <div> element that holds the parameter component.

navTarget
String constant. Which navigation button to trigger. Possible values from actuate.parameter.Constants are NAV_FIRST, NAV_PREV, NAV_NEXT, NAV_LAST.

Example This example displays the last page of the parameter component in the HTML <div> element with the myParams ID:

```
function myParamsLast(myParameter){
  myParameter.navigate("myParams", NAV_LAST);
}
```

actuate.Parameter.onUnload

Syntax void Parameter.onUnload( )

Performs garbage collection for the parameter object and unloads JavaScript variables that are no longer needed by Parameter.

Example This example unloads JavaScript variables and displays an alert box:

```
myParameter.onUnload();
alert("JS variables unloaded.");
```

actuate.Parameter.registerEventHandler

Syntax void Parameter.registerEventHandler(actuate.parameter.EventConstants event, function handler)

Registers an event handler to activate for parameter events. This function can assign several handlers to a single event.

Parameters **event**
actuate.parameter.EventConstants. A constant corresponding to a supported event. actuate.Parameter supports the following two events:

- actuate.parameter.EventConstants.ON_CHANGED
- actuate.parameter.EventConstants.ON_SELECTION_CHANGED

handler
Function. The function to execute when the event occurs. The handler must take two arguments: the parameter instance that fired the event and an event object specific to the event type.

Example To register an event handler to catch exceptions, call actuate.Parameter.registerEventHandler using the ON_EXCEPTION constant after creating the viewer object, as shown in the following example:

```
function initParameter( ){
   parameter = new actuate.Parameter("acparameter");
   parameter.registerEventHandler(actuate.parameter.EventConstants
                                  .ON_CHANGED, errorHandler);
}
```

actuate.Parameter.removeEventHandler

Syntax void Parameter.removeEventHandler(actuate.viewer.EventConstants event, function handler)

Removes an event handler.

Parameters **event**
actuate.parameter.EventConstants. A constant corresponding to a supported event. actuate.Parameter supports the following two events:

- actuate.parameter.EventConstants.ON_CHANGED
- actuate.parameter.EventConstants.ON_SELECTION_CHANGED

handler
Function. A handler function registered for the event.

Example To remove an event handler, call actuate.Parameter.removeEventHandler with a legal event constant, as shown in the following example:

```
function cleanupParameter( ){
   parameter.removeEventHandler(actuate.parameter.EventConstants.
                                 ON_CHANGED, errorHandler);
}
```

actuate.Parameter.renderContent

Syntax void Parameter.renderContent(actuate.parameter.ParameterDefinition[ ] paramDefs, function callback)

Renders the parameter component to the container.

Parameters **paramDefs**
Array of actuate.parameter.ParameterDefinition objects.

callback
Function. The function to execute after the rendering is done.

Example This example calls renderContent( ) after hiding parameter groups:

```
function showNoGroups(myParameter){
   myParameter.hideParameterGroup(zipcodes);
   myParameter.renderContent(myParameterArray,
                              cleanupParameter(myParameter));
}
```

actuate.Parameter.setAutoSuggestDelay

Syntax void Parameter.setAutoSuggestDelay(long delay)

Sets the autosuggest delay time.

Parameters **delay**
Long. Interpreted as milliseconds.

Example This example implements a custom autosuggest list. The list is 10 suggestions long and displays 3 suggestions at a time after a delay of 250 milliseconds.

```
function myCustomAutoSuggest(myParameter){
   myParameter.setAutoSuggestFetchSize(10);
   myParameter.setAutoSuggestListSize(3);
   myParameter.setAutoSuggestDelay(250);
}
```

actuate.Parameter.setAutoSuggestFetchSize

Syntax void Parameter.setAutoSuggestFetchSize(integer size)

Sets the fetch size of the autosuggestion list. Autosuggest fetches all suggestions from the server when the fetch size is not set.

Parameters **size**
Integer. The number of suggestions to fetch at a time.

Example This example implements a custom autosuggest list. The list is 10 suggestions long and displays 3 suggestions at a time after a delay of 250 milliseconds.

```
function myCustomAutoSuggest(myParameter){
   myParameter.setAutoSuggestFetchSize(10);
   myParameter.setAutoSuggestListSize(3);
   myParameter.setAutoSuggestDelay(250);
}
```

actuate.Parameter.setAutoSuggestListSize

Syntax void Parameter.setAutoSuggestListSize(integer size)

Sets the length of the autosuggestion list. Autosuggest shows all of the suggestions from the server when the list length is not set.

Parameters **size**
Integer. The number of suggestions to display.

Example This example implements a custom autosuggest list. The list is 10 suggestions long and displays 3 suggestions at a time after a delay of 250 milliseconds.

```
function myCustomAutoSuggest(myParameter){
   myParameter.setAutoSuggestFetchSize(10);
   myParameter.setAutoSuggestListSize(3);
   myParameter.setAutoSuggestDelay(250);
}
```

actuate.Parameter.setContainer

Syntax void Parameter.setContainer(string containerId)

Sets the HTML element container for the parameter content.

Parameters **containerID**
String. The name of the HTML element that displays the group of rendered parameter components.

Example This example sets the container where the parameter components render:

```
paramObj.setContainer("leftpane");
```

actuate.Parameter.setExpandedGroups

Syntax void Parameter.setExpandedGroups(groupNames)

Defines a set of groups that are expanded by default.

Parameters **groupNames**
Array of strings. The group names to expand by default.

Example This example sets the "Motorcycles", "Trucks", and "Airplanes" groups as expanded by default:

```
var myGroups = new Array["Motorcycles", "Trucks", "Airplanes"];
paramObj.setExpandedGroups(myGroups);
```

actuate.Parameter.setFont

Syntax void Parameter.setFont(string fontStyleString)

Sets the font of the parameter page content after the page is rendered.

Parameters **fontStyleString**
String. The name of a font.

Example This example sets the font to Arial for the parameters page:

```
paramObj.setFont("arial");
```

actuate.Parameter.setGroupContainer

Syntax void Parameter.setGroupContainer(string[ ] groupNames, string containerId)

Sets the HTML element container for the provided group. All parameter objects listed in groupNames are assigned to the container.

Parameters **groupNames**
Array of strings. The group names to be assigned.

containerID
String. The name of the HTML element that displays the group of rendered parameter components.

Example This example assigns the group names in the myGroups string array to the leftpane HTML element:

```
var myGroups = ["Group1", "Group2", "Group3"];
paramObj.setGroupContainer(myGroups, "leftpane");
```

actuate.Parameter.setLayout

Syntax void Parameter.setLayout(string layoutName)

Sets the parameter layout.

Parameters **layoutName**
String constant. Possible values are:

- actuate.parameter.Constants.LAYOUT_GROUP
- actuate.parameter.Constants.LAYOUT_NONE
- actuate.parameter.Constants.LAYOUT_COLLAPSIBLE

Example This example sets the parameter object's layout type to LAYOUT_COLLAPSIBLE:

```
paramObj.setLayout("LAYOUT_COLLAPSIBLE");
```

actuate.Parameter.setReadOnly

Syntax void Parameter.setReadOnly(boolean readOnly)

Sets the parameters to read-only.

Parameters **readOnly**
Boolean. True indicates that the parameters are read-only.

Example This example makes the parameters read-only:

```
paramObj.setReadOnly(true);
```

actuate.Parameter.setReportName

Syntax void Parameter.setReportName(string reportFile)

Sets the report file from which to get report parameters.

Parameters **reportFile**
String. The report file path and name. The report file can be a report design file or a report document file.

Example To set the name using an HTML input tag with an ID of Selector, use the following code:

```
myViewer.setReportName(document.getElementById("Selector").value);
```

actuate.Parameter.setService

Syntax void Parameter.setService(string iPortalURL, actuate.RequestOptions requestOptions)

Sets the target service URL to which the Parameter object links. If the service URL is not set, this Parameter object links to the default service URL set on the actuate object.

Parameters **iPortalURL**
String. The target Actuate web application URL.

requestOptions
actuate.RequestOptions object. Optional. requestOptions defines URL parameters to send with the authentication request, such as the iHub URL, Encyclopedia volume, or repository type. The URL can also include custom parameters.

Example This example sets the URL for the Actuate iPortal web application service:

```
paramObj.setService("http://localhost:8700/
                              iportal",myRequestOptions);
```

actuate.Parameter.setShowDisplayType

Syntax void Parameter.setShowDisplayType(boolean showDisplayType)

Sets whether localized data is shown or not.

Parameters **showDisplayType**
Boolean. True indicates that localized data is shown.

Example This example hides localized data:

```
paramObj.setShowDisplayType(false);
paramObj.submit(alert("Localized data hidden.");
```

actuate.Parameter.submit

Syntax void Parameter.submit(function callback)

Submits requests to the server for the report parameters. When this function is called, an AJAX request is triggered to submit all the operations. When the server finishes the processing, it returns a response and the results are rendered on the page in the parameter container.

Parameters **callback**
Function. The function to execute after the asynchronous call processing is done.

Example This example calls submit( ) after hiding localized data:

```
paramObj.setShowDisplayType(false);
paramObj.submit(alert("Localized data hidden."));
```

Class actuate.parameter.Constants

Description Global constants used for Parameter class. Table 4-19 lists the constants used for the parameter class.

Table 4-19 Actuate iPortal JavaScript API parameter constants

Event	Description
ERR_CLIENT	Constants used to tell JSAPI user that there was a client-side error
ERR_SERVER	Constants used to tell JSAPI user that there was a server-side error
ERR_USAGE	Constants used to tell JSAPI user that there was a usage API error
LAYOUT_COLLAPSIBLE	Constants to set layout of parameter component to collapsible group
LAYOUT_GROUP	Constants to set layout of parameter component to group
LAYOUT_NONE	Constants to set layout of parameter component to none
NAV_FIRST	Constants to programmatically control the first page navigation link
NAV_LAST	Constants to programmatically control the last page navigation link
NAV_NEXT	Constants to programmatically control the next page navigation link
NAV_PREV	Constants to programmatically control the previous page navigation link

Class actuate.parameter.ConvertUtility

Description actuate.parameter.ConvertUtility encodes multiple actuate.parameter.ParameterValue objects into an array of generic objects. For multi-clue or ad hoc parameters, use the array of generic objects as the input parameter for actuate.Viewer.setParameterValues.

Constructor

Syntax

```
actuate.parameter.ConvertUtility(actuate.parameter.ParameterValue[ ]
    aParamVals)
```

Constructs a new ConvertUtility object.

Parameters **aParamVals**

Array of actuate.parameter.ParameterValue objects to convert.

Function summary

Table 4-20 lists actuate.parameter.ConvertUtility functions.

Table 4-20 actuate.parameter.ConvertUtility functions

Function	Description
convert( )	Converts the ParameterValues to an array of generic objects
convertData( )	Converts locale-neutral parameter values to the user's login locale
getParameterMap( )	Returns the ParameterValues as an associative array
getParameterValues( )	Returns an array of ParameterValues

actuate.parameter.ConvertUtility.convert

Syntax

```
void ConvertUtility.convert(function callback)
```

Converts ParameterValues into an array of generic objects. The callback function takes the array as an argument.

Parameters **callback**

Function. The callback function to call after converting the results. The callback function must take the generic array of objects as an argument.

Example This example stores the name-value pair array for myParamValues in a variable called nameValueArray:

```
var nameValueArray = new Array( );
var converter = new actuate.ConvertUtility(myParamValues)
converter.convert(callback);
function callback (values){
   nameValueArray = values;
}
```

actuate.parameter.ConvertUtility.convertDate

Syntax void ConvertUtility.convertDate(function callback)

Converts locale-neutral parameter values to the user's login locale.

Parameters **callback**
Function. An optional function to call when this function completes. The callback function receives an array of actuate.parameter.ParameterValue objects as a parameter.

Example This example converts the name-value pair array for myParamValues and stores the results in a variable called nameValueArray:

```
var nameValueArray = new Array( );
var converter = new actuate.ConvertUtility(myParamValues)
converter.convertDate(callback);
function callback (values){
   nameValueArray = values;
}
```

actuate.parameter.ConvertUtility.getParameterMap

Syntax object ConvertUtility.getParameterMap( )

Returns the parameters as an associative array. This function makes the name of each parameter an object property and sets the value of that property to the associated parameter value.

Returns Object.

Example This example stores the associative array for myParamValues in a variable called nameValueArray:

```
var paramMap = new Object( );
var converter = new actuate.ConvertUtility(myParamValues)
paramMap = converter.getParameterMap( );
```

actuate.parameter.ConvertUtility.getParameterValues

Syntax actuate.parameter.ParameterValue[ ] ConvertUtility.getParameterValues( )

Returns the array of ParameterValue objects.

Returns Array of actuate.parameter.ParameterValue objects.

Example This example stores the array of ParameterValue objects for myParamValues in a variable called paramValues:

```
var paramValues = new Array( );
var converter = new actuate.ConvertUtility(myParamValues)
paramValues = converter.getParameterMap( );
```

Class actuate.parameter.EventConstants

Description Defines the supported event constants for parameters. Table 4-21 lists the parameter event constants.

Table 4-21 Actuate JavaScript API parameter event constants

Event	Description
ON_CHANGE_COMPLETED	Event name triggered when the action is complete and no internal actions are triggered automatically. For example, when a cascading parameter is changed, its child parameter is changed automatically. This event is triggered when its child parameters are updated. The event handler takes the following arguments: ■ actuate.Parameter: parameter component for which the event occurred
ON_CHANGED	Event triggered when a changed event occurs. For example, this event triggers if the value of a parameter control changes. The event handler takes the following arguments: ■ actuate.Parameter: parameter component for which the event occurred
ON_EXCEPTION	Event triggered when an exception occurs. The event handler must take an actuate.Exception object as an input argument. The Exception object contains the exception information.
ON_SELECTION_CHANGED	Event triggered when a selection change occurs. For example, this event triggers if the value of a parameter list control changes. The event handler must take an actuate.Parameter object as an input argument. This input argument is the parameter component for which the event occurred.
ON_SESSION_TIMEOUT	Session time-out event. Whenever a session time-out event occurs and the user tries to perform any operation on parameter component, a prompt dialog will be shown to ask whether the user wants to log in again or not. If the user clicks yes, the ON_SESSION_TIMEOUT event will be fired. If no handler has been registered for this event, a default built-in login dialog will be displayed. The event handler takes the following arguments: ■ actuate.Parameter: component for which the event occurred

Class actuate.parameter.NameValuePair

Description The NameValuePair object contains a display name associated with a value. The actuate.parameterDefinition.setSelectNameValueList( ) function takes an array of actuate.parameter.NameValuePair objects to use in a selection list. In this way, a ParameterDefinition can display a list of names and map them to values used internally. For example, set the name "My Default Country" for a NameValuePair to display "My Default Country" in the drop-down list in the interface, and set the value to "United States" internally for a US user.

Constructor

Syntax actuate.parameter.NameValuePair(string name, string value)

Constructs a new NameValuePair object.

Parameters **name**

String. The name to display in the selection list.

value

String. The value that selecting the name sets internally.

Function summary

Table 4-22 lists actuate.parameter.NameValuePair functions.

Table 4-22 actuate.parameter.NameValuePair functions

Function	Description
getName( )	Gets the name for this NameValuePair
getValue( )	Gets the value for this NameValuePair
setName( )	Sets the name for this NameValuePair
setValue( )	Sets the value for this NameValuePair

actuate.parameter.NameValuePair.getName

Syntax string NameValuePair.getName( )

Returns the name for this NameValuePair.

Returns String.

Example This sample code returns the name component of the myNVPair NameValuePair object:

```
alert("Name component is " + myNVPair.getName( ));
```

actuate.parameter.NameValuePair.getValue

Syntax string NameValuePair.getValue( )

Returns the value for this NameValuePair.

Returns String.

Example This sample code returns the value component of the myNVPair NameValuePair object:

```
alert("Value component is " + myNVPair.getValue( ));
```

actuate.parameter.NameValuePair.setName

Syntax void NameValuePair.setName(string name)

Sets the name for the NameValuePair.

Parameters **name**
String.

Example This sample code sets the name component of the myNVPair NameValuePair object to "My hometown":

```
myNVPair.setName("My hometown");
```

actuate.parameter.NameValuePair.setValue

Syntax void NameValuePair.setValue(string value)

Sets the value for the NameValuePair.

Parameters **value**
String.

Example This sample code sets the value component of the myNVPair NameValuePair object to "Cleveland":

```
myNVPair.setValue("Cleveland");
```

Class actuate.parameter.ParameterData

Description The ParameterData class is a high-level wrapper for an actuate.parameter.ParameterDefinition object.

Constructor

Syntax

```
string actuate.parameter.ParameterData(string reportName,
    actuate.parameter.ParameterDefinition pd)
```

Constructs a new ParameterData object.

Parameters

reportName
String. The name of the report where the parameter definition originates.

pd
actuate.parameter.ParameterDefinition object. The parameter definition set for this ParameterData object.

Function summary

Table 4-23 lists the actuate.parameter.ParameterData functions.

Table 4-23 actuateparameter.ParameterData functions

Function	Description
getCascadingParentValues( )	Returns the cascading parent value
getChildData( )	Returns the child ParameterData object
getControlType( )	Returns the controlType UI value
getCurrentValue( )	Returns the current UI value set by the UI control
getDefaultValue( )	Returns the default value for this ParameterData object
getHelpText( )	Returns the help text for this ParameterData object
getNameValueList( )	Returns the list of name-value pairs for this ParameterData object
getParameterName( )	Returns the parameter name for this ParameterData object
getParentData( )	Returns the parent ParameterData object
getPickList( )	Returns the pick list for the child ParameterData object

Table 4-23 actuateparameter.ParameterData functions

Function	Description
getPromptText( )	Returns the prompt text for this ParameterData object
getSuggestionList( )	Returns the filter-based suggestion list for this ParameterData object
isAdhoc( )	Returns true when this parameter is ad hoc
isCascadingParameter( )	Returns true when this parameter is a cascading parameter
isDynamicFilter( )	Returns true when this parameter is a dynamic filter
isMultiList( )	Returns true when this parameter is a multi-list
isRequired( )	Returns true when this parameter is required
setChildData( )	Indicates that the parameter data contains a child
setCurrentValue( )	Sets the UI value of the UI control
setParentData( )	Indicates that the parameter data contains a parent
setWebService( )	Defines a web service to send SOAP messages

actuate.parameter.ParameterData .getCascadingParentValues

Syntax

```
actuate.parameter.ParameterValue[ ]
   ParameterData.getCascadingParentValues(
   actuate.parameter.ParameterValue[ ] parentValues)
```

Returns the cascading parent value.

Parameters **parentValues**
An array of actuate.parameter.ParameterValue objects. This array is the one to be populated.

Returns An array of actuate.parameter.ParameterValue objects. This is the input array populated with the cascading parent values.

Example This sample code returns a storage array of actuate.parameter.ParameterValue objects representing the cascading parent values:

```
var parentValues = new Array( );
parentValues = myParamData.getCascadingParentValues(parentValues);
```

actuate.parameter.ParameterData.getChildData

Syntax actuate.parameter.ParameterData ParameterData.getChildData( )

Returns the child ParameterData object.

Returns actuate.parameter.ParameterData object.

Example This example assigns the child ParameterData object to a myChildData variable:

```
var myChildData = myParameterData.getChildData( );
```

actuate.parameter.ParameterData.getControlType

Syntax string ParameterData.getControlType( )

Returns the controlType UI value for this ParameterData object.

Returns String. The controlType UI value. Legal controlType UI values are:

- null
- AutoSuggest
- ControlRadioButton
- ControlList
- ControlListAllowNew
- ControlCheckBox

Example This sample code displays the controlType UI value for the myParamData object in an alert box:

```
alert(myParamData.getControlType( ));
```

actuate.parameter.ParameterData.getCurrentValue

Syntax actuate.parameter.ParameterValue ParameterData.getCurrentValue( )

Returns the current UI value set by the UI control.

Returns actuate.parameter.ParameterValue. Returns null when the UI control has not set a value.

Example This sample code assigns the current UI value to the myCurrVal variable:

```
var myCurrVal = myParameterData.getCurrentValue( );
```

actuate.parameter.ParameterData.getDefaultValue

Syntax string ParameterData.getDefaultValue( )

Returns the default value for this ParameterData object.

Returns String. The default value. Returns null when the default value is null.

Example This sample code displays the default value for myParamData in an alert box:

```
alert(myParamData.getDefaultValue( ));
```

actuate.parameter.ParameterData.getHelpText

Syntax string ParameterData.getHelpText( )

Returns the help text for this ParameterData object.

Returns String. The help text.

Example This example displays the help text for the myParamData object in an alert box:

```
alert(myParamData.getHelpText( ));
```

actuate.parameter.ParameterData.getNameValueList

Syntax actuate.parameter.NameValuePair[ ] ParameterData.getNameValueList( )

Returns the list of name-value pairs for this ParameterData object.

Returns Array of actuate.parameter.NameValuePair objects.

Example This example stores the array of NameValuePair objects for the myParamValues object in a variable called myNVList:

```
var myNVList = new Array( );
myNVList = myParamValues.getNameValueList( );
```

actuate.parameter.ParameterData.getParameterName

Syntax string ParameterData.getParameterName( )

Returns the parameter name for this ParameterData object.

Returns String. The parameter name.

Example This sample code displays the parameter name for the myParamData object in an alert box:

```
alert(myParamData.getParameterName( ));
```

actuate.parameter.ParameterData.getParentData

Syntax actuate.parameter.ParameterData ParameterData.getParentData( )

Returns the parent ParameterData object.

Returns actuate.parameter.ParameterData object.

Example This sample code assigns this ParameterData object's parent ParameterData object to the myParentData variable:

```
var myParentData = myParameterData.getParentData( );
```

actuate.parameter.ParameterData.getPickList

Syntax actuate.parameter.ParameterValue[ ] ParameterData.getPickList(function callback)

Gets the pick list for the child of this parameter data.

Parameters **callback**
Function. An optional function to call when this function completes. This function receives the following parameters:

- An array of actuate.parameter.NameValuePair objects
- An integer that represents the pick list's total leftover count

Returns An array of actuate.parameter.ParameterValue objects.

Example This sample code uses the callback function runNext( ) to display the pick list's total leftover count in an alert box and assigns the array of NameValuePair objects to the pickListNVPairs variable:

```
paramObj.getPickList(runNext);
function runNext(pairs, leftover){
   alert(leftover);
   var pickListNVPairs = new Array( );
   pickListNVPairs = pairs;
}
```

actuate.parameter.ParameterData.getPromptText

Syntax string ParameterData.getPromptText( )

Returns the prompt text for this ParameterData object.

Returns String. The prompt text.

Example This sample code displays the prompt text for the myParamData object in an alert box:

```
alert(myParamData.getPromptText( ));
```

actuate.parameter.ParameterData.getSuggestionList

Syntax string[ ] ParameterData.getSuggestionList(function callback, string filter)

Returns the filter-based suggestion list for this ParameterData object.

Parameters **callback**
Function. An optional function to call when this function completes. This function receives an array of actuate.parameter.NameValuePair objects as a parameter.

filter
String. The filter for the suggestion list.

Example This sample code uses the string "Trucks" to call back function runNext( ) to filter the suggestion list and assigns the filtered NameValuePair objects to the mySuggestions variable:

```
paramObj.getSuggestionList(runNext, "Trucks");
function runNext(suggested){
   var mySuggestions = new Array( );
   mySuggestions = suggested;
}
```

actuate.parameter.ParameterData.isAdhoc

Syntax boolean ParameterData.isAdhoc( )

Returns true when this parameter is an ad hoc parameter.

Returns Boolean. True when this parameter is ad hoc.

Example This example displays the ad hoc status of a parameter in an alert box:

```
alert(paramData.isAdhoc( ));
```

actuate.parameter.ParameterData .isCascadingParameter

Syntax boolean ParameterData.isAdhoc( )

Returns true when this parameter is a cascading parameter.

Returns Boolean. True when this parameter is a cascading parameter.

Example This example displays the cascading parameter status of a parameter in an alert box:

```
alert(paramData.isCascadingParameter( ));
```

actuate.parameter.ParameterData.isDynamicFilter

Syntax boolean ParameterData.isDynamicFilter( )

Returns true when this parameter is a dynamic filter.

Returns Boolean. True when this parameter is a dynamic filter.

Example This example displays the dynamic filter status of a parameter in an alert box:

```
alert(paramData.isDynamicFilter( ));
```

actuate.parameter.ParameterData.isMultiList

Syntax boolean ParameterData.isMultiList( )

Returns true when this parameter is shown as a multi-list UI element.

Returns Boolean. True when this parameter is shown as a multi-list UI element.

Example This example displays the multi-list UI element status of a parameter in an alert box:

```
alert(paramData.isMultiList( ));
```

actuate.parameter.ParameterData.isRequired

Syntax boolean ParameterData.isRequired( )

Returns true when this parameter is required.

Returns Boolean. True when this parameter is required.

Example This example displays the required status of a parameter in an alert box:

```
alert(paramData.isRequired( ));
```

actuate.parameter.ParameterData.setChildData

Syntax void ParameterData.setChildData(actuate.parameter.ParameterData childData)

Adds a child parameter to this parameter.

Parameters **childData**
An actuate.parameter.ParameterData object that contains the child for this ParameterData object.

Example This sample code sets the ParameterData object myChildData as the child of the ParameterData object myParamData:

```
myParamData.setChildData(myChildData);
```

actuate.parameter.ParameterData.setCurrentValue

Syntax void ParameterData.setCurrentValue(actuate.parameter.ParameterValue value)

Sets the UI value of the UI control. When a UI value changes, UIControl calls this method to update the ParameterData object.

Parameters **value**
An actuate.parameter.ParameterValue object set by the UI.

Example This sample code sets the ParameterValue object myValue as the value of the ParameterData object myParamData:

```
myParamData.setCurrentValue(myValue);
```

actuate.parameter.ParameterData.setParentData

Syntax void ParameterData.setParentData(parentData)

Sets a parent ParameterData object, making this ParameterData object its child.

Parameters **parentData**
An actuate.parameter.ParameterData object that contains the parent for this ParameterData object.

Example This sample code sets the ParameterData object myParentData as the parent of the ParameterData object myParamData:

```
myParamData.setParentData(myParentData);
```

actuate.parameter.ParameterData.setWebService

Syntax void ParameterData.setWebService(object webService)

Defines a web service to use to send SOAP messages.

Parameters **webService**
Object. A web service to send SOAP messages.

Class actuate.parameter.ParameterDefinition

Description The ParameterDefinition object contains all of the qualities, values, names, and conditions for a parameter. A ParameterDefinition object can display options to the user and respond to user-generated events. The actuate.Parameter class downloads an array of ParameterDefinition objects with downloadParameters( ). The order of this array is also the order in which the parameters are displayed. Parameters can be grouped to divide the parameters on the page into logical sets under a heading.

This class requires significant memory and bandwidth resources. ParameterValue is much smaller than ParameterDefinition. ParameterValue is the more efficient way to communicate to the server that a parameter value has changed.

Constructor

Syntax actuate.parameter.ParameterDefinition( )

Constructs a new ParameterDefinition object.

Function summary

Table 4-24 lists actuate.parameter.ParameterDefinition functions.

Table 4-24 actuate.parameter.ParameterDefinition functions

Function	Description
getAutoSuggestThreshold( )	Gets the auto suggest threshold value for this ParameterDefinition
getCascadingParentName( )	Gets the cascadingParentName value for this ParameterDefinition
getColumnName( )	Gets the columnName value for this ParameterDefinition
getColumnType( )	Gets the columnType value for this ParameterDefinition
getControlType( )	Gets the controlType value for this ParameterDefinition
getCurrentDisplayName( )	Gets the auto suggest current display name for the current value of this ParameterDefinition
getDataType( )	Gets the dataType value for this ParameterDefinition
getDefaultValue( )	Gets the defaultValue value for this ParameterDefinition

Table 4-24 actuate.parameter.ParameterDefinition functions (continued)

Function	Description
getDefaultValueIsNull( )	Gets a flag if the default value is null for this ParameterDefinition
getDisplayName( )	Gets the displayName value for this ParameterDefinition
getGroup( )	Gets the group value for this ParameterDefinition
getHelpText( )	Gets the helpText value for this ParameterDefinition
getName( )	Gets the name value for this ParameterDefinition
getOperatorList( )	Gets the list of valid operators
getPosition( )	Gets the position value for this ParameterDefinition
getSelectNameValueList( )	Gets the selectNameValueList value for this ParameterDefinition
getSelectValueList( )	Gets the selectValueList value for this ParameterDefinition
isAdHoc( )	Gets the isAdHoc value for this ParameterDefinition
isHidden( )	Gets the isHidden value for this ParameterDefinition
isPassword( )	Gets the isPassword value for this ParameterDefinition
isRequired( )	Gets the isRequired value for this ParameterDefinition
isViewParameter( )	Gets the isViewParameter value for this ParameterDefinition
setAutoSuggestThreshold( )	Sets the auto suggest threshold value for this ParameterDefinition
setCascadingParentName( )	Sets the cascadingParentName value for this ParameterDefinition
setColumnName( )	Sets the columnName value for this ParameterDefinition
setColumnType( )	Sets the columnType value for this ParameterDefinition

(continues)

Table 4-24 actuate.parameter.ParameterDefinition functions (continued)

Function	Description
setControlType( )	Sets the controlType value for this ParameterDefinition
setCurrentDisplayName( )	Sets the current display name for this ParameterDefinition
setDataType( )	Sets the dataType value for this ParameterDefinition
setDefaultValue( )	Sets the defaultValue value for this ParameterDefinition
setDefaultValueIsNull( )	Sets the defaultValue to null for this ParameterDefinition
setDisplayName( )	Sets the displayName value for this ParameterDefinition
setGroup( )	Sets the group value for this ParameterDefinition
setHelpText( )	Sets the helpText value for this ParameterDefinition
setIsAdHoc( )	Sets the isAdHoc value for this ParameterDefinition
setIsHidden( )	Sets the isHidden value for this ParameterDefinition
setIsPassword( )	Sets the isPassword value for this ParameterDefinition
setIsRequired( )	Sets the isRequired value for this ParameterDefinition
setIsViewParameter( )	Sets the isViewParameter value for this ParameterDefinition
setName( )	Sets the name value for this ParameterDefinition
setPosition( )	Sets the position value for this ParameterDefinition
setSelectNameValueList( )	Sets the selectNameValueList value for this ParameterDefinition
setSelectValueList( )	Sets the selectValueList value for this ParameterDefinition

actuate.parameter.ParameterDefinition .getAutoSuggestThreshold

Syntax integer ParameterDefinition.getAutoSuggestThreshold( )

Gets the autosuggest threshold value for this ParameterDefinition. The autosuggest threshold determines the number of characters a user types in before they are given suggestions from autosuggest.

Returns Integer.

Example To store the autosuggest threshold of the parameter definition paramdef in a variable called threshold, use code similar to the following:

```
var threshold = paramdef.getAutoSuggestThreshold( );
```

actuate.parameter.ParameterDefinition .getCascadingParentName

Syntax string ParameterDefinition.getCascadingParentName( )

Gets the cascadingParentName value for this ParameterDefinition. A cascading parent parameter is only used when one parameter depends upon another.

Returns String.

Example To store the cascading parent name of the parameter definition paramdef in a variable called parentname, use code similar to the following:

```
var parentname = paramdef.getCascadingParentName( );
```

actuate.parameter.ParameterDefinition .getColumnName

Syntax string ParameterDefinition.getColumnName( )

Gets the columnName value for this ParameterDefinition. This setting sets the column to retrieve data from for an ad hoc parameter that performs a query.

This setting has no effect on other types of parameters.

Returns String.

Example To store the column name of the parameter definition paramdef in a variable called columnname, use code similar to the following:

```
var columnname = paramdef.getColumnName( );
```

actuate.parameter.ParameterDefinition .getColumnType

Syntax string ParameterDefinition.getColumnType( )

Gets the columnType value for this ParameterDefinition. This setting sets the data type queried by an ad hoc parameter that performs a query.

This setting has no effect on other types parameters.

Returns String. Possible values are: null, "Currency", "Date", "DateOnly", "Time", "Double", "Integer", "String", "Boolean", "Structure", "Table", and "Unknown".

Example To store the column type of the parameter definition paramdef in a variable called columntype, use code similar to the following:

```
var columntype = paramdef.getColumnType( );
```

actuate.parameter.ParameterDefinition .getControlType

Syntax string ParameterDefinition.getControlType( )

Gets the controlType value for this ParameterDefinition. It determines the form element displayed for the user to set the parameter value.

Returns String. Possible values are: null, "", "ControlRadioButton", "ControlList", "ControlListAllowNew", and "ControlCheckBox".

Example To store the control type string for the parameter definition paramdef in a variable called controltype, use code similar to the following:

```
var controltype = paramdef.getControlType( );
```

actuate.parameter.ParameterDefinition .getCurrentDisplayName

Syntax string ParameterDefinition.getCurrentDisplayName( )

Gets the current display name for this ParameterDefinition.

Returns String.

Example To store the current display name of the parameter definition paramdef in a variable called displayname, use code similar to the following:

```
var displayname = paramdef.getDisplayName( );
```

actuate.parameter.ParameterDefinition .getDataType

Syntax string ParameterDefinition.getDataType( )

Gets the dataType value for this ParameterDefinition.

Returns String. Possible values are: "Currency", "Date", "DateOnly", "Time", "Double", "Integer", "String", "Boolean", "Structure", "Table", and "Unknown".

Example To store the data type of the parameter definition paramdef in a variable called type, use code similar to the following:

```
var type = paramdef.getDataType( );
```

actuate.parameter.ParameterDefinition .getDefaultValue

Syntax string ParameterDefinition.getDefaultValue( )

Gets the defaultValue value for this ParameterDefinition, if applicable.

Returns String.

Example To store the default value as a string for the parameter definition paramdef in a variable called default, use code similar to the following:

```
var default = paramdef.getDefaultValue( );
```

actuate.parameter.ParameterDefinition .getDefaultValueIsNull

Syntax boolean ParameterDefinition.getDefaultValueIsNull( )

Returns true when the parameter's default value is null.

Returns Boolean.

Example To alert the user that the default value is null for the parameter definition paramdef, use code similar to the following:

```
if (paramdef.getDefaultValueIsNull( )){
   alert('Default value is null!');
}
```

actuate.parameter.ParameterDefinition .getDisplayName

Syntax string ParameterDefinition.getDisplayName( )

Gets the displayName for this ParameterDefinition.

Returns String.

Example To store the displayed name for the parameter definition paramdef in a variable called displayname, use code similar to the following:

```
var displayname = paramdef.getDisplayName( );
```

actuate.parameter.ParameterDefinition.getGroup

Syntax string ParameterDefinition.getGroup( )

Gets the group for this ParameterDefinition, indicating if it is a member of a group.

Returns String. A group name, or null if there is no group.

Example To print the group name for the parameter definition paramdef to the current document, use code similar to the following:

```
document.write(paramdef.getGroup( ));
```

actuate.parameter.ParameterDefinition.getHelpText

Syntax string ParameterDefinition.getHelpText( )

Gets the helpText for this ParameterDefinition.

Returns String. The help text.

Example To store the help text for the parameter definition paramdef in a variable called helptext, use code similar to the following:

```
var helptext = paramdef.getHelpText( );
```

actuate.parameter.ParameterDefinition.getName

Syntax string ParameterDefinition.getName( )

Gets the name for this ParameterDefinition.

Returns String. The parameter name.

Example To store the name for the parameter definition paramdef in a variable called paramname, use code similar to the following:

```
var paramname = paramdef.getName( );
```

actuate.parameter.ParameterDefinition .getOperatorList

Syntax string[ ] ParameterDefinition.getOperatorList( )

Gets the operator list for this ParameterDefinition.

Returns An array of strings containing the operator list.

Example To store the list of operators for the parameter definition paramdef in a variable called ops, use code similar to the following:

```
var ops = new Array( );
ops = paramdef.getOperatorList( );
```

actuate.parameter.ParameterDefinition.getPosition

Syntax Integer ParameterDefinition.getPosition( )

Gets the position in the array for this ParameterDefinition.

Returns Integer.

Example To store the position of the parameter definition paramdef in a variable called position, use code similar to the following:

```
var position = paramdef.getPosition( );
```

actuate.parameter.ParameterDefinition .getSelectNameValueList

Syntax selectNameValueList[ ] ParameterDefinition.getSelectNameValueList( )

Gets the selectNameValueList for this ParameterDefinition. This list applies if the parameter is set with a selection list.

Returns Array of actuate.parameter.NameValuePair objects.

Example To retrieve the name-value pair list for the parameter definition paramdef and put it into a new array, use code similar to the following:

```
var namevalues = new array( );
namevalues = paramdef.getSelectNameValueList( ).slice( );
```

actuate.parameter.ParameterDefinition .getSelectValueList

Syntax string[ ] ParameterDefinition.getSelectValueList( )

Gets the selectValueList for this ParameterDefinition. This list applies when the parameter is set with a selection list.

Returns An array of strings containing the select value list.

Example To retrieve the list of values selectable for the parameter definition paramdef and put it into a new array, use code similar to the following:

```
var selectvalues = new array( );
selectvalues = paramdef.getSelectValueList( ).slice( );
```

actuate.parameter.ParameterDefinition.isAdHoc

Syntax boolean ParameterDefinition.isAdHoc( )

Gets the isAdHoc for this ParameterDefinition.

Returns Boolean. True indicates that this parameter is an ad hoc parameter.

Example To set the default value to null for the parameter definition paramdef if it is an ad hoc parameter, use code similar to the following:

```
if (paramdef.isAdHoc( )){
   paramdef.setDefaultValueIsNull(true);
}
```

actuate.parameter.ParameterDefinition.isHidden

Syntax boolean ParameterDefinition.isHidden( )

Gets the isHidden value for this ParameterDefinition.

Returns Boolean. True indicates that this parameter is hidden.

Example To reveal a parameter with the parameter definition paramdef if it is hidden, use code similar to the following:

```
if (paramdef.isHidden( )){
   paramdef.setIsHidden(false);
}
```

actuate.parameter.ParameterDefinition.isPassword

Syntax boolean ParameterDefinition.isPassword( )

Gets the isPassword value for this ParameterDefinition.

Returns Boolean. True indicates that the parameter is a password.

Example To set the parameter definition paramdef as required if it is a password parameter, use code similar to the following:

```
if (paramdef.isPassword( )){
   paramdef.setIsRequired(true);
}
```

actuate.parameter.ParameterDefinition.isRequired

Syntax boolean ParameterDefinition.isRequired( )

Gets the isRequired value for this ParameterDefinition.

Returns Boolean. True indicates that the parameter is required.

Example To set specific help text for the parameter definition paramdef if it is a required parameter, use code similar to the following:

```
if (paramdef.isRequired( )){
   paramdef.setHelpText("This parameter is required.");
}
```

actuate.parameter.ParameterDefinition .isViewParameter

Syntax boolean ParameterDefinition.isViewParameter( )

Gets the isViewParameter value for this ParameterDefinition.

Returns Boolean. True indicates that the parameter is a view-time parameter. False indicates that the parameter is a run-time parameter.

Example To set specific help text for the parameter definition paramdef if it is a view-time parameter, use code similar to the following:

```
if (paramdef.isViewParameter( )){
   paramdef.setHelpText("This is a view-time parameter.");
}
```

actuate.parameter.ParameterDefinition .setAutoSuggestThreshold

Syntax void ParameterDefinition.setAutoSuggestThreshold(integer threshold)

Sets the autosuggest threshold for this ParameterDefinition. The autosuggest threshold determines the number of characters a user types in before they are given suggestions from autosuggest.

Parameters **threshold**
Integer.

Example To always show the autosuggest dialog for the parameter definition paramdef, use code similar to the following:

```
paramdef.setAutoSuggestThreshold(0);
```

actuate.parameter.ParameterDefinition .setCascadingParentName

Syntax void ParameterDefinition.setCascadingParentName(string cascadingParentName)

Sets the cascadingParentName for this ParameterDefinition. This sets another parameter as this parameter's parent.

Parameters **cascadingParentName**
String.

Example To set the parent name of the parameter definition paramdef to "Clark", use code similar to the following:

```
paramdef.setCascadingParentName("Clark");
```

actuate.parameter.ParameterDefinition .setColumnName

Syntax void ParameterDefinition.setColumnName(string columnName)

Sets the columnName for this ParameterDefinition. Used for queries.

Parameters **columnName**
String.

Example To set the parameter definition paramdef to access the ProductName column, use code similar to the following:

```
paramdef.setColumnName("ProductName");
```

actuate.parameter.ParameterDefinition .setColumnType

Syntax void ParameterDefinition.setColumnType(string columnType)

Sets the columnType for this ParameterDefinition. Used for queries.

Parameters **columnType**
String. Possible values are null, "Currency", "Date", "DateOnly", "Time", "Double", "Integer", "String", "Boolean", "Structure", "Table", and "Unknown".

Example To allow the parameter definition paramdef to interpret a column as untyped data, use code similar to the following:

```
paramdef.setColumnType("Unknown");
```

actuate.parameter.ParameterDefinition .setControlType

Syntax void ParameterDefinition.setControlType(string controlType)

Sets the control type of this ParameterDefinition.

Parameters **controlType**
String. Possible values are null, "", "AutoSuggest", "ControlRadioButton", "ControlList", "ControlListAllowNew", and "ControlCheckBox".

Example To set the parameter definition paramdef to use a control list, use code similar to the following:

```
paramdef.setControlType("ControlList");
```

actuate.parameter.ParameterDefinition .setCurrentDisplayName

Syntax void ParameterDefinition.setCurrentDisplayName(string currentDiplayName)

Sets the displayed name for this parameter.

Parameters **currentDisplayName**
String.

Example To set the display name for the parameter definition paramdef to "Year", use code similar to the following:

```
paramdef.setCurrentDisplayName("Year");
```

actuate.parameter.ParameterDefinition.setDataType

Syntax void ParameterDefinition.setDataType(string dataType)

Sets the dataType for this ParameterDefinition.

Parameters **dataType**
String. Possible values are "Currency", "Date", "DateOnly", "Time", "Double", "Integer", "String", "Boolean", "Structure", "Table", and "Unknown".

Example To set the parameter definition paramdef data type to date, use code similar to the following:

```
paramdef.setDataType("Date");
```

actuate.parameter.ParameterDefinition .setDefaultValue

Syntax void ParameterDefinition.setDefaultValue(string defaultValue)

Sets the default value for this ParameterDefinition.

Parameters **defaultValue**
String.

Example To set the default value of parameter definition paramdef to "2010", use code similar to the following:

```
paramdef.setDefaultValue("2010");
```

actuate.parameter.ParameterDefinition .setDefaultValueIsNull

Syntax void ParameterDefinition.setDefaultValue(boolean value)

When true, sets the default value for this ParameterDefinition to null. Sets the default value to no value in all other cases.

Parameters **value**
Boolean.

Example To set the default value of parameter definition paramdef to null, use code similar to the following:

```
paramdef.setDefaultValueIsNull(true);
```

actuate.parameter.ParameterDefinition .setDisplayName

Syntax void ParameterDefinition.setDisplayName(string displayName)

Sets the name to display on the parameter page for this ParameterDefinition.

Parameters **displayName**
String.

Example To set the displayed name of parameter definition paramdef to "Year", use code similar to the following:

```
paramdef.setDisplayName("Year");
```

actuate.parameter.ParameterDefinition.setGroup

Syntax void ParameterDefinition.setGroup(string group)

Sets the group value for this ParameterDefinition.

Parameters **group**
String.

Example To assign the parameter definition paramdef to the "Customer Details" parameter group, use code similar to the following:

```
paramdef.setGroup("Customer Details");
```

actuate.parameter.ParameterDefinition.setHelpText

Syntax void ParameterDefinition.setHelpText(string helpText)

Sets the helpText value for this ParameterDefinition.

Parameters **helpText**
String.

Example To set specific help text for the parameter definition paramdef if it is a required parameter, use code similar to the following:

```
if (paramdef.isRequired( )){
  paramdef.setHelpText("This parameter is required.");
}
```

actuate.parameter.ParameterDefinition.setIsAdHoc

Syntax void ParameterDefinition.setIsAdHoc(boolean isAdHoc)

Sets this parameter as an ad hoc parameter.

Parameters **isAdHoc**
Boolean. True sets this parameter to ad hoc.

Example To enable the parameter definition paramdef to accept an ad hoc value, use code similar to the following:

```
paramdef.setIsAdHoc(true);
```

actuate.parameter.ParameterDefinition.setIsHidden

Syntax void ParameterDefinition.setIsHidden(boolean isHidden)

Sets the parameter to hidden.

Parameters **isHidden**
Boolean. True hides the parameter.

Example To hide a parameter defined by a parameter definition called paramdef, use code similar to the following:

```
paramdef.setIsHidden(true);
```

actuate.parameter.ParameterDefinition .setIsMultiSelectControl

Syntax void ParameterDefinition.setIsMultiSelectControl(boolean isMultiSelect)

Sets the parameter to accept multiple selected values.

Parameters **isHMultiSelect**
Boolean. True allows multiple selected values to be set for this parameter.

Example To allow a parameter defined by a parameter definition called paramdef to accept multiple selected values, use code similar to the following:

```
paramdef.setIsMultiSelectControl(true);
```

actuate.parameter.ParameterDefinition .setIsPassword

Syntax void ParameterDefinition.setIsPassword(boolean isPassword)

Sets this parameter to treat its value as a password, which hides the input on the page and encrypts the value.

Parameters **isPassword**
Boolean. True indicates a password value.

Example To set the parameter type accepted by the parameter definition paramdef to password, use code similar to the following:

```
paramdef.setIsPassword(true);
```

actuate.parameter.ParameterDefinition.setIsRequired

Syntax void ParameterDefinition.setIsRequired(boolean isRequired)

Sets the parameter to required.

Parameters **isRequired**
Boolean. True indicates a mandatory parameter.

Example To make the parameter defined by the parameter definition paramdef mandatory, use code similar to the following:

```
paramdef.setIsRequired(true);
```

actuate.parameter.ParameterDefinition .setIsViewParameter

Syntax void ParameterDefinition.setIsViewParameter(boolean isViewParameter)

Sets the isViewParameter value for this ParameterDefinition.

Parameters **isViewParameter**
Boolean.

Example To make the parameter defined by the parameter definition paramdef a view-time parameter, use code similar to the following:

```
paramdef.setIsViewParameter(true);
```

actuate.parameter.ParameterDefinition.setName

Syntax void ParameterDefinition.setName(string name)

Sets the name to use internally for this ParameterDefinition.

Parameters **name**
String.

Example To set the internal name of the parameter definition paramdef to Year, use code similar to the following:

```
paramdef.setName("Year");
```

actuate.parameter.ParameterDefinition.setPosition

Syntax void ParameterDefinition.setPosition(integer position)

Sets the position value for this ParameterDefinition. The index indicates the position in the array of the ParameterDefinition.

Parameters **position**
Integer.

Example To shift the parameter definition paramdef down on position in the parameter array, use code similar to the following:

```
paramdef.setPosition(++paramdef.getPosition( ));
```

actuate.parameter.ParameterDefinition .setSelectNameValueList

Syntax void ParameterDefinition.setSelectNameValueList
    (actuate.parameter.NameValuePair[ ] selectNameValueList)

Sets the selectNameValueList value for this ParameterDefinition.

Parameters **selectNameValueList**
Array of actuate.parameter.NameValuePair objects.

Example To set the parameter definition paramdef to select the same name-value list as the parameter definition nparam, use code similar to the following:

```
paramdef.setSelectNameValueList(nparam.getSelectNameValueList( ));
```

actuate.parameter.ParameterDefinition .setSelectValueList

Syntax void ParameterDefinition.setSelectValueList(array[ ] selectValueList)

Sets the selectValueList value for this ParameterDefinition.

Parameters **selectValueList**
Array.

Example To set the parameter definition paramdef to select the values 2007-2009, use code similar to the following:

```
var values = new Array("2007", "2008", "2009");
paramdef.setSelectValueList(values);
```

Class actuate.parameter.ParameterValue

Description ParameterValue is a container for the value of Parameter to be passed to a report for processing. When a user sets a value in the interface, the corresponding ParameterValue must change.

Because ParameterValue is much smaller that ParameterDefinition, it is the recommended means of communicating to the server that a parameter value has changed or passing a parameter value to a viewer element. Sending an entire ParameterDefinition has a larger effect on system performance.

Constructor

Syntax actuate.parameter.ParameterValue( )

Constructs a new ParameterValue object.

Function summary

Table 4-25 lists actuate.parameter.ParameterValue functions.

Table 4-25 actuate.parameter.ParameterValue functions

Function	Description
getColumnName( )	Gets the name of the column in this ParameterValue
getColumnType( )	Gets the data type value of the column for this ParameterValue
getDataType( )	Gets the dataType value for this ParameterValue
getDisplayName( )	Gets the displayed name for this ParameterValue
getGroup( )	Gets the group value for this ParameterValue
getName( )	Gets the name value for this ParameterValue
getPosition( )	Gets the position value for this ParameterValue
getPromptParameter( )	Gets the promptParameter value for this ParameterValue
getValue( )	Gets the value or values for this ParameterValue
getValueIsNull( )	Gets the valueIsNull value for this ParameterValue
isViewParameter( )	Gets the isViewParameter value for this ParameterValue

(continues)

Table 4-25 actuate.parameter.ParameterValue functions (continued)

Function	Description
setColumnName( )	Sets the name of the column in this ParameterValue
setColumnType( )	Sets the data type value of the column for this ParameterValue
setDataType( )	Sets the dataType value for this ParameterValue
setDisplayName( )	Sets the displayed name for this ParameterValue
setGroup( )	Sets the group value for this ParameterValue
setIsViewParameter( )	Sets the isViewParameter value for this ParameterValue
setName( )	Sets the name value for this ParameterValue
setPosition( )	Sets the position value for this ParameterValue
setPromptParameter( )	Sets the promptParameter value for this ParameterValue
setValue( )	Sets the value for this ParameterValue
setValueIsNull( )	Sets the valueIsNull value for this ParameterValue

actuate.parameter.ParameterValue.getColumnName

Syntax string ParameterValue.getColumnName( )

Gets the column name value for this ParameterValue. Columns are supported as part of ad hoc parameters.

Returns String. The name of the column.

Example To store the column name for the parameter value pvalue in a variable called columnname, use code similar to the following:

```
var columnname = pvalue.getColumnName( );
```

actuate.parameter.ParameterValue.getColumnType

Syntax string ParameterValue.getColumnType( )

Gets the data type value of the column for this ParameterValue. Columns are supported as part of ad hoc parameters.

Returns String. Possible values are null, "", "Currency", "Date", "DateOnly", "Time", "Double", "Integer", "String", "Boolean", "Structure", "Table", and "Unknown".

Example To store the column type for the parameter value pvalue in a variable called columntype, use code similar to the following:

```
var columntype = pvalue.getColumnType( );
```

actuate.parameter.ParameterValue.getDataType

Syntax string ParameterValue.getDataType( )

Gets the dataType value for this ParameterValue.

Returns String. Possible values are null, "", "Currency", "Date", "DateOnly", "Time", "Double", "Integer", "String", "Boolean", "Structure", "Table", and "Unknown".

Example To store the data type for the parameter value pvalue in a variable called type, use code similar to the following:

```
var type = pvalue.getDataType( );
```

actuate.parameter.ParameterValue.getDisplayName

Syntax string ParameterValue.getDisplayName( )

Gets the displayed name for this ParameterValue.

Returns String. The displayed name

Example To store the displayed name of the parameter value pvalue in a variable called displayedName, use code similar to the following:

```
var displayedName = pvalue.getDisplayName( );
```

actuate.parameter.ParameterValue.getGroup

Syntax string ParameterValue.getGroup( )

Gets the group value for this ParameterValue.

Returns String.

Example To store the group that the parameter value pvalue belongs to in a variable called group, use code similar to the following:

```
var group = pvalue.getGroup( );
```

actuate.parameter.ParameterValue.getName

Syntax string ParameterValue.getName( )

Gets the name value for this ParameterValue.

Returns String.

Example To store the name of the parameter value pvalue in a variable called name, use code similar to the following:

```
var name = pvalue.getName( );
```

actuate.parameter.ParameterValue.getPosition

Syntax integer ParameterValue.getPosition( )

Gets the position value for this ParameterValue.

Returns Integer.

Example To save the position of the parameter value pvalue in the parameter list to a variable called pos, use code similar to the following:

```
var pos = pvalue.getPosition( );
```

actuate.parameter.ParameterValue .getPromptParameter

Syntax boolean ParameterValue.getPromptParameter( )

Gets the promptParameter value for this ParameterValue.

Returns Boolean.

Example To store the prompt parameter of the parameter value pvalue in a variable called prompt, use code similar to the following:

```
var prompt = pvalue.getPromptParameter( );
```

actuate.parameter.ParameterValue.getValue

Syntax string[ ] ParameterValue.getValue( )

Gets the value values for this ParameterValue.

Returns String or array of strings. The value or values of this ParameterValue object.

Example To store the value of the parameter value pvalue in a variable called value, use code similar to the following:

```
var value = pvalue.getValue( );
```

actuate.parameter.ParameterValue.getValueIsNull

Syntax boolean ParameterValue.getValueIsNull( )

Gets the valueIsNull value for this ParameterValue.

Returns Boolean. True indicates that this ParameterValue is null.

Example To alert the user that the value of the parameter value pvalue is null, use code similar to the following:

```
if (pavalue.getValueIsNull( )){
   alert('Default value is null!');
}
```

actuate.parameter.ParameterValue.isViewParameter

Syntax boolean ParameterValue.isViewParameter( )

Gets the isViewParameter value for this ParameterValue.

Returns Boolean. True indicates that this ParameterValue is visible.

Example To set specific help text for the parameter value pvalue if it is a view-time parameter, use code similar to the following:

```
if (pvalue.isViewParameter( )){
   pvalue.setHelpText("This is a view-time parameter.");
}
```

actuate.parameter.ParameterValue.setColumnName

Syntax void ParameterValue.setColumnName(string columnName)

Sets the column name value for this ParameterValue.

Parameters **columnName**
String. The name of the column.

Example To set the column name for the parameter value pvalue to Year, use code similar to the following:

```
pvalue.setColumnName("Year");
```

actuate.parameter.ParameterValue.setColumnType

Syntax void ParameterValue.setColumnType(string columnType)

Sets the data type of the column for this ParameterValue. Used for queries.

Parameters **columnType**
String. Possible values are "Currency", "Date", "DateOnly", "Time", "Double", "Integer", "String", "Boolean", "Structure", "Table", and "Unknown".

Example To set the column type for the parameter value pvalue to Date, use code similar to the following:

```
pvalue.setColumnType("Date");
```

actuate.parameter.ParameterValue.setDataType

Syntax void ParameterValue.setDataType(string dataType)

Sets the dataType value for this ParameterValue.

Parameters **dataType**
String. Possible values are "Currency", "Date", "DateOnly", "Time", "Double", "Integer", "String", "Boolean", "Structure", "Table", and "Unknown".

Example To set the data type for the parameter value pvalue to Date, use code similar to the following:

```
pvalue.setDataType("Date");
```

actuate.parameter.ParameterValue.setDisplayName

Syntax void ParameterValue.setDisplayName(string name)

Sets the displayed name value for this ParameterValue.

Parameters **name**
String. A displayed parameter name.

Example To set the display name of the parameter value pvalue to Year, use code similar to the following:

```
pvalue.setDisplayName("Year");
```

actuate.parameter.ParameterValue.setGroup

Syntax void ParameterValue.setGroup(string group)

Sets the group value for this ParameterValue.

Parameters **group**
String. The name of the group.

Example To set the group for the parameter value pvalue to Customer Details, use code similar to the following:

```
pvalue.setGroup("Customer Details");
```

actuate.parameter.ParameterValue .setIsViewParameter

Syntax void ParameterValue.setIsViewParameter(boolean isViewParameter)

Sets the isViewParameter value for this ParameterValue.

Parameters **isViewParameter**
Boolean. True indicates a view-time parameter.

Example To make the parameter value pvalue into a view-time parameter, use code similar to the following:

```
pvalue.setIsViewParameter(true);
```

actuate.parameter.ParameterValue.setName

Syntax void ParameterValue.setName(string name)

Sets the name value for this ParameterValue.

Parameters **name**
String. A parameter name.

Example To set the name of the parameter value pvalue to Year, use code similar to the following:

```
pvalue.setName("Year");
```

actuate.parameter.ParameterValue.setPosition

Syntax void ParameterValue.setPosition(integer position)

Sets the position value for this ParameterValue.

Parameters **position**
Integer. The position from the top of the parameter list.

Example To move the parameter value pvalue one place farther down in the parameter list, use code similar to the following:

```
pvalue.setPosition(++pvalue.getPosition( ));
```

actuate.parameter.ParameterValue .setPromptParameter

Syntax void ParameterValue.setPromptParameter(boolean promptParameter)

Sets the promptParameter value for this ParameterValue.

Parameters **promptParameter**
Boolean. True indicates that this parameter prompts the user.

Example To set the parameter value pvalue to not prompt the user, use code similar to the following:

```
pvalue.setPromptParameter(false);
```

actuate.parameter.ParameterValue.setValue

Syntax void ParameterValue.setValue(string[ ] value)

Sets the value or values for this ParameterValue.

Parameters **value**
String or array of strings. The value or values of this ParameterValue object.

Example To set the value of the parameter value pvalue to 2010, use code similar to the following:

```
pvalue.setValue("2010");
```

To set the values of the ParameterValue object pvalues to 2008, 2009, and 2010, use code similar to the following:

```
pvalue.setValue({"2008", "2009", "2010"});
```

actuate.parameter.ParameterValue.setValueIsNull

Syntax void ParameterValue.setValueIsNull(boolean valueIsNull)

Sets the valueIsNull value for this ParameterValue.

Parameters **valueIsNull**
Boolean. True indicates that this ParameterValue is null.

Example To set the value of parameter value pvalue to null, use code similar to the following:

```
pvalue.setValueIsNull(true);
```

Class actuate.report.Chart

Description Provides functions to operate on a chart element, such as changing its format or retrieving data from specific elements.

Constructor

The actuate.report.Chart object is created when actuate.viewer.PageContent.getChartByBookmark( ) is called.

Function summary

Table 4-26 lists actuate.report.Chart functions.

Table 4-26 actuate.report.Chart functions

Function	Description
clearFilters( )	Clears the filters applied to the given column
drillDownCategory( )	Drills down into a chart by category
drillDownSeries( )	Drills down into a chart by series
drillUpCategory( )	Drills up one level by category
drillUpSeries( )	Drills up one level by series
getBookmark( )	Returns the report element bookmark name
getClientChart( )	Returns an HTML5 instance of this chart
getHtmlDom( )	Returns the HTML element DOM object
getInstanceId( )	Returns the report element instance id
getPageContent( )	Returns the page content to which this element belongs
getType( )	Returns the report element type
hide( )	Hides this element
setChartTitle( )	Sets the title for this chart
setDimension( )	Sets the number of dimensions for the chart element
setFilters( )	Applies filters to this chart element
setSize( )	Sets the width and height of the chart element
setSubType( )	Sets a chart subtype to the chart element
show( )	Shows this element

(continues)

Table 4-26 actuate.report.Chart functions (continued)

Function	Description
submit( )	Submits all the asynchronous operations that the user has requested on this report and renders the chart component on the page

actuate.report.Chart.clearFilters

Syntax void Chart.clearFilters(string columnName)

Clears the filters for a given column.

Parameters **columnName**
String. The name of the column.

Example This example clears existing filters from the PRODUCTLINE column of a chart and changes the chart title:

```
function resetFilter(bchart){
   bchart.clearFilters("PRODUCTLINE");
   bchart.setChartTitle("Orders By Country");
   bchart.submit( );
}
```

actuate.report.Chart.drillDownCategory

Syntax void Chart.drillDownCategory(string categoryData)

Drills down into a chart by category.

Parameters **categoryData**
String. The name of the data category to drill down to.

actuate.report.Chart.drillDownSeries

Syntax void Chart.drillDownSeries(string seriesName)

Drills down into a chart by series.

Parameters **seriesName**
String. The name of the data series to drill down to.

actuate.report.Chart.drillUpCategory

Syntax void Chart.drillUpCategory( )

Drills up into a chart by one data category level.

actuate.report.Chart.drillUpSeries

Syntax void Chart.drillUpSeries( )

Drills up into a chart by one series level.

actuate.report.Chart.getBookmark

Syntax string Chart.getBookmark( )

Returns the chart's bookmark name.

Returns String. The chart's bookmark name.

Example This example sets the chart's title to the bookmark name:

```
function titleBookmark(bchart){
   bchart.setChartTitle(bchart.getBookmark( ));
   bchart.submit( );
}
```

actuate.report.Chart.getClientChart

Syntax actuate.report.HTML5Chart.ClientChart Chart.getClientChart( )

Returns the HTML5 Chart instance if this chart has an HTML5 Chart output format, otherwise returns null.

Returns actuate.report.HTML5Chart.ClientChart. The HTML5 formatted chart or null.

Example This example displays the chart ID of the HTML5 chart in an alert box:

```
function showHTML5ChartID(myChart){
   var myHTML5Chart = myChart.getClientChart( );
   var HTML5ChartID = myHTML5Chart.getViewerId( );
   alert (HTML5ChartID);
}
```

actuate.report.Chart.getHtmlDom

Syntax HTMLElement Chart.getHtmlDom( )

Returns the HTML element for this chart.

Returns HTMLElement. The HTML DOM element.

Example This example displays the HTML DOM element for this chart inside a red border:

```
function showHtmlDom(myChart){
   var domNode = myChart.getHtmlDom( );
   var box = document.createElement('div');
   box.style.border = '2px solid red';
   var label = document.createElement('h2');
   label.innerHTML = 'The HTML DOM:';
   box.appendChild(label);
   box.appendChild(domNode);
   document.body.appendChild(box);
}
```

actuate.report.Chart.getInstanceId

Syntax string Chart.getInstanceId( )

Returns the instance id of this report element.

Returns String. The instance id.

Example This example displays the instance ID of the report element in an alert box:

```
function showID(myChart){
   var elementID = myChart.getInstanceId( );
   alert (elementID);
}
```

actuate.report.Chart.getPageContent

Syntax actuate.viewer.PageContent Chart.getPageContent( )

Returns the content of the page to which this chart belongs.

Returns actuate.report.PageContent. The report content.

Example This example displays the viewer ID of the page content in an alert box:

```
function showViewID(myChart){
   var pageContent = myChart.getPageContent( );
   var pageViewerID = pageContent.getViewerId( );
   alert (pageViewerID);
}
```

actuate.viewer.Chart.getType

Syntax string Chart.getType( )

Returns the chart's report element type.

Returns String. This method returns the string "Chart" when the type is actuate.report.Chart.CHART and the string "Flash Chart" when the type is actuate.report.Chart.FLASH_CHART.

Example This example displays the chart type in an alert box:

```
alert ("Chart is of type " + myChart.getType( ));
```

actuate.report.Chart.hide

Syntax void Chart.hide( )

Hides this element.

Example To hide the chart bchart, use code similar to the following:

```
alert("Hiding chart" + bchart.getBookmark( ));
bchart.hide( );
bchart.submit( );
```

actuate.report.Chart.setChartTitle

Syntax void Chart.setChartTitle(string title)

Sets the title for this chart element.

Parameters **title**
String. The title for the chart.

Example This example sets the chart's title to the bookmark name:

```
function titleBookmark(bchart){
   bchart.setChartTitle(bchart.getBookmark( ));
   bchart.submit( );
}
```

actuate.report.Chart.setDimension

Syntax void Chart.setDimension(actuate.report.Chart dimension)

Sets the number of dimensions for the chart element. The chart dimension only works if supported by the chart's type. A 3D chart does not support multiple value axes. Remove all of the *y*-axes after the first before converting a chart to 3D.

Parameters **dimension**
actuate.report.Chart. The number of dimensions in which to display the chart element. Supported values are 2D and 2D with depth. The constants defined for this argument are:

- actuate.report.Chart.CHART_DIMENSION_2D
- actuate.report.Chart.CHART_DIMENSION_2D_WITH_DEPTH

Example This example changes the chart bchart's dimension to 2D with depth:

```
bchart.setChartTitle(bchart.getBookmark( ) + ": 2D with Depth");
bchart.setDimension(actuate.report.Chart.CHART_DIMENSION_2D_WITH_D
   EPTH );
bchart.submit( );
```

actuate.report.Chart.setFilters

Syntax void Chart.setFilters(actuate.data.Filter filter)

void Chart.setFilters(actuate.data.Filter[ ] filters)

Applies filters to this chart element. To apply more than one filter to a chart element, call this function multiple times, once for each filter object.

Parameters **filter**
An actuate.data.Filter object. A single filter condition to apply to this chart element.

filters
An array of actuate.data.Filter objects. Filter conditions to apply to this chart element.

Example This example applies a filter to the chart and changes the chart's title to reflect the filter:

```
function chartFilter(bchart){
   var filter = new actuate.data.Filter("PRODUCTLINE", "=",
                                          "Trucks and Buses");
   var filters = new Array( );
   filters.push(filter);

   bchart.setFilters(filters);
   bchart.setChartTitle("Orders By Country (Trucks and Buses)");
   bchart.submit( );
}
```

actuate.report.Chart.setSize

Syntax void Chart.setSize(integer width, integer height)

Sets the width and height of the chart element displayed.

Parameters **width**
Integer. The width in pixels.

height
Integer. The height in pixels.

Example To set the chart bchart to be 600 pixels wide by 800 pixels high, use code similar to the following:

```
alert("Resizing " + bchart.getBookmark( ) + " to 600x800");
bchart.setSize(600,800);
bchart.submit( );
```

actuate.report.Chart.setSubType

Syntax void Chart.setSubType(string chartType)

Sets a subtype for this chart element. When the report calls submit( ), the report redraws the chart element as the requested type.

Parameters **chartType**
String. The format in which to redraw the chart element. The constants that define the chart subtypes are:

- CHART_SUBTYPE_PERCENTSTACKED
- CHART_SUBTYPE_SIDEBYSIDE
- CHART_SUBTYPE_STACKED

Example To change the subtype of the chart bchart to side-by-side, use code similar to the following:

```
bchart.setChartTitle("Side by Side Chart");
bchart.setSubType(actuate.report.Chart.CHART_SUBTYPE_SIDEBYSIDE);
bchart.submit( );
```

actuate.report.Chart.show

Syntax void Chart.show( )

Shows this element.

Example To reveal the hidden chart bchart, use code similar to the following:

```
alert("Showing chart" + bchart.getBookmark( ));
bchart.show( );
bchart.submit( );
```

actuate.report.Chart.submit

Syntax void Chart.submit(function callback)

Submits all the asynchronous operations for this chart. The submit( ) function triggers an AJAX request for all asynchronous operations. When the server finishes the processing, it returns a response and the results are rendered on the page in the chart container.

Parameters **callback**

Function. Optional. A function to execute after the asynchronous call processing is done. Submit passes the current actuate.Viewer object to the callback as an input parameter.

Example This example sets the chart's title to the bookmark name and pops up an alert box after calling submit( ):

```
function titleBookmark(bchart){
   bchart.setChartTitle(bchart.getBookmark( ));
   bchart.submit(alert("Title Changed"));
}
```

Class actuate.report.DataItem

Description A container for a data element in a report. DataItem provides functions to operate on a data element, such as retrieving the data value and getting the HTML DOM element from the report data element.

Constructor

The DataItem object is constructed by actuate.viewer.PageContent.getDataItemByBookmark( ).

Function summary

Table 4-27 lists actuate.report.DataItem functions.

Table 4-27 actuate.report.DataItem functions

Function	Description
getBookmark( )	Returns the bookmark name for this data item
getData( )	Returns the data value on this data element
getHtmlDom( )	Returns the HTML element for this data item
getPageContent( )	Returns the page content to which this element belongs
getType( )	Returns the report element type
hide( )	Hides this element
show( )	Shows this element
submit( )	Applies the changes made to this DataItem

actuate.report.DataItem.getBookmark

Syntax string DataItem.getBookmark( )

Returns the bookmark name for this data item.

Returns String.

Example This example displays the data item's bookmark in an alert box:

```
alert(myDataItem.getBookmark( ));
```

actuate.report.DataItem.getData

Syntax string DataItem.getData( )

Returns the data value of this data element.

Returns String. The data value.

Example This example displays the data element's data value in an alert box:

```
alert(myDataItem.getData( ));
```

actuate.report.DataItem.getHtmlDom

Syntax HTMLElement DataItem.getHtmlDom( )

Returns the HTML element for this data item.

Returns HTMLElement.

Example This example displays the HTML DOM element for this data item inside a red border:

```
function showHtmlDom(myDataItem){
   var domNode = myDataItem.getHtmlDom( );
   var box = document.createElement('div');
   box.style.border = '2px solid red';
   var label = document.createElement('h2');
   label.innerHTML = 'The HTML DOM:';
   box.appendChild(label);
   box.appendChild(domNode);
   document.body.appendChild(box);
}
```

actuate.report.DataItem.getInstanceId

Syntax string DataItem.getInstanceId( )

Returns the instance id of this report element.

Returns String. The instance id.

Example This example displays the instance ID of the report element in an alert box:

```
function showID(myDataItem){
   var elementID = myDataItem.getInstanceId( );
   alert (elementID);
}
```

actuate.report.DataItem.getPageContent

Syntax actuate.viewer.PageContent DataItem.getPageContent( )

Returns the page content to which this data item belongs.

Returns actuate.report.PageContent. report content.

Example This example displays the viewer ID of the page content in an alert box:

```
function showViewID(myDataItem){
   var pageContent = myDataItem.getPageContent( );
   var pageViewerID = pageContent.getViewerId( );
   alert (pageViewerID);
}
```

actuate.report.DataItem.getType

Syntax string DataItem.getType( )

Returns the report element type of this object, which is data.

Returns String. "Data".

Example This example checks the report element type and displays an alert if the type is not "Data":

```
if (myDataItem.getType( ) != "Data"){
   alert("Type mismatch, report element type is not data");
}
```

actuate.report.DataItem.hide

Syntax void DataItem.hide( )

Hides this element.

Example Use hide( ) to hide a data item object, as shown in the following code:

```
myDataItem.hide( );
```

actuate.report.DataItem.show

Syntax void DataItem.show( )

Shows this element.

Example Use show( ) to reveal a hidden data item object, as shown in the following code:

```
myDataItem.show( );
```

actuate.report.DataItem.submit

Syntax void DataItem.submit(function callback)

Submits all the asynchronous operations for this DataItem. Submit( ) triggers an AJAX request for all asynchronous operations. When the server finishes the processing, it returns a response and the results are rendered on the page in the DataItem container.

Parameters **callback**

Function. The function to execute after the asynchronous call processing is done.

Example Use submit( ) to execute changes on a data item object, as shown in the following code:

```
myDataItem.submit( );
```

Class actuate.report.FlashObject

Description A container for a Flash object in a report. FlashObject provides functions to operate on a Flash object, such as retrieving content and getting the HTML DOM element from the report Flash element.

Constructor

The FlashObject object is constructed by actuate.viewer.PageContent.getFlashObjectByBookmark( ).

Function summary

Table 4-28 lists actuate.report.FlashObject functions.

Table 4-28 actuate.report.FlashObject functions

Function	Description
clearFilters( )	Removes filters from this FlashObject
getBookmark( )	Returns the bookmark name for this FlashObject
getHtmlDom( )	Returns the HTML element for this FlashObject
getInstanceId( )	Returns the report element instance id
getPageContent( )	Returns the page content to which this element belongs
getType( )	Returns the FlashObject's element type
hide( )	Hides this element
setFilters( )	Adds filters to this FlashObject
show( )	Shows this element
submit( )	Applies changes made to this FlashObject

actuate.report.FlashObject.clearFilters

Syntax void FlashObject.clearFilters(string columnName)

Clears the filters of a given column.

Parameters **columnName**
String. The name of the column from which to clear the filters.

Example This example clears existing filters from the PRODUCTLINE column:

```
function resetFilter(flashobj){
   flashobj.clearFilters("PRODUCTLINE");
   flashobj.submit( );
}
```

actuate.report.FlashObject.getBookmark

Syntax string FlashObject.getBookmark( )

Returns the bookmark of this FlashObject element.

Returns String.

Example This example displays the Flash object's bookmark in an alert box:

```
function alertBookmark(myFlashobj){
   alert(myFlashobj.getBookmark( ));
}
```

actuate.report.FlashObject.getHtmlDom

Syntax HTMLElement FlashObject.getHtmlDom( )

Returns the HTML element for this FlashObject.

Returns HTMLElement.

Example This example displays the HTML DOM element for this data item inside a red border:

```
function showHtmlDom(myFlashobj){
   var domNode = myFlashobj.getHtmlDom( );
   var box = document.createElement('div');
   box.style.border = '2px solid red';
   var label = document.createElement('h2');
   label.innerHTML = 'The HTML DOM:';
   box.appendChild(label);
   box.appendChild(domNode);
   document.body.appendChild(box);
}
```

actuate.report.FlashObject.getInstanceId

Syntax string FlashObject.getInstanceId( )

Returns the instance id of this report element.

Returns String. The instance id.

Example This example displays the instance ID of the report element in an alert box:

```
function showID(myFlashObject){
   var elementID = myFlashObject.getInstanceId( );
   alert (elementID);
}
```

actuate.report.FlashObject.getPageContent

Syntax actuate.viewer.PageContent FlashObject.getPageContent( )

Returns the page content to which this FlashObject belongs.

Returns actuate.viewer.PageContent. report content.

Example This example displays the viewer ID of the page content in an alert box:

```
function showViewID(myFlashobj){
   var pageContent = myFlashobj.getPageContent( );
   var pageViewerID = pageContent.getViewerId( );
   alert (pageViewerID);
}
```

actuate.report.FlashObject.getType

Syntax string FlashObject.getType( )

Returns the report element type of this object, which is FlashObject.

Returns String. "FlashObject".

Example This example checks the report element type and displays an alert if the type is not "FlashObject":

```
if (myFlashObject.getType( ) != "FlashObject"){
   alert("Type mismatch, report element type is not FlashObject");
}
```

actuate.report.FlashObject.hide

Syntax void FlashObject.hide( )

Hides this element.

Example Use hide( ) to hide the Flash object, as shown in the following code:

```
myFlashobj.hide( );
```

actuate.report.FlashObject.setFilters

Syntax void FlashObject.setFilters(actuate.data.Filter[ ] filters)

Sets the given filters.

Parameters **filters**
An array of actuate.data.Filter objects. The filter conditions to apply to this chart element.

Example This example applies a filter to the Flash object:

```
function newFilter(myFlashobj){
   var filter = new
      actuate.data.Filter("PRODUCTLINE", "=", "Trucks and Buses");
   var filters = new Array( );
   filters.push(filter);
   myFlashobj.setFilters(filters);
}
```

actuate.report.FlashObject.show

Syntax void FlashObject.show( )

Shows this element.

Example Use show( ) to reveal a hidden Flash object, as shown in the following code:

```
myFlashobj.show( );
```

actuate.report.FlashObject.submit

Syntax void FlashObject.submit(function callback)

Submits all the asynchronous operations for this FlashObject. Submit( ) triggers an AJAX request for all asynchronous operations. When the server finishes the processing, it returns a response and the results are rendered on the page in the FlashObject container.

Parameters **callback**
Function. The function to execute after the asynchronous call processing is done.

Example This example clears existing filters from the PRODUCTLINE column and pops up an alert box:

```
function alertResetFilter(flashobj){
   flashobj.clearFilters("PRODUCTLINE");
   flashobj.submit(alert("Filters Cleared"));
}
```

Class actuate.report.Gadget

Description A container for a Flash gadget object in a report. The Gadget class provides functions to operate on a Flash gadget object, such as retrieving content and getting the HTML DOM element from the report Flash element.

Constructor

The Gadget object is constructed by viewer.PageContent.getGadgetByBookmark( ).

Function summary

Table 4-29 lists actuate.report.Gadget functions.

Table 4-29 actuate.report.Gadget functions

Function	Description
clearFilters( )	Removes filters from this gadget
getBookmark( )	Returns the bookmark name for this gadget
getHtmlDom( )	Returns the HTML element for this gadget
getInstanceId( )	Returns the report element instance id
getPageContent( )	Returns the page content to which this element belongs
getType( )	Returns the gadget's element type, which is gadget
hide( )	Hides this element
setFilters( )	Adds filters to this gadget
setGadgetType( )	Sets the gadget type
setSize( )	Resizes the gadget's width and height
show( )	Shows this element
submit( )	Applies changes made to this gadget

actuate.report.Gadget.clearFilters

Syntax void Gadget.clearFilters(string columnName)

Clears the filters of a given column.

Parameters **columnName**
String. The name of the column from which to clear the filters.

Example This example clears existing filters from the PRODUCTLINE column:

```
function resetFilter(myGadget){
   myGadget.clearFilters("PRODUCTLINE");
   myGadget.submit( );
}
```

actuate.report.Gadget.getBookmark

Syntax string Gadget.getBookmark( )

Returns the bookmark of this Gadget element.

Returns String. The gadget's bookmark.

Example This example displays the gadget's bookmark in an alert box:

```
function alertBookmark(myGadget){
   alert(myGadget.getBookmark( ));
}
```

actuate.report.Gadget.getHtmlDom

Syntax HTMLElement Gadget.getHtmlDom( )

Returns the HTML element for this gadget.

Returns HTMLElement.

Example This example displays the HTML DOM element for this gadget inside a red border:

```
function showHtmlDom(myGadget){
   var domNode = myGadget.getHtmlDom( );
   var box = document.createElement('div');
   box.style.border = '2px solid red';
   var label = document.createElement('h2');
   label.innerHTML = 'The HTML DOM:';
   box.appendChild(label);
   box.appendChild(domNode);
   document.body.appendChild(box);
}
```

actuate.report.Gadget.getInstanceId

Syntax string Gadget.getInstanceId( )

Returns the instance id of this report element.

Returns String. The instance id.

Example This example displays the instance ID of the report element in an alert box:

```
function showID(myGadget){
   var elementID = myGadget.getInstanceId( );
   alert (elementID);
}
```

actuate.report.Gadget.getPageContent

Syntax actuate.viewer.PageContent Gadget.getPageContent( )

Returns the page content to which this gadget belongs.

Returns actuate.viewer.PageContent. report content.

Example This example displays the viewer ID of the page content in an alert box:

```
function showViewID(myGadget){
   var pageContent = myGadget.getPageContent( );
   var pageViewerID = pageContent.getViewerId( );
   alert (pageViewerID);
}
```

actuate.report.Gadget.getType

Syntax string Gadget.getType( )

Returns the report element type of this object, which is Gadget.

Returns String. "Gadget".

Example This example checks the report element type and displays an alert if the type is not "Gadget":

```
if (myGadget.getType( ) != "Gadget"){
   alert("Type mismatch, report element type is not Gadget");
}
```

actuate.report.Gadget.hide

Syntax void Gadget.hide( )

Hides this element.

Example Use hide( ) to hide a gadget, as shown in the following code:

```
myGadget.show( );
```

actuate.report.Gadget.setFilters

Syntax void Gadget.setFilters(actuate.data.Filter[ ] filters)

Sets the given filters.

Parameters **filters**
An array of actuate.data.Filter objects. The filter conditions to apply to this chart element.

Example This example applies a filter to the gadget:

```
function newFilter(myGadget){
   var filter = new
      actuate.data.Filter("PRODUCTLINE", "=", "Trucks and Buses");
   var filters = new Array( );
   filters.push(filter);
   myGadget.setFilters(filters);
}
```

actuate.report.Gadget.setGadgetType

Syntax void Gadget.setGadgetType(string chartType)

Specifies the gadget type for the Gadget element. The chart type is a constant.

Parameters **chartType**
String. The possible values are constants as listed below:

- GADGET_TYPE_BULLET: Bullet gadget subtype
- GADGET_TYPE_CYLINDER: Cylinder gadget subtype
- GADGET_TYPE_LINEARGAUGE: LinearGauge gadget subtype
- GADGET_TYPE_METER: Meter gadget subtype
- GADGET_TYPE_SPARK: Spark gadget subtype
- GADGET_TYPE_THERMOMETER: Thermometer gadget subtype

Example To change the gadget type to a meter, use code similar to the following:

```
myGadget.setGadgetType(actuate.report.Gadget.GADGET_TYPE_METER);
```

actuate.report.Gadget.setSize

Syntax void Gadget.setSize(integer width, integer height)

Specifies the width and height of a gadget in pixels.

Parameters **width**
Integer. The width in pixels.

height
Integer. The height in pixels.

Example To set the gadget to a 300-by-300-pixel square area, use code similar to the following:

```
myGadget.setSize(300, 300);
```

actuate.report.Gadget.show

Syntax void Gadget.show( )

Shows this element.

Example Use show( ) to reveal a hidden gadget, as shown in the following code:

```
myGadget.show( );
```

actuate.report.Gadget.submit

Syntax void Gadget.submit(function callback)

Submits all the asynchronous operations for this gadget. Submit( ) triggers an AJAX request for all asynchronous operations. When the server finishes the processing, it returns a response and the results are rendered on the page in the gadget container.

Parameters **callback**
Function. The function to execute after the asynchronous call processing is done.

Example This example clears existing filters from the PRODUCTLINE column and pops up an alert box:

```
function alertResetFilter(myGadget){
   myGadget.clearFilters("PRODUCTLINE");
   myGadget.submit(alert("Filters Cleared"));
}
```

Class actuate.report.HTML5Chart.ClientChart

Description A container for an HTML5-enabled chart element in a report. ClientChart provides functions to operate on a ClientChart element on the client side only, such as retrieving the chart size or setting the title and series values for the currently displayed chart.

Constructor

The ClientChart object is constructed by actuate.report.Chart.getClientChart( ).

Function summary

Table 4-30 lists actuate.report.HTML5Chart.ClientChart functions.

Table 4-30 actuate.report.HTML5Chart.ClientChart functions

Function	Description
addSeries( )	Adds a series to the chart
getCategoryCount( )	Returns the number of categories in the chart
getChartHeight( )	Returns the height of the chart in pixels
getChartWidth( )	Returns the width of the chart in pixels
getClientOptions( )	Returns the chart options
getCore( )	Returns the core Highcharts object
getSeriesCount( )	Returns the number of run-time series in the chart
getXAxisMax( )	Returns the maximum value of X-axis series
getXAxisMin( )	Returns the minimum value of X-axis series
getYAxisMax( )	Returns the maximum value of Y-axis series
getYAxisMin( )	Returns the minimum value of Y-axis series
isChartWithAxes( )	Returns whether chart has axes
redraw( )	Redraws the chart according to chart options
removeSeries( )	Removes specified series
setSeriesVisible( )	Hides or displays specified series
setTitle( )	Updates chart title
setValues( )	Updates values of specified series
setXAxisRange( )	Changes the minimum and maximum of the X-axis and zooms in on the new data range

Table 4-30 actuate.report.HTML5Chart.ClientChart functions

Function	Description
setYAxisRange( )	Changes the minimum and maximum of the Y-axis and zooms in on the new data range

actuate.report.HTML5Chart.ClientChart.addSeries

Syntax void ClientChart.addSeries(string seriesName, Array values)

Adds a data series to this ClientChart.

Parameters **seriesName**
String. A name for the series.

values
Array. The values for the series, defining X and Y value pairs.

Example This example adds the monthly revenue series as an array of numbers:

```
myClientChart.addSeries('monthly revenue', [1,5.5, 2,4.5, 3,7.8,
   4,7.7, 5,1.2, 6,8.5 7,1.9, 8,4.5, 9,12, 10,9.1, 11,4, 12,6.6]);
```

actuate.report.HTML5Chart.ClientChart .getCategoryCount

Syntax integer ClientChart.getCategoryCount( )

Returns the number of categories in this ClientChart.

Returns Integer. The number of categories.

Example This example displays the number of categories in myClientChart as an alert:

```
alert("This HTML5 client chart has" +
   myClientChart.getCategoryCount( ) + "categories.");
```

actuate.report.HTML5Chart.ClientChart .getChartHeight

Syntax integer ClientChart.getChartHeight( )

Returns the height of this ClientChart in pixels.

Returns Integer. The height of the chart in pixels.

Example This example displays the height of myClientChart as an alert:

```
alert("Height: " + myClientChart.getHeight( ));
```

actuate.report.HTML5Chart.ClientChart .getChartWidth

Syntax integer ClientChart.getChartWidth( )

Returns the width of this ClientChart in pixels.

Returns Integer. The width of the chart in pixels.

Example This example displays the width of myClientChart as an alert:

```
alert("Width: " + myClientChart.getChartWidth( ));
```

actuate.report.HTML5Chart.ClientChart .getClientOptions

Syntax actuate.report.HTML5Chart.ClientOption ClientChart.getClientOptions( )

Returns the ClientOptions set for this ClientChart.

Returns actuate.report.HTML5Chart.ClientOption object. The client options.

Example This example retrieves the client options for myClientChart and stores them in the myClientOptions variable:

```
var myClientOptions = myClientChart.getClientOptions( );
```

actuate.report.HTML5Chart.ClientChart.getCore

Syntax actuate.report.HTML5Chart.Highcharts ClientChart.getCore( )

Returns the Highcharts object contained in this ClientChart.

Returns actuate.report.HTML5Chart.Highcharts object. A highchart.

Example This example retrieves the Highcharts object from myClientChart and stores it in the myHighchart variable:

```
var myHighchart = myClientChart.getCore( );
```

actuate.report.HTML5Chart.ClientChart .getSeriesCount

Syntax integer ClientChart.getSeriesCount( )

Returns the number of run-time series in this ClientChart.

Returns Integer. The number of series.

Example This example displays the number of run-time series in myClientChart as an alert:

```
alert("Runtime Series: " + myClientChart.getSeriesCount( ));
```

actuate.report.HTML5Chart.ClientChart.getXAxisMax

Syntax float ClientChart.getXAxisMax( )

Returns the maximum value of the series associated with the X-axis in this ClientChart.

Returns Float. The axis series' maximum.

Example This example displays the maximum value of the series associated with the X-axis in myClientChart as an alert:

```
alert("Max for X-axis series: " + myClientChart.getXAxisMax( ));
```

actuate.report.HTML5Chart.ClientChart.getXAxisMin

Syntax float ClientChart.getXAxisMin( )

Returns the minimum value of the series associated with the X-axis in this ClientChart.

Returns Float. The axis series' minimum.

Example This example displays the minimum value of the series associated with the X-axis in myClientChart as an alert:

```
alert("Min for X-axis series: " + myClientChart.getXAxisMin( ));
```

actuate.report.HTML5Chart.ClientChart.getYAxisMax

Syntax float ClientChart.getYAxisMax(integer axisIndex)

Returns the maximum value of a series associated with the Y-axis in this ClientChart.

Parameters **axisIndex**
Integer. Optional. Axis index. The minimum value is 0, which is the default value, indicating the first Axis.

Returns Float. The axis series' maximum.

Example This example displays the maximum value of the series associated with the Y-axis in myClientChart as an alert:

```
alert("Max for Y-axis series: " + myClientChart.getYAxisMax( ));
```

actuate.report.HTML5Chart.ClientChart.getYAxisMin

Syntax float ClientChart.getYAxisMin(integer axisIndex)

Returns the minimum value of a series associated with the Y-axis in this ClientChart.

Parameters **axisIndex**
Integer. Optional. Axis index. The minimum value is 0, which is the default value, indicating the first Axis.

Returns Float. The axis series' minimum.

Example This example displays the minimum value of the series associated with the Y-axis in myClientChart as an alert:

```
alert("Min for Y-axis series: " + myClientChart.getYAxisMin( ));
```

actuate.report.HTML5Chart.ClientChart .isChartWithAxes

Syntax boolean ClientChart.isChartWithAxes( )

Returns whether this chart has axes.

Returns Boolean. True indicates axes, false otherwise.

Example This example displays whether myClientChart has axes:

```
alert("Chart has axes: " + myClientChart.isChartWithAxes( ));
```

actuate.report.HTML5Chart.ClientChart.redraw

Syntax void ClientChart.redraw(actuate.report.HTML5Chart.ClientOption chartOptions)

Redraws this ClientChart with options.

Parameters **chartOptions**
actuate.report.HTML5Chart.ClientOption object. Optional. The chart options.

Example This example redraws myClientChart with the default options:

```
myClientChart.redraw( );
```

actuate.report.HTML5Chart.ClientChart.removeSeries

Syntax void ClientChart.removeSeries(string seriesName, boolean redraw)

Removes a series by name.

Parameters **seriesName**
String. The name of the series to remove.

redraw
Boolean. Optional. Specifies whether to redraw the chart. Default is true.

Example This example removes the series monthly revenue from myClientChart and redraws the chart:

```
myClientChart.removeSeries('monthly revenue', true);
```

actuate.report.HTML5Chart.ClientChart .setSeriesVisible

Syntax void ClientChart.setSeriesVisible(string seriesName, boolean visible)

Makes a series visible.

Parameters **seriesName**
String. The name of the series to change.

visible
Boolean. Optional. True indicates visible. Default is true.

Example This example sets the series monthly revenue as visible for myClientChart:

```
myClientChart.setSeriesVisible('monthly revenue', true);
```

actuate.report.HTML5Chart.ClientChart.setTitle

Syntax void ClientChart.setTitle(string title)

Sets the title of this ClientChart.

Parameters **title**
String. Chart title text.

Example This example sets the title of myClientChart to 'Annual Report':

```
myClientChart.setTitle('Annual Report');
```

actuate.report.HTML5Chart.ClientChart.setValues

Syntax void ClientChart.setValues(string seriesName, float[ ] values, boolean redraw)

Sets the value for a series.

Parameters **seriesName**
String. Name of the series to change.

values
Array of float. The values for the series, defining X and Y value pairs.

redraw
Boolean. Optional. Specifies whether to redraw the chart. Default is true.

Example This example adds the monthly revenue series as an array of numbers:

```
myClientChart.setValues('monthly revenue', [1,5.5, 2,4.5, 3,7.8,
   4,7.7, 5,1.2, 6,8.5 7,1.9, 8,4.5, 9,12, 10,9.1, 11,4, 12,6.6]);
```

actuate.report.HTML5Chart.ClientChart .setXAxisRange

Syntax void ClientChart.setXAxisRange(float min, float max, boolean redraw)

Sets the value range for the X-axis.

Parameters

min
Float. A new minimum value.

max
Float. A new maximum value.

redraw
Boolean. Optional. Specifies whether to redraw the chart. Default is true.

Example This example sets the X-axis range to 1 through 3 and redraws the chart:

```
myClientChart.setXAxisRange(1,3);
```

actuate.report.HTML5Chart.ClientChart .setYAxisRange

Syntax void ClientChart.setYAxisRange(float min, float max, boolean redraw, integer axisIndex)

Sets the value range for the Y-axis.

Parameters

min
Float. A new minimum value.

max
Float. A new maximum value.

redraw
Boolean. Optional. Specifies whether to redraw the chart. Default is true.

axisIndex
Integer. Optional. Axis index. The minimum value is 0, which is the default value, indicating the first Axis.

Example This example sets the Y-axis range to 0 through 15 and redraws the chart:

```
myClientChart.setYAxisRange(0,15);
```

Class actuate.report.HTML5Chart.ClientOption

Description A container for a ClientOption element in a report. ClientOption provides functions to change ClientChart features, such as orientation, type, and title.

Constructor

Syntax void actuate.report.HTML5Chart.ClientOption( )

Generates a new ClientOption object to manage the chart options for a ClientChart.

Function summary

Table 4-31 lists actuate.report.HTML5Chart.ClientOption functions.

Table 4-31 actuate.report.HTML5Chart.ClientOption functions

Function	Description
addSeries( )	Adds a series to the chart
explodePieSlice( )	Explodes specified pie's slice
isChartWithAxes( )	Checks if current chart is chart with axes
pivotChart( )	Inverts chart
setChartType( )	Updates chart type
setSeriesVisible( )	Hides or shows specified series
setTitle( )	Updates chart title
setXAxisTitle( )	Updates X-axis title
setYAxisTitle( )	Updates Y-axis title

actuate.report.HTML5Chart.ClientOption.addSeries

Syntax void ClientOption.addSeries(string seriesName, float[ ] values)

Adds a data series to this ClientOption.

Parameters **seriesName**
String. A name for the series.

values
Array of float. The values for the series, defining X and Y value pairs.

Example This example adds the monthly revenue series as an array of numbers:

```
myClientOption.addSeries('monthly revenue', [1,5.5, 2,4.5, 3,7.8,
   4,7.7, 5,1.2, 6,8.5 7,1.9, 8,4.5, 9,12, 10,9.1, 11,4, 12,6.6]);
```

actuate.report.HTML5Chart.ClientOption .explodePieSlice

Syntax void ClientOption.explodePieSlice(string categoryName, boolean sliced)

Explodes the specified pie chart's slice.

Parameters **categoryName**
String. The name of a category.

sliced
Boolean. Optional. True means the chart is sliced. Default is true.

Example This example explodes the Q1 category from a chart with myClientOption:

```
myClientOption.explodePieSlice('Q1');
```

actuate.report.HTML5Chart.ClientOption .isChartWithAxes

Syntax boolean ClientChart.isChartWithAxes( )

Returns whether this chart has axes.

Returns Boolean.

Example This example displays whether myClientOption has axes:

```
alert("Options has axes: " + myClientOption.isChartWithAxes( ));
```

actuate.report.HTML5Chart.ClientOption.pivotChart

Syntax void ClientChart.pivotChart( )

Switches the axes of the chart, if the chart has axes.

Example This example switches the axes in myClientOption and then redraws myClientChart with the switched axes:

```
var myClientOption = myClientChart.getClientOption( )
myClientOption.pivotChart( );
myClientChart.redraw(myClientOption);
```

actuate.report.HTML5Chart.ClientOption .setChartType

Syntax void ClientOption.setChartType(string chartType, boolean isCurve)

Sets the chart type in this ClientOption.

Parameters **chartType**
String. The chart type. Valid values are line, area, bar, scatter, and pie.

isCurve
Boolean. Optional. Indicates if line or area chart is curve. Default value is false.

Example This example changes the chart type to pie in myClientOption:

```
myClientOption.setChartType('pie');
```

actuate.report.HTML5Chart.ClientOption .setSeriesVisible

Syntax void ClientOption.setSeriesVisible(string seriesName, boolean visible)

Makes a series visible.

Parameters **seriesName**
String. The name of the series to change.

visible
Boolean. Optional. Default is true.

Example This example sets the series months as visible for myClientOption:

```
myClientOption.setSeriesVisible('monthly revenue', true);
```

actuate.report.HTML5Chart.ClientOption.setTitle

Syntax void ClientOption.setTitle(string title)

Sets the title of this ClientOption.

Parameters **title**
String. Chart title text.

Example This example sets the title of myClientOption to 'Annual Report':

```
myClientOption.setTitle('Annual Report');
```

actuate.report.HTML5Chart.ClientOption .setXAxisTitle

Syntax void ClientOption.setTitle(string title)

Sets the X-axis title of this ClientOption.

Parameters **title**
String. X-axis title text.

Example This example sets the title of the X-axis in myClientOption to 'Month':

```
myClientOption.setXAxisTitle('Month');
```

actuate.report.HTML5Chart.ClientOption .setYAxisTitle

Syntax void ClientOption.setTitle(string title)

Sets the Y-axis title of this ClientOption.

Parameters **title**

String. Y-axis title text.

Parameters **chartOptions**
Integer. Optional. Axis index. The minimum value is 0, which is the default value, indicating the first Axis.

Example This example sets the title of the Y-axis in myClientOption to 'Dollars, in millions':

```
myClientOption.setYAxisTitle('Dollars, in millions');
```

Class actuate.report.HTML5Chart.ClientPoint

Description Represents a data point in a chart. ClientPoint provides functions to manage a point in a series on an individual basis, including selections, options, and events. The options for ClientPoint are defined in the Highcharts point class, which is documented at the following URL:

```
http://www.actuate.com/documentation/R11SP4/actuatebirt
   /highcharts/Highcharts-Options-Reference.htm
```

Constructor

Syntax void actuate.report.HTML5Chart.ClientPoint( )

Generates a new ClientPoint object to manage a data point for a ClientChart.

Function summary

Table 4-32 lists actuate.report.HTML5Chart.ClientPoint functions.

Table 4-32 actuate.report.HTML5Chart.ClientPoint functions

Function	Description
applyOptions( )	Changes the point values or options
destroy( )	Destroys a point to clear memory
remove( )	Removes a point
select( )	Toggles the selection of a point
update( )	Updates the point with new options

actuate.report.HTML5Chart.ClientPoint.applyOptions

Syntax void ClientPoint.applyOptions({float | object} options)

Applies the options containing the x and y data and possibly some extra properties. This is called on point initialization or from point.update.

Parameters **options**
Float, array of float, or object. The point options. If options is a single number, the point gets that number as the Y value. If options is an array, the point gets the first two numbers as an X and Y value pair. If options is an object, advanced options as outlined in the Highcharts options.point are applied. The fields include color, events, id, marker, legend, Index (pie chart only), name, sliced (pie chart only), x, and y.

Example This example changes the Y value of myClientPoint to 12:

```
myClientPoint.applyOptions(12);
```

actuate.report.HTML5Chart.ClientPoint.destroy

Syntax void ClientPoint.destroy( )

Destroys a point to clear memory. Its reference still stays in series.data.

Example This example destroys the options and values for myClientPoint:

```
myClientPoint.destroy( );
```

actuate.report.HTML5Chart.ClientPoint.remove

Syntax void ClientPoint.remove(boolean redraw, {boolean | object} animation)

Removes this point and optionally redraws the series and axes.

Parameters **redraw**
Boolean. Optional. Specifies whether to redraw the chart. Default is true.

animation
Boolean or object. Optional. Whether to apply animation, and optionally animation configuration. Default is true.

Example This example removes myClientPoint from a series, and redraws the chart with animation to display the changed series:

```
myClientPoint.remove( );
```

actuate.report.HTML5Chart.ClientPoint.select

Syntax void ClientPoint.select(boolean selected, boolean accumulate)

Selects this point.

Parameters **selected**
Boolean. Specifies whether to select or deselect the point.

accumulate
Boolean. Whether to add this point to the previous selection. By default, this is true when the Ctrl (PC) or Cmd (Macintosh) key is held during selection.

Example This example selects MyClientPoint and deselects all other points:

```
myClientPoint.select(true, false);
```

actuate.report.HTML5Chart.ClientPoint.remove

Syntax void ClientPoint.remove(boolean redraw, {boolean | object} animation)

Updates this point and optionally redraws the series and axes.

Parameters **redraw**
Boolean. Optional. Specifies whether to redraw the chart. Default is true.

animation
Boolean or object. Optional. Whether to apply animation, and optionally animation configuration. Default is true.

Example This example removes myClientPoint and redraws the chart:

```
myClientPoint.remove();
```

actuate.report.HTML5Chart.ClientPoint.update

Syntax void ClientPoint.update({float|float[ ]|object} options, boolean redraw, {boolean | object} animation)

Updates this point and optionally redraws the series and axes.

Parameters **options**
Float, array of float, or object. The point options. If options is a single number, the point gets that number as the Y value. If options is an array, the point gets the first two numbers as an X and Y value pair. If options is an object, advanced options as outlined in the Highcharts options.point are applied. The fields include color, events, id, marker, legend, Index (pie chart only), name, sliced (pie chart only), x, and y.

redraw
Boolean. Optional. Specifies whether to redraw the chart. Default is true.

animation
Boolean or object. Optional. Whether to apply animation, and optionally animation configuration. Default is true.

Example This example updates myClientPoint with an X value of 1 and a Y value of 12, then redraws the point:

```
myClientPoint.update([1,12]);
```

Class actuate.report.HTML5Chart.ClientSeries

Description A container for a ClientSeries in a ClientChart. ClientSeries provides functions to manage a series and the graph of that series. In the ClientSeries object, all the points are accessible from the ClientSeries.data array.

Constructor

Syntax void actuate.report.HTML5Chart.ClientSeries( )

Generates a new ClientSeries object to manage a series for a ClientChart.

Function summary

Table 4-33 lists actuate.report.HTML5Chart.ClientSeries functions.

Table 4-33 actuate.report.HTML5Chart.ClientSeries functions

Function	Description
addPoint( )	Adds a point to the series
cleanData( )	Sorts the data and removes duplicates
destroy( )	Clears DOM objects and frees up memory
hide( )	Hides the series graph
redraw( )	Redraws the series after an update in the axes
remove( )	Removes a series and optionally redraws the chart
render( )	Renders the series graph and markers
select( )	Sets the selected state of the series graph
setData( )	Replaces the series data with a new set of data
setVisible( )	Sets the visibility of the series graph
show( )	Shows the series graph

actuate.report.HTML5Chart.ClientSeries.addPoint

Syntax void ClientSeries.addPoint({float | object} options, boolean redraw, boolean shift, {boolean | object} animation)

Adds a point dynamically to the series.

Parameters **options**
Object. The point options. If options is a single number, the point gets that number as the Y value. If options is an array, the point gets the first two numbers as an X and Y value pair. If options is an object, advanced options as outlined in

the Highcharts options.point are applied. The fields include color, events, id, marker, legend, Index (pie chart only), name, sliced (pie chart only), x, and y.

redraw
Boolean. Optional. Specifies whether to redraw the chart. Default is true.

shift
Boolean. When shift is true, the graph of the series shifts one point toward the end of the series and a point added to the beginning of the series. Default is false.

animation
Boolean or object. Optional. Whether to apply animation, and optionally animation configuration. Default is true.

Example This example adds a point of value 12 to the end of myClientSeries:

```
myClientSeriesaddPoint(12);
```

actuate.report.HTML5Chart.ClientSeries.cleanData

Syntax void ClientSeries.cleanData( )

Sorts the series and removes duplicate points or values.

Example This example sorts myClientSeries and removes its duplicate points and values:

```
myClientSeries.cleanData( );
```

actuate.report.HTML5Chart.ClientSeries.destroy

Syntax void ClientSeries.destroy( )

Clears DOM series objects and frees memory.

Example This example clears the memory of myClientSeries and its member objects:

```
myClientSeries.destroy( );
```

actuate.report.HTML5Chart.ClientSeries.hide

Syntax void ClientSeries.hide( )

Hides the graph of this series.

Example This example hides myClientSeries graph from the chart:

```
myClientSeries.hide( );
```

actuate.report.HTML5Chart.ClientSeries.redraw

Syntax void ClientSeries.redraw( )

Redraws the graph of this series after updating the data and axes.

Example This example redraws the graph of myClientSeries:

```
myClientSeries.redraw( );
```

actuate.report.HTML5Chart.ClientSeries.remove

Syntax void ClientSeries.remove(boolean redraw, {boolean | object} animation)

Removes this series and optionally redraws the chart.

Parameters **redraw**
Boolean. Optional. Specifies whether to redraw the chart. Default is true.

animation
Boolean or object. Optional. Whether to apply animation, and optionally animation configuration. Default is true.

Example This example removes the graph of myClientSeries from the chart:

```
myClientSeries.remove( );
```

actuate.report.HTML5Chart.ClientSeries.render

Syntax void ClientSeries.render( )

Renders the graph of this series and its markers.

Example This example renders the graph of myClientSeries to the chart:

```
myClientSeries.render( );
```

actuate.report.HTML5Chart.ClientSeries.select

Syntax void ClientSeries.select(boolean selected)

Selects this series.

Parameters **selected**
Boolean. Optional. Specifies whether to select or deselect the series. If undefined, toggles selection.

Example This example selects myClientSeries:

```
myClientSeries.select(true);
```

actuate.report.HTML5Chart.ClientSeries.setData

Syntax void ClientSeries.setData({float | object}[ ] data, boolean redraw)

Replaces the series data with a new set of data.

Parameters **data**
Array of float and/or object. An array of data points for the series. The points can be given in three ways:

1. A list of numerical values, which are assigned as Y values, paired with X values starting with 0 and incrementing by 1 for each additional number. For example:

   ```
   [0, 5, 3, 5]
   ```

2. A list of arrays with two values, which are assigned as X and Y value pairs. If the first value is a string, it is applied as the name of the point, and the x value is incremented following the above rules. For example:

   ```
   [[4, 2], [6, 3], [8, 2]]
   ```

3. A list of objects with named values, which are assigned to points using the Highchart point configuration specification options.point. For example:

   ```
   [{ name: 'Point 1',
      color: '#00FF00',
      y: 0
   },
   {  name: 'Point 2',
      color: '#FF00FF',
      y: 5
   }]
   ```

redraw
Boolean. Optional. Specifies whether to redraw the chart. Default is true.

Example This example replaces the points in myClientSeries with three new points:

```
myClientSeries.setData([[4, 2], [6, 3], [8, 2]]);
```

actuate.report.HTML5Chart.ClientSeries.setVisible

Syntax void ClientSeries.setVisible(boolean vis, boolean redraw)

Sets the visibility of this series.

Parameters **vis**
Boolean. Optional. Specifies whether to display the series. True displays the series, false hides it. If no value is provided, the visibility changes to false if visibility is true, and true if visibility is false.

redraw
Boolean. Optional. Specifies whether to redraw the chart. Default is true.

Example This example sets myClientSeries to visible and redraws it:

```
myClientSeries.setVisible(true);
```

actuate.report.HTML5Chart.ClientSeries.show

Syntax void ClientSeries.show( )

Displays the graph of this series.

Example This example displays the graph of myClientSeries:

```
myClientSeries.show( );
```

Class actuate.report.HTML5Chart.Highcharts

Description A container for a Highcharts element in a ClientChart. For reference material for highcharts, consult the BIRT Designer Professional help or access the highchart documentation at the following URL:

```
http://www.actuate.com/documentation/R11SP4/actuatebirt
   /highcharts/Highcharts-Options-Reference.htm
```

Constructor

Syntax void actuate.report.HTML5Chart.Highcharts( )

Generates a new Highcharts object to manage the Highchart for a ClientChart.

Class actuate.report.HTML5Chart.Renderer

Description A container for a Highchart renderer object. Directly accesses the Highchart's rendering layer to draw primitive shapes like circles, rectangles, paths or text directly. The renderer represents a wrapper object for SVG in modern browsers and VML in older versions of Microsoft Internet Explorer.

Constructor

Syntax void actuate.report.HTML5Chart.Renderer( )

Generates a new Renderer object to manage the Highchart rendering options for a ClientChart.

Function summary

Table 4-34 lists actuate.report.HTML5Chart.Renderer functions.

Table 4-34 actuate.report.HTML5Chart.Renderer functions

Function	Description
arc( )	Draws and returns an arc
circle( )	Draws a Scalable Vector Graphic circle
clipRect( )	Defines a clipping rectangle
destroy( )	Destroys the renderer and its allocated members
g( )	Creates a group
image( )	Displays an image
path( )	Draws a path
rect( )	Draws and returns a rectangle
setSize( )	Resizes the box and re-aligns all aligned elements
text( )	Adds text to the Scalable Vector Graphic object

actuate.report.HTML5Chart.Renderer.arc

Syntax object Renderer.arc(integer x, integer y, integer r, integer innerR, float start, float end)

Generates and draws an arc on the chart.

Parameters **x**

Integer. The X position of the arc's center, measured in pixels from the left edge of the rendering area.

y
Integer. The Y position of the arc's center, measured in pixels from the top edge of the rendering area.

r
Integer. The outer radius, measured in pixels.

innerR
Integer. The inner radius, measure in pixels.

start
Float. The starting angle of the arc, measured in radians, where 0 is directly right and -Math.PI/2 is directly upward. The arc is drawn clockwise from start to end.

end
Float. The ending angle of the arc, measured in radians, where 0 is directly right and -Math.PI/2 is directly upward.

Returns Highchart element object. The Element class is a JavaScript wrapper for SVG elements used in the rendering layer of Highchart. For reference material for highcharts, consult the BIRT Designer Professional help or access the highchart documentation at the following URL:

```
http://www.actuate.com/documentation/R11SP4/actuatebirt
   /highcharts/Highcharts-Options-Reference.htm#element
```

Example This example draws a 50-pixel wide half-circle arc, concave down, with a center 200 pixels from the left edge and 150 pixels from the top edge of the chart area:

```
myRenderer.arc(200, 150, 100, 50, -Math.PI, 0);
```

actuate.report.HTML5Chart.Renderer.circle

Syntax object Renderer.circle(integer x, integer y, integer r)

Generates and draws a Scalable Vector Graphic circle on the chart.

Parameters **x**
Integer. The X position of the circle's center, measured in pixels from the left edge of the rendering area.

y
Integer. The Y position of the circle's center, measured in pixels from the top edge of the rendering area.

r
Integer. The radius, measured in pixels.

Returns Highchart element object. The Element class is a JavaScript wrapper for SVG elements used in the rendering layer of Highchart. For reference material for highcharts, consult the BIRT Designer Professional help.

Example This example draws a circle with a center 200 pixels from the left edge and 150 pixels from the top edge of the chart area:

```
myRenderer.circle(200, 150, 100);
```

actuate.report.HTML5Chart.Renderer.clipRect

Syntax object Renderer.clipRect(string id, integer x, integer y, integer width, integer height)

Generates and draws a clipping rectangle on the chart.

Parameters **id**
String. A string to identify the element.

x
Integer. The X position of the rectangle's upper left corner, measured in pixels from the left edge of the rendering area.

y
Integer. The Y position of the rectangle's upper left corner, measured in pixels from the top edge of the rendering area.

width
Integer. The width, in pixels.

height
Integer. The height, in pixels.

Returns Highchart element object. The Element class is a JavaScript wrapper for SVG elements used in the rendering layer of Highchart. For reference material for highcharts, consult the BIRT Designer Professional help.

Example This example draws a 100-pixel-by-100-pixel rectangle 100 pixels from the left and top edges of chart area:

```
myRenderer.cliprect('myClipRect', 100, 100, 100, 100);
```

actuate.report.HTML5Chart.Renderer.destroy

Syntax void Renderer.destroy( )

Destroys this renderer and its allocated elements.

Example This example destroys the myRenderer object and frees its memory:

```
myRenderer.destroy( );
```

actuate.report.HTML5Chart.Renderer.g

Syntax object Renderer.g(string name)

Adds an SVG/VML group to the Renderer object.

Parameters **name**
String. The name of the group. Used in the class name, which will be "highcharts-"+ name. Other Element objects are added to the group by using this group as the first parameter in .add( ) for the element wrappers.

Returns Highchart element object. The Highchart.Element class is a JavaScript wrapper for SVG elements used in the rendering layer of Highchart. For reference material for highcharts, consult the BIRT Designer Professional help.

Example This example creates a new group called myGroup:

```
myRenderer.g('myGroup');
```

actuate.report.HTML5Chart.Renderer.image

Syntax object Renderer.image(string src, integer x, integer y, integer width, integer height)

Generates and draws a image on the chart.

Parameters **src**
String. A URL for the image.

x
Integer. The X position of the image's upper left corner, measured in pixels from the left edge of the rendering area.

y
Integer. The Y position of the image's upper left corner, measured in pixels from the top edge of the rendering area.

width
Integer. The width, in pixels.

height
integer. The height, in pixels.

Returns Highchart element object. The Highchart.Element class is a JavaScript wrapper for SVG elements used in the rendering layer of Highchart. For reference material for highcharts, consult the BIRT Designer Professional help.

Example This example adds the sun.png image to the chart 100 pixels from the left and top of the edge of the chart:

```
myRenderer.image('http://highcharts.com/demo/gfx/sun.png', 100,
   100, 30, 30);
```

actuate.report.HTML5Chart.Renderer.path

Syntax object Renderer.path(object[ ] path)

Adds a path to the renderer based on SVG's path commands. In SVG-capable browsers, all path commands are supported, but in VML only a subset is supported, including the moveTo, lineTo, and curve commands.

Parameters **path**
Array of string and integer objects. An SVG path with attributes split up in array form.

Returns Highchart element object. The Highchart.Element class is a JavaScript wrapper for SVG elements used in the rendering layer of Highchart. For reference material for highcharts, consult the BIRT Designer Professional help.

Example This example draws a path from the upper left corner of the rendering area (0, 0) to the points (100, 100), (200, 50), and (300, 100), where the first number represents the distance from the left edge of the rendering area and the second number represents the distance from the top edge of the rendering area:

```
myRenderer.path(['M', 0, 0, 'L', 100, 100, 200, 50, 300, 100]);
```

actuate.report.HTML5Chart.Renderer.rect

Syntax object Renderer.rect(integer x, integer y, integer width, integer height, integer r, integer strokeWidth)

Generates and draws a rectangle on the chart.

Parameters **x**
Integer. The X position of the rectangle's upper left corner, measured in pixels from the left edge of the rendering area.

y
Integer. The Y position of the rectangle's upper left corner, measured in pixels from the top edge of the rendering area.

width
Integer. The width, in pixels.

height
Integer. The height, in pixels.

r
Integer. The corner radius, measured in pixels.

strokeWidth
Integer. Stroke measurement to support crisp drawing.

Returns Highchart element object. The Highchart.Element class is a JavaScript wrapper for SVG elements used in the rendering layer of Highchart. For reference material for highcharts, consult the BIRT Designer Professional help.

Example This example draws a 100-pixel-by-100-pixel rectangle 100 pixels from the left and top edges of chart area with 5-pixel-radius quarter-circles as edges:

```
myRenderer.rect(100, 100, 100, 100, 5);
```

actuate.report.HTML5Chart.Renderer.setSize

Syntax void Renderer.setSize(integer width, integer height, boolean animate)

Resizes the rendering area and re-aligns all aligned elements.

Parameters

width
Integer. The width, in pixels.

height
Integer. The height, in pixels.

animate
Boolean. Optional. Whether to animated the resize. Default is true.

Example This example resizes the renderer area to 500 pixels by 500 pixels:

```
myRenderer.setSize(500, 500);
```

actuate.report.HTML5Chart.Renderer.text

Syntax object Renderer.text(string str, integer x, integer y, boolean useHTML)

Adds text to the Scalable Vector Graphic object.

Parameters

str
String. The text in this text element.

x
Integer. The X position of the text's lower left corner, measured in pixels from the left edge of the rendering area.

y
Integer. The Y position of the text's lower left corner, measured in pixels from the top edge of the rendering area.

useHTML
Boolean. Specifies whether to use HTML to render the text.

Returns Highchart element object. The Highchart.Element class is a JavaScript wrapper for SVG elements used in the rendering layer of Highchart. For reference material for highcharts, consult the BIRT Designer Professional help.

Example This example adds a text graphic that reads "Series 1" 140 pixels from the left edge of the rendering area and 150 pixels from the top edge of the rendering area:

```
myRenderer.text('Series 1', 140, 150, false);
```

Class actuate.report.Label

Description A container for a Label element in a report. Label provides functions to operate on a Label element, such as retrieving the label text and getting the HTML DOM element from the report label.

Constructor

The Label object is constructed by viewer.PageContent.getLabelByBookmark( ).

Function summary

Table 4-35 lists actuate.report.Label functions.

Table 4-35 actuate.report.Label functions

Function	Description
getBookmark( )	Returns the bookmark name for this Label
getHtmlDom( )	Returns the HTML element for this Label
getInstanceId( )	Returns the report element instance id
getLabel( )	Returns the text of this Label element
getPageContent( )	Returns the page content to which this element belongs
getType( )	Returns the Label's element type
hide( )	Hides this element
show( )	Shows this element
submit( )	Applies changes made to this gadget

actuate.report.Label.getBookmark

Syntax string Label.getBookmark( )

Returns the bookmark name for this Label.

Returns String. The Label's bookmark.

Example This example displays the Label's bookmark in an alert box:

```
alert(myLabel.getBookmark( ));
```

actuate.report.Label.getHtmlDom

Syntax HTMLElement Label.getHtmlDom( )

Returns the HTML element for this Label.

Returns HTMLElement.

Example This example displays the HTML DOM element for this Label inside a red border:

```
function showHtmlDom(myLabel){
   var domNode = myLabel.getHtmlDom( );
   var box = document.createElement('div');
   box.style.border = '2px solid red';
   var label = document.createElement('h2');
   label.innerHTML = 'The HTML DOM:';
   box.appendChild(label);
   box.appendChild(domNode);
   document.body.appendChild(box);
}
```

actuate.report.Label.getInstanceId

Syntax string Label.getInstanceId( )

Returns the instance id of this report element.

Returns String. The instance id.

Example This example displays the instance ID of the report element in an alert box:

```
function showID(myLabel){
   var elementID = myLabel.getInstanceId( );
   alert (elementID);
}
```

actuate.report.Label.getLabel

Syntax string Label.getLabel( )

Returns the text of this Label element.

Returns String. The Label text.

Example This example displays the text of the myLabel object in an alert box:

```
alert("Label element text is " + myLabel.getLabel( ));
```

actuate.report.Label.getPageContent

Syntax actuate.viewer.PageContent Label.getPageContent( )

Returns the page content to which this Label belongs.

Returns actuate.viewer.PageContent. report content.

Example This example displays the viewer ID of the page content in an alert box:

```
function showViewID(myLabel){
   var pageContent = myLabel.getPageContent( );
   var pageViewerID = pageContent.getViewerId( );
   alert (pageViewerID);
}
```

actuate.report.Label.getType

Syntax string Label.getType( )

Returns the report element type of this object, which is Label.

Returns String. "Label".

Example This example checks the report element type and displays an alert if the type is not "Label":

```
if (myElement.getType( ) != "Label"){
   alert("Type mismatch, report element type is not Label")
}
```

actuate.report.Label.hide

Syntax void Label.hide( )

Hides this element.

Example Use hide( ) to hide a report label, as shown in the following code:

```
myLabel.hide( );
```

actuate.report.Label.show

Syntax void Label.show( )

Shows this element.

Example Use show( ) to reveal a report label, as shown in the following code:

```
myLabel.show( );
```

actuate.report.Label.submit

Syntax void Label.submit(function callback)

Submits all the asynchronous operations for this Label. Submit( ) triggers an AJAX request for all asynchronous operations. When the server finishes the processing, it returns a response and the results are rendered on the page in the label container.

Parameters **callback**

Function. The function to execute after the asynchronous call processing is done.

Example Use submit( ) to execute changes on a Label object, as shown in the following code:

```
myLabel.submit( );
```

Class actuate.report.Table

Description A container for a Table element in a report. Table provides functions to operate on a Table element, such as manipulating columns, groups, and data.

Constructor

The Table object is constructed by viewer.PageContent.getTableByBookmark( ).

Function summary

Table 4-36 lists actuate.report.Table functions.

Table 4-36 actuate.report.Table functions

Function	Description
clearFilters( )	Clears the filters from the given column
getBookmark( )	Returns the bookmark name for this Table
getColumn( )	Gets the Table data by column index and returns only the data from the current visible page
getHtmlDom( )	Returns the HTML element for this Table
getInstanceId( )	Returns the report element instance id
getPageContent( )	Returns the page content to which this element belongs
getRow( )	Gets the Table data by row index
getType( )	Returns the report element type
groupBy( )	Adds an inner group to this Table
hide( )	Hides this element
hideColumn( )	Hides a Table column by specifying the column name
hideDetail( )	Hides detailed information for displayed groups
removeGroup( )	Removes an inner group
setFilters( )	Applies filters to this Table
setSorters( )	Adds sorters to this Table
show( )	Shows this element
showColumn( )	Shows a Table column by specifying the column name
showDetail( )	Shows detailed information for displayed groups

Table 4-36 actuate.report.Table functions

Function	Description
submit( )	Submits all the asynchronous operations that the user has requested on this report and renders the Table component on the page
swapColumns( )	Swaps two columns, reordering the columns

actuate.report.Table.clearFilters

Syntax void Table.clearFilters(string columnName)

Clears the filters of a given column.

Parameters **columnName**
String. The name of the column.

Example This example clears existing filters from the PRODUCTLINE column:

```
function resetFilter(myTable){
   myTable.clearFilters("PRODUCTLINE");
   myTable.submit( );
}
```

actuate.report.Table.getBookmark

Syntax string Table.getBookmark( )

Returns the Table's name.

Returns String. The name of the Table.

Example This example displays the Table's bookmark in an alert box:

```
function alertBookmark(myTable){
   alert(myTable.getBookmark( ));
}
```

actuate.report.Table.getColumn

Syntax array[ ] Table.getColumn(integer columnIndex)

Gets the Table data by column index. Returns the data from the current visible page.

Parameters **columnIndex**
Integer. Optional. The numerical index of the column from which to retrieve data. The getColumn( ) function returns the values for the first column when no value is provided for columnIndex.

Returns Array. A list of data in the format of the column.

Example This example returns the first column in myTable:

```
function getMyColumn(myTable) {
   return myTable.getColumn( );
}
```

actuate.report.Table.getHtmlDom

Syntax HTMLElement Table.getHtmlDom( )

Returns the Table's name.

Returns String. The name of the Table.

Example This example displays the HTML DOM element for this Table inside a red border:

```
function showHtmlDom(myTable){
   var domNode = myTable.getHtmlDom( );
   var box = document.createElement('div');
   box.style.border = '2px solid red';
   var label = document.createElement('h2');
   label.innerHTML = 'The HTML DOM:';
   box.appendChild(label);
   box.appendChild(domNode);
   document.body.appendChild(box);
}
```

actuate.report.Table.getInstanceId

Syntax string Table.getInstanceId( )

Returns the instance id of this report element.

Returns String. The instance id.

Example This example displays the instance ID of the report element in an alert box:

```
function showID(myTable){
   var elementID = myTable.getInstanceId( );
   alert (elementID);
}
```

actuate.report.Table.getPageContent

Syntax actuate.viewer.PageContent Table.getPageContent( )

Returns the page content to which this Table belongs.

Returns actuate.viewer.PageContent. report content.

Example This example displays the viewer ID of the page content in an alert box:

```
function showViewID(myTable){
   var pageContent = myTable.getPageContent( );
   var pageViewerID = pageContent.getViewerId( );
   alert (pageViewerID);
}
```

actuate.report.Table.getRow

Syntax array[ ] Table.getRow(integer rowIndex)

Gets the Table data by row index. Returns the data from the current visible page.

Parameters **rowIndex**
Integer. Optional. The numerical index of the row from which to retrieve data. The getRow( ) function returns the values for the first row when no value for rowIndex is provided.

Returns Array. A list of data in the format of the columns that cross the row.

Example This example retrieves the first row in myTable:

```
function getMyRow(myTable) {
   return myTable.getRow( );
}
```

actuate.report.Table.getType

Syntax string Table.getType( )

Returns the report element type of this object, which is Table.

Returns String. "Table".

Example This example returns the report element type of this object in an alert box:

```
function getTableType(myTable) {
   alert("Element type is: " + myTable.getType( ));
}
```

actuate.report.Table.groupBy

Syntax void Table.groupBy(string columnName)

Groups the data in a table by the values in a given column. If there is an existing group, this operation will add the new group after the existing group.

Parameters **columnName**
String. The name of the column to use for the innermost group to the Table.

Example This example groups the data in myTable by the values in the TOTAL column:

```
function groupByColumn(myTable) {
   myTable.groupBy("TOTAL");
}
```

actuate.report.Table.hide

Syntax void Table.hide( )

Hides this element.

Example This example hides myTable:

```
myTable.hide( );
```

actuate.report.Table.hideColumn

Syntax void Table.hideColumn(string columnName)

Hides a table column by specifying the column name.

Parameters **columnName**
String. The data binding name for the column to hide.

Example This example hides the TOTAL column from myTable:

```
function myHiddenColumn(myTable) {
   myTable.hideColumn("TOTAL");
   myTable.submit( );
}
```

actuate.report.Table.hideDetail

Syntax void Table.hideDetail(string columnName)

Hides information for a column from the grouped data displayed on the page. If every column is hidden, only the group name is visible.

Parameters **columnName**
String. The data binding name for the column to hide.

Example This example hides the TOTAL column from the grouped data visible for myTable:

```
function hideMyDetail(myTable) {
   myTable.hideDetail("TOTAL");
   myTable.submit( );
}
```

actuate.report.Table.removeGroup

Syntax void Table.removeGroup()

Removes the innermost group.

Example This example removes the innermost group from myTable and displays an alert box after calling submit( ):

```
function removeMyGroup(myTable) {
   myTable.removeGroup();
   myTable.submit(alert("Group removed"));
}
```

actuate.report.Table.setFilters

Syntax void Table.setFilters(actuate.data.Filter filter)

void Table.setFilters(actuate.data.Filter[ ] filters)

Applies a filter or filters to this Table element.

Parameters **filter**
actuate.data.Filter object. A single filter condition to apply to this Table.

filters
An array of actuate.data.Filter objects. Filter conditions to apply to this Table.

Example To add a filter to the Table to display only entries with a CITY value of NYC, use the following code:

```
var filters = new Array( );
var city_filter = new actuate.data.Filter("CITY",
   actuate.data.Filter.EQ, "NYC");
filters.push(city_filter);
table.setFilters(filters);
```

actuate.report.Table.setSorters

Syntax void Table.setSorters(actuate.data.Sorter sorter)

void Table.setSorters(actuate.data.Sorter[ ] sorters)

Applies a sorter or sorters to this Table.

Parameters **sorter**
actuate.data.Sorter object. A single sort condition to apply to this Table.

sorters
An array of actuate.data.Sorter objects. Sort conditions to apply to this Table.

Example This example adds the myStateSorter and myCitySorter sorters to myTable:

```
function setAllMySorters(myTable) {
   myTable.setSorters(["myStateSorter", "myCitySorter"]);
}
```

actuate.report.Table.show

Syntax void Table.show( )

Shows this element.

Example Use show( ) to reveal a report Table, as shown in the following code:

```
myTable.show( );
```

actuate.report.Table.showColumn

Syntax void Table.showColumn(string columnName)

Shows the Table column by specifying the column name.

Parameters **enabled**
String. The data binding name for the column to display.

Example This example shows the PRODUCTLINE column in myTable:

```
function showMyColumn(myTable) {
   myTable.showColumn("PRODUCTLINE");
   myTable.submit( );
}
```

actuate.report.Table.showDetail

Syntax void Table.showDetail(string columnName)

Displays information for a column from the grouped data displayed on the page. If every column is hidden, only the group name is visible.

Parameters **columnName**
String. The data binding name for the column to display.

Example This example shows the information from the PRODUCTLINE column in the grouped data that is displayed for myTable:

```
function showMyDetail(myTable) {
   myTable.showDetail("PRODUCTLINE");
   myTable.submit( );
}
```

actuate.report.Table.submit

Syntax void Table.submit(function callback)

Submits all the asynchronous operations for this Table element. The submit( ) function triggers an AJAX request to submit all the asynchronous operations. When the server finishes the processing, it returns a response and the results are rendered on the page in the table container.

Parameters **callback**
Function. The function called after the asynchronous call processing finishes.

Example This example clears existing filters from the PRODUCTLINE column and pops up an alert box:

```
function alertResetFilter(myTable){
   myTable.clearFilters("PRODUCTLINE");
   myTable.submit(alert("Filters Cleared"));
}
```

actuate.report.Table.swapColumns

Syntax void Table.swapColumns(string columnName1, string columnName2)

Swaps the columns to reorder to column sequence of the Table.

Parameters **columnName1**
String. The first column to swap in the column order.

columnName2
String. The second column to swap in the column order.

Example This example swaps the TOTAL and PRODUCTLINE columns in myTable:

```
function swapMyColumns(myTable){
   myTable.swapColumns("TOTAL", "PRODUCTLINE");
   myTable.submit( );
}
```

Class actuate.report.TextItem

Description A container for a Text element in a report. TextItem provides functions to operate on a Text element, such as retrieving the text value and getting the HTML DOM element from the report Text element.

Constructor

The TextItem object is constructed by viewer.PageContent.getTextByBookmark( ).

Function summary

Table 4-37 lists actuate.report.TextItem functions.

Table 4-37 actuate.report.TextItem functions

Function	Description
getBookmark( )	Returns the bookmark name for this Text
getHtmlDom( )	Returns the HTML element for this Text
getInstanceId( )	Returns the report element instance id
getPageContent( )	Returns the page content to which this element belongs
getText( )	Returns the text in this Text element
getType( )	Returns the Text element's type
hide( )	Hides this element
show( )	Shows this element
submit( )	Applies changes made to this element

actuate.report.TextItem.getBookmark

Syntax string TextItem.getBookmark( )

Returns the bookmark name for this Text item.

Returns String.

Example This example displays the table's bookmark in an alert box:

```
function alertBookmark(myTextItem){
   alert(myTextItem.getBookmark( ));
}
```

actuate.report.TextItem.getHtmlDom

Syntax HTMLElement TextItem.getHtmlDom( )

Returns the HTML element for this Text.

Returns HTMLElement.

Example This example displays the HTML DOM element for this Text item inside a red border:

```
function showHtmlDom(myTextItem){
   var domNode = myTextItem.getHtmlDom( );
   var box = document.createElement('div');
   box.style.border = '2px solid red';
   var label = document.createElement('h2');
   label.innerHTML = 'The HTML DOM:';
   box.appendChild(label);
   box.appendChild(domNode);
   document.body.appendChild(box);
}
```

actuate.report.TextItem.getInstanceId

Syntax string TextItem.getInstanceId( )

Returns the instance id of this report element.

Returns String. The instance id.

Example This example displays the instance ID of the report element in an alert box:

```
function showID(myTextItem){
   var elementID = myTextItem.getInstanceId( );
   alert (elementID);
}
```

actuate.report.TextItem.getPageContent

Syntax actuate.viewer.PageContent TextItem.getPageContent( )

Returns the page content to which this Text belongs.

Returns actuate.viewer.PageContent. report content.

Example This example displays the viewer ID of the page content in an alert box:

```
function showViewID(myTextItem){
   var pageContent = myTextItem.getPageContent( );
   var pageViewerID = pageContent.getViewerId( );
   alert (pageViewerID);
}
```

actuate.report.TextItem.getText

Syntax string TextItem.getText( )

Returns the text of this Text element.

Returns String. The content text.

Example This example displays the text of the myTextItem object in an alert box:

```
alert("Text content for myTextItem is " + myTextItem.getText( ));
```

actuate.report.TextItem.getType

Syntax string TextItem.getType( )

Returns the report element type of this object, which is Text.

Returns String. "Text".

Example This example checks the report element type and displays an alert if the type is not "Text":

```
if (myTextItem.getType( ) != "Text"){
   alert("Type mismatch, report element type is not Text");
}
```

actuate.report.TextItem.hide

Syntax void TextItem.hide( )

Hides this element.

Example This example hides myTextItem:

```
myTextItem.hide( );
myTextItem.submit( );
```

actuate.report.TextItem.show

Syntax void TextItem.show( )

Shows this element.

Example This example shows myTextItem:

```
myTextItem.show( );
myTextItem.submit( );
```

actuate.report.TextItem.submit

Syntax void TextItem.submit(function callback)

Submits all the asynchronous operations for this TextItem. The submit( ) function triggers an AJAX request for all asynchronous operations. The server returns a response after processing. The results render on the page in the TextItem container.

Parameters **callback**
Function. The function to execute after the asynchronous call processing is done.

Example This example uses submit( ) after calling show( ) to show myTextItem:

```
myTextItem.show( );
myTextItem.submit( );
```

Class actuate.ReportExplorer

Description The actuate.ReportExplorer class retrieves and displays a navigable repository or file system interface that enables users to navigate folders and select files. This generic user interface enables the user to browse and select repository contents.

Constructor

Syntax actuate.ReportExplorer(string container)

Constructs a ReportExplorer object, initializing the ReportExplorer component.

Parameters **container**
String. The name of the HTML element that displays the rendered ReportExplorer component or a container object. The constructor initializes the ReportExplorer component but does not render it.

Function summary

Table 4-38 lists actuate.ReportExplorer functions.

Table 4-38 actuate.ReportExplorer functions

Function	Description
getFolderName( )	Gets the root folder name
getLatestVersionOnly( )	Gets the latestVersionOnly flag
getResultDef( )	Gets the resultDef value for this GetFolderItems
getSearch( )	Gets the search value for this GetFolderItems
onUnload( )	Unloads unused JavaScript variables
registerEventHandler( )	Registers the event handler
removeEventHandler( )	Removes the event handler
setContainer( )	Sets the div container
setFolderName( )	Sets the root folder name
setLatestVersionOnly( )	Sets the latestVersionOnly flag
setResultDef( )	Sets the resultDef value for this GetFolderItems
setSearch( )	Sets the search value for this GetFolderItems
setService( )	Sets the JSAPI web service
setStartingFolder( )	Sets the path for the initial folder selection
setUseDescriptionAsLabel( )	Sets flag to use descriptions as file/folder labels

Table 4-38 actuate.ReportExplorer functions

Function	Description
showFoldersOnly( )	Sets the flag to only display folders
submit( )	Applies changes made to this element

actuate.ReportExplorer.getFolderName

Syntax string ReportExplorer.getFolderName( )

Returns the name of the root folder for this ReportExplorer.

Returns String. The folder name.

Example This example displays the root folder's name in an alert box:

```
function alertRootFolder(myReportExplorer){
   alert(myReportExplorer.getFolderName( ));
}
```

actuate.ReportExplorer.getLatestVersionOnly

Syntax boolean ReportExplorer.getLatestVersionOnly( )

Returns the latest version only flag for this ReportExplorer.

Returns Boolean. True indicates that ReportExplorer displays only the latest version of each report.

Example This example displays the latest version only flag in an alert box:

```
function alertLatestVersionFlag(myReportExplorer){
   alert(myReportExplorer.getLatestVersionOnly( ));
}
```

actuate.ReportExplorer.getResultDef

Syntax string[ ] ReportExplorer.getResultDef( )

Returns the results definition.

Returns Array of strings. Valid values are: "Name", "FileType", "Version", "VersionName", "Description", "Timestamp", "Size", and "PageCount".

Example This example displays the results definition an alert box:

```
function alertResultsDefinition(myReportExplorer){
   alert(myReportExplorer.getResultDef( ));
}
```

actuate.ReportExplorer.getSearch

Syntax actuate.ReportExplorer.FileSearch ReportExplorer.getSearch( )

Returns the FileSearch object assigned to this ReportExplorer.

Returns actuate.reportexplorer.FileSearch object. The file search settings.

Example This example sets the FileSearch setting for reportexplorer1 to the FileSearch settings of reportexplorer2:

```
reportexplorer1.setSearch(reportexplorer2.getSearch( ));
```

actuate.ReportExplorer.onUnload

Syntax void ReportExplorer.onUnload( )

Unloads JavaScript variables that are no longer needed by ReportExplorer.

Example This example cleans up unused JavaScript variables for myReportExplorer:

```
myReportExplorer.onUnload( );
```

actuate.ReportExplorer.registerEventHandler

Syntax void ReportExplorer.registerEventHandler(string eventName, function handler)

Registers an event handler to activate for parameter eventName. This function can assign several handlers to a single event.

Parameters **eventName**
String. Event name to capture.

handler
Function. The function to execute when the event occurs. The handler must take two arguments: the ReportExplorer instance that fired the event and an event object specific to the event type.

Example This example registers the errorHandler( ) function to respond to the ON_EXCEPTION event:

```
myReportExplorer.registerEventHandler(actuate.ReportExplorer.
                    EventConstants.ON_EXCEPTION, errorHandler);
```

actuate.ReportExplorer.removeEventHandler

Syntax void ReportExplorer.removeEventHandler(string eventName, function handler)

Removes an event handler to activate for parameter eventName.

Parameters **eventName**
String. Event name to remove from the internal list of registered events.

handler
Function. The function to disable.

Example This example removes the errorHandler( ) function from responding to the ON_EXCEPTION event:

```
myReportExplorer.removeEventHandler(actuate.ReportExplorer.
                          EventConstants.ON_EXCEPTION, errorHandler);
```

actuate.ReportExplorer.setContainer

Syntax void ReportExplorer.setContainer(string containerId)

Sets the HTML element container for the ReportExplorer content.

Parameters **containerID**
String. The name of the HTML element that displays the group of rendered ReportExplorer components.

Example This example sets MyReportExplorer to render the <div> element labeled "History":

```
myReportExplorer.setContainer("History");
```

actuate.ReportExplorer.setFolderName

Syntax void ReportExplorer.setFolderName(string folderName)

Sets the name of the root folder for this ReportExplorer.

Parameters **folderName**
String. The name of the repository folder to use as the root folder. Use a repository path to use subfolders for the root folder.

Example This example sets the report explorer root folder to /Public:

```
myReportExplorer.setFolderName("/Public");
```

actuate.ReportExplorer.setLatestVersionOnly

Syntax void ReportExplorer.setLatestVersionOnly(boolean latestVersionOnly)

Sets the latest version only flag for this ReportExplorer.

Parameters **latestVersionOnly**
Boolean. True removes all but the latest versions from the report explorer.

Example This example sets ReportExplorer to display only the latest versions of all files:

```
myReportExplorer.setLatestVersionOnly( true );
```

actuate.ReportExplorer.setResultDef

Syntax void ReportExplorer.setResultDef(string[ ] resultDef)

Sets the results definition.

Parameters **resultDef**
Array of strings. Valid values are: "Name", "FileType", "Version", "VersionName", "Description", "Timestamp", "Size", and "PageCount".

Example This example sets the result set to five columns of data including name, file type, version, version name, and description:

```
var resultDef = "Name|FileType|Version|VersionName|Description";
myReportExplorer.setResultDef( resultDef.split("|") );
```

actuate.ReportExplorer.setSearch

Syntax void ReportExplorer.setSearch(actuate.ReportExplorer.FileSearch search)

Assigns a FileSearch object to this ReportExplorer.

Parameters **search**
actuate.reportexplorer.FileSearch object. The file search settings.

Example This example sets the FileSearch setting for reportexplorer1 to the FileSearch settings of reportexplorer2:

```
reportexplorer1.setSearch(reportexplorer2.getSearch( ));
```

actuate.ReportExplorer.setService

Syntax void ReportExplorer.setService(string iportalURL, actuate.RequestOptions requestOptions)

Sets the target service URL to which this explorer links. When the service URL is not set, this viewer links to the default service URL which is set on the actuate object.

Parameters **iPortalURL**
String. The target Actuate web application URL, either a Java Component or Information Console.

requestOptions
actuate.RequestOptions object. Optional. requestOptions defines URL parameters to send with the authentication request, such as the iHub URL, Encyclopedia volume, or repository type. The URL can also include custom parameters.

Example This example sets the URL for the Actuate iPortal web application service:

```
myExplorer.setService("http://localhost:8700
                          /iportal", myRequestOptions);
```

actuate.ReportExplorer.setStartingFolder

Syntax void ReportExplorer.setStartingFolder(string strfoldername)

Sets the fully qualified path of the initially selected folder in the explorer tree.

Parameters **strfoldername**
String. The fully qualified path of a folder.

Example This example sets the initially selected folder to Public in the local repository:

```
myExplorer.setStartingFolder("C:\Program Files\Actuate11\iHub2
   \servletcontainer\iportal\WEB-INF\repository\Public");
```

actuate.ReportExplorer.setUseDescriptionAsLabel

Syntax void ReportExplorer.setUseDescriptionAsLabel(boolean useDescription)

Sets the explorer to display the folder description as the folder label instead of the folder name.

Parameters **useDescription**
Boolean. True displays descriptions for folders instead of folder names.

Example This example displays descriptions for folders instead of folder names:

```
myExplorer.setUseDescriptionAsLabel(true);
```

actuate.ReportExplorer.showFoldersOnly

Syntax void ReportExplorer.showFoldersOnly(boolean flag)

Sets ReportExplorer to display folders but not files.

Parameters **flag**
Boolean. True displays folders but not files.

Example This example displays folders in ReportExplorer but not files:

```
myExplorer.showFoldersOnly(true);
```

actuate.ReportExplorer.submit

Syntax void ReportExplorer.submit(function callback)

Submits requests to the server for ReportExplorer. When this function is called, an AJAX request is triggered to submit all the operations. When the server finishes the processing, it returns a response and the results are rendered on the page in the ReportExplorer container.

Parameters **callback**
Function. The function to execute after the asynchronous call processing is done.

Example This example submits ReportExplorer with a root folder that set with setStartingFolder( ) and result definition set with setResultDef( ):

```
myExplorer.setStartingFolder("/Dashboard/Contents");
var resultDef = "Name|FileType|Version|VersionName|Description";
myExplorer.setResultDef( resultDef.split("|") );
myExplorer.submit( );
```

Class actuate.reportexplorer.Constants

Description Global constants used for ReportExplorer class. Table 4-39 lists the constants used for the ReportExplorer class.

Table 4-39 Actuate iPortal JavaScript API ReportExplorer constants

Event	Description
ERR_CLIENT	Constant used to tell JSAPI user that there was a client-side error
ERR_SERVER	Constant used to tell JSAPI user that there was a server-side error
ERR_USAGE	Constant used to tell JSAPI user that there was a usage API error
NAV_FIRST	Constant reference for the first page navigation link
NAV_LAST	Constant reference for the last page navigation link
NAV_NEXT	Constant reference for the next page navigation link
NAV_PREV	Constant reference for the previous page navigation link

Class actuate.reportexplorer.EventConstants

Description Defines the event constants supported by this API for report explorers. Table 4-40 lists the ReportExplorer event constants.

Table 4-40 Actuate JavaScript API ReportExplorer event constants

Event	Description
ON_EXCEPTION	Event triggered when an exception occurs. An event handler registered to this event must take an actuate.Exception object as an input argument. The Exception object contains the exception information.
ON_SELECTION_CHANGED	Event triggered when a selection change occurs. For example, this event triggers if the value of a ReportExplorer list control changes. An event handler registered to this event must take an actuate.ReportExplorer.File object corresponding to the file object in which the selection occurred and a string that contains a repository path as input arguments.
ON_SESSION_TIMEOUT	Event triggered when a user attempts to perform any operation after a session has timed out and chooses yes on a prompt dialog asking whether or not to reload the page content. An event handler registered to this event takes no input arguments.

Class actuate.reportexplorer.File

Description A reference object for displaying and controlling a file reference.

Constructor

Syntax actuate.reportexplorer.File( )

Constructs a new File object.

Function summary

Table 4-41 lists actuate.reportexplorer.File functions.

Table 4-41 actuate.reportexplorer.File functions

Function	Description
getAccessType( )	Gets the accessType value for this File
getDescription( )	Gets the description value for this File
getFileType( )	Gets the fileType value for this File
getId( )	Gets the id value for this File
getName( )	Gets the name value for this File
getOwner( )	Gets the owner value for this File
getPageCount( )	Gets the pageCount value for this File
getSize( )	Gets the size value for this File
getTimeStamp( )	Gets the timeStamp value for this File
getUserPermissions( )	Gets the userPermissions value for this File
getVersion( )	Gets the version value for this File
getVersionName( )	Gets the versionName value for this File
setAccessType( )	Sets the accessType value for this File
setDescription( )	Sets the description value for this File
setFileType( )	Sets the fileType value for this File
setId( )	Sets the id value for this File
setName( )	Sets the name value for this File
setOwner( )	Sets the owner value for this File
setPageCount( )	Sets the pageCount value for this File
setSize( )	Sets the size value for this File

(continues)

Table 4-41 actuate.reportexplorer.File functions (continued)

Function	Description
setTimeStamp( )	Sets the timeStamp value for this File
setUserPermissions( )	Sets the userPermissions value for this File
setVersion( )	Sets the version value for this File
setVersionName( )	Sets the versionName value for this File

actuate.reportexplorer.File.getAccessType

Syntax string File.getAccessType( )

Gets the access type.

Returns String. Either "private" or "shared" according to whether the file has been shared or not.

Example To stop a script from running if a file is private, use code similar to the following:

```
if(file.getAccessType( ) == "private"){ return;}
```

actuate.reportexplorer.File.getDescription

Syntax string File.getDescription( )

Gets the description from the file.

Returns String. The description.

Example To stop a script from running if a file does not have a description, use code similar to the following:

```
if(file.getDescription( ) == (null || "")){ return;}
```

actuate.reportexplorer.File.getFileType

Syntax string File.getFileType( )

Gets the file extension for this File.

Returns String. The file type.

Example To store the file extension of the File object file in a variable called type, use code similar to the following:

```
var type = file.getFileType( );
```

actuate.reportexplorer.File.getId

Syntax integer File.getId( )

Gets the file ID value.

Returns Integer. The file ID.

Example To store the file id of the File object file in a variable called id, use code similar to the following:

```
var id = file.getFileId( );
```

actuate.reportexplorer.File.getName

Syntax string File.getName( )

Gets the name of the file.

Returns String. The file name.

Example To store the name of the File object file in a variable called name, use code similar to the following:

```
var name = file.getName( );
```

actuate.reportexplorer.File.getOwner

Syntax string File.getOwner( )

Gets the name of the File's owner.

Returns String. The owner's name

Example To store the name of the owner of the File object file in a variable called owner, use code similar to the following:

```
var owner = file.getOwner( );
```

actuate.reportexplorer.File.getPageCount

Syntax integer File.getPageCount( )

Gets the number pages in the file, if applicable.

Returns Integer. The number of pages.

Example To halt a script if the number of pages exceeds 100 in the file referenced by the File object largefile, use code similar to the following:

```
if (largefile.getPageCount( ) > 100) {return;}
```

actuate.reportexplorer.File.getSize

Syntax integer File.getSize( )

Gets the size value for this File.

Returns Integer.

Example To store a File object size in a variable called size, use code similar to the following:

```
var size = file.getSize( );
```

actuate.reportexplorer.File.getTimeStamp

Syntax string File.getTimeStamp( )

Gets the time stamp for this file.

Returns String. A date and time of the file's creation or last modification.

Example To store the time stamp for the file referenced by the File object oldfile in a variable called timestamp, use code similar to the following:

```
var timestamp = oldfile.getTimeStamp( );
```

actuate.reportexplorer.File.getUserPermissions

Syntax string File.getUserPermissions( )

Gets the user permissions.

Returns String. The current user's permissions for this file.

Example To store a file's permissions in the permissions variable, use code similar to the following:

```
var permissions = file.getUserPermissions( );
```

actuate.reportexplorer.File.getVersion

Syntax integer File.getVersion( )

Gets the version of the file.

Returns Integer. The version.

Example To store the file version in the version variable, use code similar to the following:

```
var version = file.getVersion( );
```

actuate.reportexplorer.File.getVersionName

Syntax string File.getVersionName( )

Gets the version name.

Returns String. The version name.

Example To store a version name in the version variable, use code similar to the following:

```
var version = file.getVersionName( );
```

actuate.reportexplorer.File.setAccessType

Syntax void File.setAccessType(string accessType)

Sets the access type.

Parameters **accessType**
String. "private" or "shared" indicating whether the file has been shared or not.

Example To make a file private, use code similar to the following:

```
file.setAccessType("private")
```

actuate.reportexplorer.File.setDescription

Syntax void File.setDescription(string description)

Sets the description from the file.

Parameters **description**
String. The description.

Example To clear a file's description, use code similar to the following:

```
file.setDescription("");
```

actuate.reportexplorer.File.setFileType

Syntax void File.setFileType(string fileType)

Sets the file type for this file.

Parameters **fileType**
String. The file type, which is a file extension.

Example To assign a file's type if none is assigned, use code similar to the following:

```
if (file.getFileType == null) {file.setFileType("txt");}
```

actuate.reportexplorer.File.setId

Syntax void File.setId(integer id)

Sets the file ID value.

Parameters **id**
Integer. A file ID number.

Example To set a file's ID to 42, use code similar to the following:

```
file.setId("42");
```

actuate.reportexplorer.File.setName

Syntax void File.setName(string name)

Sets the name of the file.

Parameters **name**
String. The name.

Example To set a file's name to releasedates, use code similar to the following:

```
file.setName("releasedates");
```

actuate.reportexplorer.File.setOwner

Syntax void File.setOwner(string owner)

Sets the name of the owner.

Parameters **owner**
String. A user name.

Example To set a file's owner to Administrator, use code similar to the following:

```
file.setOwner("Administrator");
```

actuate.reportexplorer.File.setPageCount

Syntax void File.setPageCount(integer pageCount)

Sets the number pages in the file.

Parameters **pageCount**
Integer. The number of pages, which must be less than the current number of pages.

Example To set a File object's page to 100 if available, use code similar to the following:

```
if(file.getPageCount( ) > 100) {file.setPageCount(100);}
```

actuate.reportexplorer.File.setSize

Syntax void File.setSize(integer size)

Sets the size of the file.

Parameters **size**
Integer. File size in bytes.

Example To set a file's size to 0, use code similar to the following:

```
file.setSize(0);
```

actuate.reportexplorer.File.setTimeStamp

Syntax void File.setTimeStamp(string timeStamp)

Sets the time stamp.

Parameters **timeStamp**
String. A date and time of the file's creation or last modification.

Example To set a file's time stamp to the current time, use code similar to the following:

```
var currenttime = new Date( );
file.setTimeStamp(currenttime.toLocaleString( ));
```

actuate.reportexplorer.File.setUserPermissions

Syntax void File.setUserPermissions(string userPermissions)

Sets the user permissions.

Parameters **userPermissions**
String. The current user's permissions for this file.

Example To apply the user permissions for file1 to file2, use code similar to the following:

```
file2.setUserPermissions(file1.getUserPermissions( ));
```

actuate.reportexplorer.File.setVersion

Syntax void File.setVersion(integer version)

Sets the version of the file.

Parameters **version**
Integer. The version.

Example To set the file's version to 1 for the first version, use code similar to the following:

```
file.setVersion(1);
```

actuate.reportexplorer.File.setVersionName

Syntax void File.setVersionName(string versionName)

Sets the version name.

Parameters **versionName**
String. A version name.

Example To set a file's version name to 2004, use code similar to the following:

```
file.setVersionName("2004");
```

Class actuate.reportexplorer.FileCondition

Description Used inactuate.reportexplorer.FileSearch objects for comparison. Contains a display field associated with a filter string called a match. This can be used for the purposes of comparing field values for searching, filtering, or batch operations. For example, a file condition can match the FileType field with rptdesign to identify all the rptdesign files for a filter.

Constructor

Syntax actuate.reportexplorer.FileCondition( )

Constructs a new FileCondition object.

Function summary

Table 4-42 lists actuate.reportexplorer.FileCondition functions.

Table 4-42 actuate.reportexplorer.FileCondition functions

Function	Description
getField( )	Gets the field for this FileCondition
getMatch( )	Gets the match value for this FileCondition
setField( )	Sets the field for this FileCondition
setMatch( )	Sets the match value for this FileCondition

actuate.reportexplorer.FileCondition.getField

Syntax string FileCondition.getField( )

Returns the field for this FileCondition.

Returns String. Possible values are "Name", "FileType", "Description", "PageCount", "Size", "TimeStamp", "Version", "VersionName", and "Owner".

Example To store the display field of fcondition in a variable called field, use code similar to the following:

```
var field = fcondition.getField( );
```

actuate.reportexplorer.FileCondition.getMatch

Syntax string FileCondition.getMatch( )

Returns the match value for this FileCondition.

Returns String. A string for comparison.

Example To store the matching condition of fcondition in a variable called match, use code similar to the following:

```
var match = fcondition.getMatch( );
```

actuate.reportexplorer.FileCondition.setField

Syntax void FileCondition.setField(string field)

Sets the field for the FileCondition.

Parameters **field**
String. Possible values are "Name", "FileType", "Description", "PageCount", "Size", "TimeStamp", "Version", "VersionName", and "Owner".

Example To set the display field to FileType for fcondition, use code similar to the following:

```
fcondition.setField("FileType");
```

actuate.reportexplorer.FileCondition.setMatch

Syntax void FileCondition.setMatch(string match)

Sets the match value for the FileCondition.

Parameters **match**
String. A string for comparison.

Example To set the match value for fcondition to rptdesign, use code similar to the following:

```
fcondition.setField("rptdesign");
```

Class actuate.reportexplorer.FileSearch

Description Searches the contents of files according to one or more file conditions. FileSearch represents a JavaScript version of com.actuate.schemas.FileSearch.

Constructor

Syntax actuate.reportexplorer.FileSearch( )

Constructs a new FileSearch object.

Function summary

Table 4-43 lists actuate.reportexplorer.FileSearch functions.

Table 4-43 actuate.reportexplorer.FileSearch functions

Function	Condition
getAccessType( )	Gets the accessType value for this FileSearch
getCondition( )	Gets the condition value for this FileSearch
getConditionArray( )	Gets the ConditionArray value for this FileSearch
getCountLimit( )	Gets the countLimit value for this FileSearch
getDependentFileId( )	Gets the id value for this FileSearch
getDependentFileName( )	Gets the file name value for this FileSearch
getFetchDirection( )	Gets the fetch direction for this FileSearch
getFetchHandle( )	Gets the fetchHandle value for this FileSearch
getFetchSize( )	Gets the fetchSize value for this FileSearch
getIncludeHiddenObject( )	Gets the includeHiddenObject value for this FileSearch
getOwner( )	Gets the owner
getPrivilegeFilter( )	Gets the privilegeFilter value for this FileSearch
getRequiredFileId( )	Gets the requiredFileId for this FileSearch
getRequiredFileName( )	Gets the requiredFileName value for this FileSearch
setAccessType( )	Sets the accessType value for this FileSearch
setCondition( )	Sets the condition value for this FileSearch
setConditionArray( )	Sets the ConditionArray value for this FileSearch

(continues)

Table 4-43 actuate.reportexplorer.FileSearch functions (continued)

Function	Condition
setCountLimit( )	Sets the id value for this FileSearch
setDependentFileId( )	Sets the id value for this FileSearch
setDependentFileName( )	Sets the file name value for this FileSearch
setFetchDirection( )	Sets the owner value for this FileSearch
setFetchHandle( )	Sets the fetchHandle value for this FileSearch
setFetchSize( )	Sets the fetchSize value for this FileSearch
setIncludeHiddenObject( )	Sets the timeStamp value for this FileSearch
setOwner( )	Sets the Owner
setPrivilegeFilter( )	Sets the PrivilegeFilter value for this FileSearch
setRequiredFileId( )	Sets the requiredFileId for this FileSearch
setRequiredFileName( )	Sets the requiredFileName value for this FileSearch

actuate.reportexplorer.FileSearch.getAccessType

Syntax string FileSearch.getAccessType( )

Gets the access type.

Returns String. Either "private" or "shared" according to whether the FileSearch has been shared or not.

Example To halt a script if a FileSearch is private, use code similar to the following:

```
if(fsearch.getAccessType( ) == "private"){ return;}
```

actuate.reportexplorer.FileSearch.getCondition

Syntax actuate.reportexplorer.FileCondition FileSearch.getCondition( )

Gets the condition from the FileSearch.

Returns actuate.reportexplorer.FileCondition object. A condition to apply in a search.

Example To halt a script if a FileSearch does not have a condition, use code similar to the following:

```
if(fsearch.getCondition( ) == null){ return;}
```

actuate.reportexplorer.FileSearch.getConditionArray

Syntax actuate.reportexplorer.FileCondition[ ] FileSearch.getConditionArray( )

Gets the file condition array for this FileSearch.

Returns Array of actuate.reportexplorer.FileCondition objects. Multiple conditions to apply in a search.

Example To retrieve the array of file conditions from the FileSearch object fsearch, use code similar to the following:

```
var conditions = new Array( );
conditions = fsearch.getConditionArray( );
```

actuate.reportexplorer.FileSearch.getCountLimit

Syntax integer FileSearch.getCountLimit( )

Gets the maximum number of match results to display set for this file search.

Returns Integer. The maximum number of match results to display. 0 indicates unlimited.

Example To retrieve the count limit from the FileSearch object fsearch, use code similar to the following:

```
var limit = fsearch.getCountLimit( );
```

actuate.reportexplorer.FileSearch.getDependentFileId

Syntax string FileSearch.getDependentFileId( )

Gets the file ID of the FileSearch, identifying the file it is set to search.

Returns String. The file ID.

Example To retrieve the file Id from the FileSearch object fsearch, use code similar to the following:

```
var id = fsearch.getDependantFileId( );
```

actuate.reportexplorer.FileSearch .getDependentFileName

Syntax string FileSearch.getDependentFileName( )

Gets the file name of the FileSearch.

Returns String. The file name.

Example To retrieve the file name from the FileSearch object fsearch, use code similar to the following:

```
var name = fsearch.getDependantFileName( );
```

actuate.reportexplorer.FileSearch.getFetchDirection

Syntax boolean FileSearch.getFetchDirection( )

Gets the fetch direction of the FileSearch.

Returns Boolean. True indicates ascending order.

Example To switch the fetch direction for the FileSearch object fsearch, use code similar to the following:

```
fsearch.setFetchDirection(!fsearch.getFetchDirection( ));
```

actuate.reportexplorer.FileSearch.getFetchHandle

Syntax string FileSearch.getFetchHandle( )

Gets the fetch handle.

Returns String. The fetch handle.

Example To retrieve the fetch handle from the FileSearch object fsearch, use code similar to the following:

```
var handle = fsearch.getFetchHandle( );
```

actuate.reportexplorer.FileSearch.getFetchSize

Syntax integer FileSearch.getFetchSize( )

Gets the fetch size.

Returns Integer. The fetch size.

Example To halt a script if a FileSearch has a fetch size of 0, use code similar to the following:

```
if(fsearch.getFetchSize( ) == 0){ return;}
```

actuate.reportexplorer.FileSearch .getIncludeHiddenObject

Syntax boolean FileSearch.getIncludeHiddenObject( )

Gets the includeHiddenObject value for this FileSearch.

Returns Boolean. True includes hidden object.

Example To alert the user that hidden objects are enabled for a FileSearch, use code similar to the following:

```
if(fsearch.getIncludeHiddenObejct( )){
   alert("Hidden objects are enabled.");
}
```

actuate.reportexplorer.FileSearch.getOwner

Syntax string FileSearch.getOwner( )

Gets the owner's name.

Returns String. The owner's user name.

Example To retrieve the owner of fsearch, use code similar to the following:

```
var owner = fsearch.getOwner( );
```

actuate.reportexplorer.FileSearch.getPrivilegeFilter

Syntax actuate.reportexplorer.PrivilegeFilter FileSearch.getPrivilegeFilter( )

Gets the privilege filter.

Returns actuate.reportexplorer.PrivilegeFilter object. A privilege filter.

Example To retrieve the privilege filter for fsearch, use code similar to the following:

```
var privileges = fsearch.getPrivilegeFilter( );
```

actuate.reportexplorer.FileSearch.getRequiredFileId

Syntax integer FileSearch.getRequiredFileId( )

Gets the requiredFileId of FileSearch.

Returns Integer. A field ID.

Example To retrieve the required field ID assigned to fsearch, use code similar to the following:

```
var id = fsearch.getRequiredFileId( );
```

actuate.reportexplorer.FileSearch .getRequiredFileName

Syntax string FileSearch.getRequiredFileName( )

Gets the requiredFileName name.

Returns String. A file name.

Example To retrieve the file name assigned to fsearch, use code similar to the following:

```
var id = fsearch.getRequiredFileName( );
```

actuate.reportexplorer.FileSearch.setAccessType

Syntax void FileSearch.setAccessType(string accessType)

Sets the access type.

Parameters **accessType**
String. Either "private" or "shared" according to whether FileSearch has been shared or not.

Example To make a FileSearch fsearch private, use code similar to the following:

```
fsearch.setAccessType("private");
```

actuate.reportexplorer.FileSearch.setCondition

Syntax void FileSearch.setCondition(actuate.reportExplorer.FileCondition condition)

Sets a search condition for this FileSearch.

Parameters **condition**
actuate.reportexplorer.FileCondition object. A condition to apply to this search.

Example To clear FileSearch fsearch's condition, use code similar to the following:

```
fsearch.setCondition(null);
```

actuate.reportexplorer.FileSearch.setConditionArray

Syntax void FileSearch.setConditionArray(actuate.reportExplorer.FileCondition[ ] ConditionArray)

Sets multiple search conditions for this FileSearch.

Parameters **ConditionArray**
Array of actuate.reportexplorer.FileCondition objects. Conditions to apply to this search.

Example To clear FileSearch fsearch's condition array, use code similar to the following:

```
fsearch.setConditionArray(null);
```

actuate.reportexplorer.FileSearch.setCountLimit

Syntax void FileSearch.setCountLimit(integer countlimit)

Sets the maximum number of match results to display.

Parameters **countlimit**
Integer. The maximum number of match results to display. 0 indicates unlimited.

Example To set FileSearch fsearch to stop searching after finding 100 matches, use code similar to the following:

```
fsearch.setCountLimit(100);
```

actuate.reportexplorer.FileSearch.setDependentFileId

Syntax void FileSearch.setDependentFileId(string dependentFileId)

Sets the file ID of the FileSearch.

Parameters **dependentFileId**
String. A file ID.

Example To set FileSearch fsearch's File ID to current, use code similar to the following:

```
fsearch.setDependentFileId("current");
```

actuate.reportexplorer.FileSearch .setDependentFileName

Syntax void FileSearch.setDependentFileName(string dependentFileName)

Sets the file name of FileSearch.

Parameters **dependentFileName**
String. A file name.

Example To set FileSearch fsearch's file name to current, use code similar to the following:

```
fsearch.setDependentFileName("current");
```

actuate.reportexplorer.FileSearch.setFetchDirection

Syntax void FileSearch.setFetchDirection(boolean fetchDirection)

Sets the fetch direction for this FileSearch.

Parameters **fetchDirection**
Boolean. True indicates ascending order.

Example To switch the fetch direction for the FileSearch object fsearch, use code similar to the following:

```
fsearch.setFetchDirection(!fsearch.getFetchDirection( ));
```

actuate.reportexplorer.FileSearch.setFetchHandle

Syntax void FileSearch.setFetchHandle(string fetchHandle)

Sets the fetch handle for FileSearch.

Parameters **fetchHandle**
String. A fetch handle.

Example To set FileSearch fsearch's fetch handle to ezsearch, use code similar to the following:

```
fsearch.setFetchHandle("ezsearch");
```

actuate.reportexplorer.FileSearch.setFetchSize

Syntax void FileSearch.setFetchSize(integer fetchSize)

Sets the fetch size.

Parameters **fetchSize**
Integer. The fetch size.

Example To set FileSearch fsearch's fetch size to 12, use code similar to the following:

```
fsearch.setFetchSize(12);
```

actuate.reportexplorer.FileSearch .setIncludeHiddenObject

Syntax void FileSearch.setIncludeHiddenObject(boolean includeHiddenObject)

Sets the includeHiddenObject value for this FileSearch.

Parameters **includeHiddenObject**
Boolean. True includes hidden object.

Example To prohibit FileSearch fsearch from including hidden objects, use code similar to the following:

```
fsearch.setIncludeHiddenObject(false);
```

actuate.reportexplorer.FileSearch.setOwner

Syntax void FileSearch.setOwner(string owner)

Sets the owner for this FileSearch.

Parameters **owner**
String. The owner's user name.

Example To set FileSearch fsearch's owner to administrator, use code similar to the following:

```
fsearch.setOwner("administrator");
```

actuate.reportexplorer.FileSearch.setPrivilegeFilter

Syntax void FileSearch.setPrivilegeFilter(actuate.reportexplorer.PrivilegeFilter privilegeFilter)

Sets the privilege filter.

Parameters **privilegeFilter**
actuate.reportexplorer.PrivilegeFilter object. The privilege filter.

Example To assign the privilege filter pfilter to the FileSearch fsearch, use code similar to the following:

```
fsearch.setPrivilegeFilter(pfilter);
```

actuate.reportexplorer.FileSearch.setRequiredFileId

Syntax void FileSearch.setRequiredFileId(string requiredFileId)

Sets the requiredFileId for this FileSearch.

Parameters **requiredFileId**
String. A file ID.

Example To set FileSearch fsearch's File ID to permanent, use code similar to the following:

```
fsearch.setRequiredFileId("permanent");
```

actuate.reportexplorer.FileSearch .setRequiredFileName

Syntax void FileSearch.setRequiredFileName(string requiredFileName)

Sets the required file name.

Parameters **requiredFileName**
String. A file name.

Example To set FileSearch fsearch's file name to permanent, use code similar to the following:

```
fsearch.setRequiredFileName("permanent");
```

Class actuate.reportexplorer.FolderItems

Description A container for the contents of a folder. FolderItems represents a JavaScript version of com.actuate.schemas.GetFolderItemsResponse.

Constructor

Syntax actuate.reportexplorer.FolderItems( )

Constructs a new FolderItems object.

Function summary

Table 4-44 lists actuate.reportexplorer.FolderItems functions.

Table 4-44 actuate.reportexplorer.FolderItems functions

Function	Description
getFetchHandle( )	Gets the fetchHandle value for GetFolderItemsResponse
getItemList( )	Gets the itemList value for GetFolderItemsResponse
getTotalCount( )	Gets the totalCount value for GetFolderItemsResponse
setFetchHandle( )	Sets the fetchHandle value for GetFolderItemsResponse
setItemList( )	Sets the itemList value for GetFolderItemsResponse
setTotalCount( )	Sets the totalCount value for GetFolderItemsResponse

actuate.reportexplorer.FolderItems.getFetchHandle

Syntax string FolderItems.getFetchHandle( )

Retrieves the fetch handle for this folder's contents.

Returns String. The fetch handle.

Example To retrieve the fetch handle from fitems, use code similar to the following:

```
var handle = fitems.getFetchHandle( );
```

actuate.reportexplorer.FolderItems.getItemList

Syntax actuate.reportexplorer.File[ ] FolderItems.getItemList( )

Gets the list of file contents for the folder.

Returns Array of actuate.reportexplorer.File objects. A list of the folder contents.

Example To store fitems' item list in the files variable, use code similar to the following:

```
files = fitems.getItemList( );
```

actuate.reportexplorer.FolderItems.getTotalCount

Syntax string FolderItems.getTotalCount( )

Returns the maximum number of list items to retrieve from this folder.

Returns String. The total count.

Example To retrieve the total count from fitems, use code similar to the following:

```
var count = fitems.getTotalCount( );
```

actuate.reportexplorer.FolderItems.setFetchHandle

Syntax void FolderItems.setFetchHandle(string fetchHandle)

Sets the fetch handle value for this FolderItems object.

Parameters **fetchHandle**
String. The fetch handle.

Example To set FolderItems fitems' fetch handle to dir, use code similar to the following:

```
fitems.setFetchHandle("dir");
```

actuate.reportexplorer.FolderItems.setItemList

Syntax void FolderItems.setItemList(ctuate.reportexplorer.File[ ] itemList)

Sets the list of contents for this folder.

Parameters **itemList**
Array of actuate.reportexplorer.File objects. A list of the folder contents.

Example To assign the item list from fitems1 to fitems2, use code similar to the following:

```
fitems2.setItemList(fitems1.getItemList( ));
```

actuate.reportexplorer.FolderItems.setTotalCount

Syntax void FolderItems.setDataType(string totalCount)

Sets the maximum number of list items to retrieve from this folder.

Parameters **totalCount**
String. The total count.

Example To reset the count total for fitems, use code similar to the following:

```
fitems.setTotalCount("0");
```

Class actuate.reportexplorer.PrivilegeFilter

Description The PrivilegeFilter class contains a set of user-identifying information and access rights that are associated with identified users. PrivilegeFilter represents a JavaScript version of com.actuate.schemas.PrivilegeFilter.

Constructor

Syntax actuate.reportexplorer.PrivilegeFilter( )

Constructs a new PrivilegeFilter object.

Function summary

Table 4-45 lists actuate.reportexplorer.PrivilegeFilter functions.

Table 4-45 actuate.reportexplorer.PrivilegeFilter functions

Function	Description
getAccessRights( )	Gets the accessRights value for this PrivilegeFilter
getGrantedRoleId( )	Gets the grantedRoleId value for this PrivilegeFilter
getGrantedRoleName( )	Gets the grantedRoleName value for this PrivilegeFilter
getGrantedUserId( )	Gets the grantedUserId value for this PrivilegeFilter
getGrantedUserName( )	Gets the grantedUserName value for this PrivilegeFilter
setAccessRights( )	Sets the accessRights value for this PrivilegeFilter
setGrantedRoleId( )	Sets the grantedRoleId value for this PrivilegeFilter
setGrantedRoleName( )	Sets the grantedRoleName value for this PrivilegeFilter
setGrantedUserId( )	Sets the grantedUserId value for this PrivilegeFilter
setGrantedUserName( )	Sets the grantedUserName value for this PrivilegeFilter

actuate.reportexplorer.PrivilegeFilter .getAccessRights

Syntax string privilegeFilter.getAccessRights( )

Gets the repository access rights value for this PrivilegeFilter.

Returns String. Repository access rights.

Example To halt a script if a PrivilegeFilter pfilter's access rights are null, use code similar to the following:

```
if(pfilter.getAccessRights( ) == null){ return;}
```

actuate.reportexplorer.PrivilegeFilter .getGrantedRoleId

Syntax string PrivilegeFilter.getGrantedRoleId( )

Gets the grantedRoleId value for this PrivilegeFilter.

Returns String. A role ID.

Example To retrieve the granted role ID for a PrivilegeFilter pfilter, use code similar to the following:

```
var roleid = pfilter.getGrantedRoleId( );
```

actuate.reportexplorer.PrivilegeFilter .getGrantedRoleName

Syntax string PrivilegeFilter.getGrantedRoleName( )

Gets the grantedRoleName value for this PrivilegeFilter.

Returns String. A role name.

Example To retrieve the granted role name for a PrivilegeFilter pfilter, use code similar to the following:

```
var rolename = pfilter.getGrantedRoleName( );
```

actuate.reportexplorer.PrivilegeFilter .getGrantedUserId

Syntax string PrivilegeFilter.getGrantedUserId( )

Gets the grantedUserId value for this PrivilegeFilter.

Returns String. A user ID.

Example To retrieve the granted user ID for a PrivilegeFilter pfilter, use code similar to the following:

```
var userid = pfilter.getGrantedUserId( );
```

actuate.reportexplorer.PrivilegeFilter .getGrantedUserName

Syntax string PrivilegeFilter.getGrantedUserName( )

Gets the grantedUserName value for this PrivilegeFilter.

Returns String. A user name.

Example To retrieve the granted user name for a PrivilegeFilter pfilter, use code similar to the following:

```
var username = pfilter.getGrantedUserName( );
```

actuate.reportexplorer.PrivilegeFilter .setAccessRights

Syntax void PrivilegeFilter.setAccessRights(string accessRights)

Sets the repository access rights value for this PrivilegeFilter.

Parameters **accessRights**
String. The access rights.

Example To copy the set of access rights from PrivilegeFilter pfilter1 to PrivilegeFilter pfilter2, use code similar to the following:

```
pfilter2.setAccessRights(pfilter1.getAccessRights( ));
```

actuate.reportexplorer.PrivilegeFilter .setGrantedRoleId

Syntax void PrivilegeFilter.setGrantedRoleId(string grantedRoleId)

Sets the grantedRoleId of the column for this PrivilegeFilter.

Parameters **grantedRoleId**
String. A role ID.

Example To set the granted role ID of the PrivilegeFilter pfilter to All, use code similar to the following:

```
pfilter.setGrantedRoleId("All");
```

actuate.reportexplorer.PrivilegeFilter .setGrantedRoleName

Syntax void PrivilegeFilter.setGrantedRoleName(string grantedRoleName)

Sets the grantedRoleName value for this PrivilegeFilter.

Parameters **grantedRoleName**
String. A role name.

Example To set the granted role name of the PrivilegeFilter pfilter to Everyone, use code similar to the following:

```
pfilter.setGrantedRoleName("Everyone");
```

actuate.reportexplorer.PrivilegeFilter .setGrantedUserId

Syntax void PrivilegeFilter.setGrantedUserId(string grantedUserId)

Sets the grantedUserId value for this PrivilegeFilter.

Parameters **grantedUserId**
String. A user ID.

Example To set the granted user ID of the PrivilegeFilter pfilter to administrator, use code similar to the following:

```
pfilter.setGrantedRoleId("Administrator");
```

actuate.reportexplorer.PrivilegeFilter .setGrantedUserName

Syntax void PrivilegeFilter.setGrantedUserName(string grantedUserName)

Sets the grantedUserName value for this PrivilegeFilter.

Parameters **grantedUserName**
String. A user name.

Example To set the granted user name of the PrivilegeFilter pfilter to administrator, use code similar to the following:

```
pfilter.setGrantedRoleId("Administrator");
```

Class actuate.RequestOptions

Description The request options that loginServlet requires to authenticate requests. RequestOptions is used by other classes to provide authentication information. It also adds any customized options to the request URL.

Constructor

Syntax actuate.RequestOptions(actuate.RequestOptions requestOptions)

Constructs a new RequestOptions object.

Parameters **requestOptions**
actuate.RequestOptions object. Optional. Provides request option settings to copy into this RequestOptions object.

Function summary

Table 4-46 lists actuate.RequestOptions functions.

Table 4-46 actuate.RequestOptions functions

Function	Description
getIServerUrl( )	Returns the BIRT iHub URL value
getLocale( )	Returns the current locale
getRepositoryType( )	Returns the repository type
getVolume( )	Returns the Encyclopedia volume
getVolumeProfile( )	Returns the volume profile
setCustomParameters( )	Appends a custom report explorer to the request URL
setIServerUrl( )	Sets the BIRT iHub URL value
setLocale( )	Sets the locale
setRepositoryType( )	Sets the repository type: enterprise or workgroup
setVolume( )	Sets the Encyclopedia volume
setVolumeProfile( )	Sets the volume profile

actuate.RequestOptions.getIServerUrl

Syntax string RequestOptions.getIserverurl( )

Returns the BIRT iHub URL.

Returns String. The URL for BIRT iHub.

Example To retrieve the BIRT iHub URL from the RequestOptions object reqOpts, use code similar to the following:

```
var iHubUrl = reqOpts.getIServerUrl( );
```

actuate.RequestOptions.getLocale

Syntax string RequestOptions.getLocale( )

Returns the current locale or null if no locale is set.

Returns String. The locale value; null for default.

Example This example pops up an alert box if the locale value is set to the default:

```
var locale = reqOpts.getLocale( );
if (locale == null){
  alert("Locale value is default");
}
```

actuate.RequestOptions.getRepositoryType

Syntax string RequestOptions.getRepositoryType( )

Returns the repository type: enterprise or workgroup.

Returns String. Valid repository type values are enterprise or workgroup.

Example To retrieve the repository type for the RequestOptions object reqOpts, use code similar to the following:

```
var repositorytype = reqOpts.getRepositoryType( );
```

actuate.RequestOptions.getVolume

Syntax string RequestOptions.getVolume( )

Returns the Encyclopedia volume.

Returns String. The name of the Encyclopedia volume.

Example To retrieve the Encyclopedia volume for the RequestOptions object reqOpts, use code similar to the following:

```
var encyVol = reqOpts.getVolume( );
```

actuate.RequestOptions.getVolumeProfile

Syntax string RequestOptions.getVolumeProfile( )

Returns the volume profile by name. Valid volume profile names are listed in the service's WEB-INF\volumeProfile.xml file.

Returns String. The volume profile.

Example To retrieve the volume profile for the RequestOptions object reqOpts, use code similar to the following:

```
var volProfile = reqOpts.getVolumeProfile( );
```

actuate.RequestOptions.setCustomParameters

Syntax void RequestOptions.setCustomParameters(object parameters)

Returns a custom parameter in the request URL.

Parameters **parameters**
Object. An associative array of name:value pairs for URL parameters.

Example To add "&myParam=myValue" in a request URL derived from RequestOptions object, use code similar to the following:

```
MyRequestOptions.setCustomParameters({myParam: "myValue"});
```

actuate.RequestOptions.setIServerUrl

Syntax void RequestOptions.setIServerUrl(string iServerUrl)

Sets the BIRT iHub URL.

Parameters **Iserverurl**
String. The BIRT iHub URL value.

Example This example sets the BIRT iHub URL for the reqOpts RequestOptions object:

```
reqOpts.setIserverUrl("http://localhost:8700");
```

actuate.RequestOptions.setLocale

Syntax void RequestOptions.setLocale(string Locale)

Sets the locale.

Parameters **Locale**
String. Optional. The locale value. Null indicates the default locale.

Example This example resets the locale for the reqOpts RequestOptions object to the default:

```
reqOpts.setLocale( );
```

actuate.RequestOptions.setRepositoryType

Syntax void RequestOptions.setRepositoryType(string repositoryType)

Sets the repository type, either enterprise or workgroup.

Parameters **repositoryType**
String. Valid repository type values are enterprise or standalone, as designated by the Actuate web application service with which to connect. Use the following constants:

- actuate.RequestOptions.REPOSITORY_ENCYCLOPEDIA
- actuate.RequestOptions.REPOSITORY_STANDALONE

Example This example sets the repository to workgroup:

```
reqOpts.setRepositoryType(
   actuate.RequestOptions.REPOSITORY_STANDALONE);
```

actuate.RequestOptions.setVolume

Syntax void RequestOptions.setVolume(string volume)

Sets the Encyclopedia volume.

Parameters **volume**
String. The Encyclopedia volume.

Example To set the Encyclopedia volume to marcom if the RequestOptions object reqOpts volume is null, use code similar to the following:

```
if( reqOpts.getVolume( ) == null){
   reqOpts.setVolume("marcom");
}
```

actuate.RequestOptions.setVolumeProfile

Syntax void RequestOptions.setVolumeProfile(string volumeProfile)

Sets the volume profile to use. Valid volume profile names are listed in the service's WEB-INF\volumeProfile.xml file.

Parameters **volumeProfile**
String. The volume profile.

Example To set the volume profile to myServer if the RequestOptions object reqOpts volume profile is null, use code similar to the following:

```
if( reqOpts.getVolume( ) == null){
   reqOpts.setVolumeProvile("myServer");
}
```

Class actuate.Viewer

Description The actuate.Viewer class retrieves and displays Actuate BIRT report contents in an HTML container. The actuate.Viewer class displays the report by page. The goto functions of this class change the current position and page displayed in the viewer.

Constructor

Syntax actuate.Viewer(object viewContainer)

actuate.Viewer(string viewContainerId)

Constructs a new viewer object. The container is an HTML object defined on the HTML page.

Parameters **viewContainer**
Object. A document object that references the <div> element that holds the viewer.

viewContainerId
String. The value of the id parameter for the <div> element that holds the viewer.

Example To assign the viewer to display in a <div id='containerName' /> tag on the page, use the following constructor call:

```
var myViewer = new actuate.Viewer("containerName");
```

Function summary

Table 4-47 lists actuate.Viewer functions.

Table 4-47 actuate.Viewer functions

Function	Description
disableIV( )	Disables Interactive Viewer features
downloadReport( )	Exports a report using the specified format
downloadResultSet( )	Exports data to an external file
enableIV( )	Enables Interactive Viewer features
getChart( )	Retrieves a chart by bookmark
getClientHeight( )	Gets the viewer's height
getClientWidth( )	Gets the viewer's width
getContentByBookmark( )	Gets the report content by bookmark
getContentByPageRange( )	Gets the report content by page range

Table 4-47 actuate.Viewer functions (continued)

Function	Description
getContentMargin( )	Gets the margin dimensions of the content in pixels
getCurrentPageContent( )	Returns the report content displayed in the viewer
getCurrentPageNum( )	Returns the current page number
getDataItem( )	Retrieves a data item by bookmark
getFlashObject( )	Retrieves a Flash object by bookmark
getGadget( )	Retrieves a gadget by bookmark
getHeight( )	Returns the viewer height setting
getHelpBase( )	Gets the help URL
getId( )	Returns the ID of this object
getIportalUrl( )	Returns the Actuate web application URL that this Viewer accesses
getLabel( )	Retrieves a label by bookmark
getReportletBookmark( )	Returns the bookmark of a Reportlet displayed in the viewer
getReportName( )	Returns the report file displayed in the viewer
getRequestOptions( )	Returns the viewer's request options
getTable( )	Retrieves a table by bookmark
getText( )	Retrieves a text element by bookmark
getTotalPageCount( )	Returns the total number of pages
getUIConfig( )	Gets the UIConfig object assigned to the viewer
getUIOptions( )	Returns the UIOptions object
getViewer( )	Returns a viewer object containing the given bookmarked element
getWidth( )	Returns the viewer width setting
gotoBookmark( )	Goes to the position in the report specified by the bookmark
gotoPage( )	Goes to the specified page
isInteractive( )	Returns whether interactive viewing features are enabled

(continues)

Table 4-47 actuate.Viewer functions (continued)

Function	Description
saveReportDesign( )	Saves a report design to the repository
saveReportDocument( )	Saves a report document to the repository
setContentMargin( )	Sets the viewer content margin
setFocus( )	Sets the focus element on the viewer
setHeight( )	Sets the viewer height
setHelpBase( )	Sets the base help URL
setParameters( )	Sets the parameters to run a report using a list of literal string pairs
setParameterValues( )	Sets the parameters to run a report using a generated object
setReportletBookmark( )	Sets bookmark name for a Reportlet
setReportName( )	Sets the report file to render within this Viewer
setService( )	Sets the target service URL
setSize( )	Sets the size of the viewer
setSupportSVG( )	Sets the Scalable Vector Graphic support flag to enable Scalable Vector Graphics content
setUIOptions( )	Sets UIOptions using a UIOptions object
setViewingMode( )	Sets the dashboard viewing mode
setWidth( )	Sets the width of the viewer
showDownloadReportDialog( )	Enables the export report dialog window
showDownloadResultSetDialog( )	Enables the download data dialog window
showFacebookCommentPanel( )	Shows the Facebook comments panel.
showParameterPanel( )	Shows the parameter panel
showPrintDialog( )	Enables the print dialog window
showTocPanel( )	Shows the table of contents panel
submit( )	Submits all the asynchronous operations that the user has requested on this Viewer and renders the viewer component on the page

actuate.Viewer.disableIV

Syntax void Viewer.disableIV(function callback)

Disables the Interactive Viewer features of this viewer object. This is an asynchronous setting committed by submit( ).

Parameters **callback**
Function. The callback function to call after the Interactive Viewer is disabled.

Example To disable the Interactive Viewer option for myViewer, use code similar to the following:

```
myViewer.disableIV(function alertUser( ){alert("IV disabled");});
```

actuate.Viewer.downloadReport

Syntax void Viewer.downloadReport(string format, string pages,
    actuate.viewer.RenderOptions renderoption)

Exports the report with a specified format. The downloadReport function does not return any object. The report is exported to the client side. Then the browser opens a download window for the user to specify a location for the report.

Parameters **format**
String. The format in which to export the report. Valid values and their corresponding formats are:

- afp: IBM Advanced Function Printing
- doc: Word
- docx: Word 2007
- html: HTML-encoded web page
- ppt: PowerPoint
- pptx: PowerPoint 2007
- pdf: Adobe PDF
- ps: PostScript
- xls: Excel
- xlsx: Excel 2007

pages
String. The pages to retrieve. Indicate page ranges by using the first page number of the range and the last page number separated by a dash. To use more than one value, separate individual page numbers or page ranges with commas.

renderoption
actuate.viewer.RenderOptions object. Optional. Sets the rendering options for the download, which currently only applies to multisheet xls format reports.

Example To download the first five pages of the report in the viewer, use the following code:

```
viewer.downloadReport("pdf", "1-5", null);
```

actuate.Viewer.downloadResultSet

Syntax void Viewer.downloadResultSet(actuate.data.Request request, function callback)

Gets all the data from the report as specified by the request. This function makes an AJAX call to the server for the data that is not in the current page. Write a callback function to process the result set. The callback must take an actuate.data.ResultSet object as an argument.

Parameters **request**
actuate.data.Request object. The request to generate the result set.

callback
Function. The callback function to call after retrieving the results. The callback function must take an actuate.data.ResultSet object as an argument.

Example This example creates an actuate.data.ResultSet object from the report in myViewer as specified by myRequest and passes it to a callback function:

```
myViewer.downloadResultSet(myRequest, callback);
```

actuate.Viewer.enableIV

Syntax void Viewer.enableIV(function callback)

Enables interactive viewing features for this Viewer, which enables the selection and modification of report content. This function must be used in the callback of viewer.submit( ) as shown in the following example:

```
function runInteractive(){
myviewer.setReportName("/Public/BIRT and BIRT Studio Examples
   /Sales by Customer.rptdesign");
myviewer.submit(function() {myviewer.enableIV(callback);});
}
```

Parameters **callback**
Function. The callback function to call after enabling the Interactive Viewer features.

Example This function must be used in the callback of viewer.submit( ) as shown in the following example:

```
function runInteractive(){
myviewer.setReportName("/Public/BIRT and BIRT Studio Examples
   /Sales by Customer.rptdesign");
myviewer.submit(function() {myviewer.enableIV(callback);});
}
```

actuate.Viewer.getChart

Syntax actuate.report.Chart Viewer.getChart(string bookmark)

Returns an instance of the chart referenced by a bookmark.

Parameters **bookmark**
String. The bookmark name.

Returns actuate.report.Chart object.

Example This example returns the chart with the bookmark ChartBookmark:

```
function getMyChartByBookmark(myReport) {
   var bviewer = myReport.getViewer("Chart");
   var bpagecontents = bviewer.getCurrentPageContent( );
   return bpagecontents.getChart("ChartBookmark");
}
```

actuate.Viewer.getClientHeight

Syntax integer Viewer.getClientHeight( )

Gets the browser window's height.

Returns Integer. Height in pixels.

Example To reset the viewer height to 20 pixels less than the browser window if it is larger than the browser window, use code similar to the following:

```
if(myViewer.getClientHeight( ) < myViewer.getHeight( )){
   myViewer.setHeight(myViewer.getClientHeight( ) - 20);
}
```

actuate.Viewer.getClientWidth

Syntax integer Viewer.getClientWidth( )

Gets the browser window's width.

Returns Integer. Width in pixels.

Example To reset the viewer width to 20 pixels less than the browser window if it is larger than the browser window, use code similar to the following:

```
if(myViewer.getClientWidth( ) < myViewer.getWidth( )){
   myViewer.setWidth(myViewer.getClientWidth( ) - 20);
}
```

actuate.Viewer.getContentByBookmark

Syntax void Viewer.getContentByBookmark(string bookmark, string format,
function callback)

Gets the report content by bookmark and passes the content as data to a callback.

Parameters **bookmark**
String. The bookmark of a report element to retrieve.

format
String. The output format, which is either html or xhtml.

callback
Function. Callback to be called once the operation is finished. The callback must take actuate.data.ReportContent object as an argument.

Example To retrieve the content with the bookmark FirstChart as html, use code similar to the following:

```
myViewer.getContentByBookmark("FirstChart", "html", processChart);
```

actuate.Viewer.getContentByPageRange

Syntax void Viewer.getContentByPageRange(string PageRange, string format,
function callback)

Gets the report content by Page Range and passes the content as data to a callback.

Parameters **PageRange**
String. Page range to retrieve the report content, separated by a dash.

format
String. The output format, which is either html or xhtml.

callback
Function. Callback to be called once the operation is finished. The callback must take actuate.data.ReportContent object as an argument.

Example To retrieve the content from pages 3 through 5 as html, use code similar to the following:

```
myViewer.getContentByPageRange("3-5", "html", processPages);
```

actuate.Viewer.getContentMargin

Syntax integer | object Viewer.getContentMargin( )

Gets the viewer content margin.

Returns Integer or Object. An integer indicates the same margin on all sides, in pixels. The object contains the pixel values for the top, bottom, left, and right margins of the viewer in an array. For example, a 25-pixel top content margin and no margin in the other directions would be the object array {top:25, left:0, right:0, bottom:0}.

Example To set the margin of the viewer newViewer to match the margin of myViewer, use code similar to the following:

```
newViewer.setContentMargin(myViewer.getContentMargin( ));
```

actuate.Viewer.getCurrentPageContent

Syntax actuate.viewer.Content Viewer.getCurrentPageContent( )

Returns the report content displayed in the viewer. This function is the entry point for retrieving the report elements from this viewer object.

Returns actuate.viewer.PageContent object.

Example Use this function to access the bookmarks for specific elements in the page content. For example, to access the table "mytable" on the page loaded in the myViewer viewer object, use the following code:

```
var element = myViewer.getCurrentPageContent( ).
   getTableByBookmark("mytable");
```

actuate.Viewer.getCurrentPageNum

Syntax integer Viewer.getCurrentPageNum( )

Returns the page number for the page currently being displayed.

Returns Integer. The current page number.

Example This function is useful to move to another page relative to the current page. To go to the next page in a document, use the following code:

```
viewer.gotoPage(viewer.getCurrentPageNum( ) + 1);
```

actuate.Viewer.getDataItem

Syntax actuate.report.DataItem Viewer.getDataItem(string bookmark)

Returns an instance of report data referenced by a bookmark.

Parameters **bookmark**
String. The bookmark name.

Returns actuate.report.DataItem object.

Example To get the report data with the bookmark FirstDataItem and store it in the variable myDataItem, use code similar to the following:

```
var myDataItem = myViewer.getDataItem("FirstDataItem");
```

actuate.Viewer.getFlashObject

Syntax actuate.report.FlashObject Viewer.getFlashObject(string bookmark)

Returns an instance of the Flash object referenced by a bookmark.

Parameters **bookmark**
String. The bookmark name.

Returns actuate.report.FlashObject object.

Example To get the Flash object with the bookmark FirstFlashObject and store it in the variable myFlashObject, use code similar to the following:

```
var myFlashObject = myViewer.getFlashObject("FirstFlashObject");
```

actuate.Viewer.getGadget

Syntax actuate.report.Gadget Viewer.getGadget(string bookmark)

Returns an instance of the gadget referenced by a bookmark.

Parameters **bookmark**
String. The bookmark name.

Returns actuate.report.Gadget object.

Example To get the gadget with the bookmark FirstGadget and store it in the variable myGadget, use code similar to the following:

```
var myGadget = myViewer.getGadget("FirstGadget");
```

actuate.Viewer.getHeight

Syntax string Viewer.getHeight( )

Returns the height value of the viewer.

Returns String.

Example This example decreases the viewer's height by 10:

```
var height = myViewer.getHeight( );
myViewer.setHeight(height - 10);
```

actuate.Viewer.getHelpBase

Syntax string Viewer.getHelpBase( )

Returns the URL of the help base. The help base is the base URL for the product help documentation.

Returns String. The base URL of the help documentation.

Example This example displays the help base URL in an alert box:

```
alert("The help base URL is " + myViewer.getHelpBase( ))
```

actuate.Viewer.getLabel

Syntax actuate.report.Label Viewer.getLabel(string bookmark)

Returns an instance of the label referenced by a bookmark.

Parameters **bookmark**
String. The bookmark name.

Returns actuate.report.Label object.

Example To get the label with the bookmark FirstLabel and store it in the variable myLabel, use code similar to the following:

```
var myLabel = myViewer.getLabel("FirstLabel");
```

actuate.Viewer.getReportletBookmark

Syntax string Viewer.getReportletBookmark( )

Returns the bookmark of the current report page or element.

Returns String. Bookmark.

Example This example displays the bookmark of the current report page in an alert box:

```
alert ("Report bookmark is " + myViewer.getReportletBookmark( ));
```

actuate.Viewer.getReportName

Syntax string Viewer.getReportName( )

Returns the name of the report file, either a report design file or report document file, that is currently displayed in this Viewer.

Returns String.

Example This example displays the currently displayed report file name in an alert box:

```
alert ("Currently displaying " + myViewer.getReportName( ));
```

actuate.Viewer.getTable

Syntax actuate.report.Table Viewer.getTable(string bookmark)

Returns an instance of the table referenced by a bookmark.

Parameters **bookmark**
String. The bookmark name.

Returns actuate.report.Table object.

Example To get the table with the bookmark FirstTable and store it in the variable myTable, use code similar to the following:

```
var myTable = myViewer.getTable("FirstTable");
```

actuate.Viewer.getText

Syntax actuate.report.Text Viewer.getText(string bookmark)

Returns an instance of the Text object referenced by a bookmark.

Parameters **bookmark**
String. The bookmark name.

Returns actuate.report.Text object.

Example To get the Text object with the bookmark Title and store it in the variable myText, use code similar to the following:

```
var myText = myViewer.getText("Title");
```

actuate.Viewer.getTotalPageCount

Syntax integer Viewer.getTotalPageCount( )

Returns the total number of pages in the report being viewed.

Returns Integer.

Example This function is useful to move to the last page of a document. To go to the last page in a document, use the following code:

```
viewer.gotoPage(viewer.getTotalPageCount( ));
```

actuate.Viewer.getUIConfig

Syntax actuate.viewer.UIConfig Viewer.getUIConfig( )

Returns the current UI configuration.

Returns actuate.viewer.UIConfig object. This function returns null when no UIConfig object is set.

Example To retrieve and store the content pane from the viewer, use the following code:

```
var contentpane = viewer.getUIConfig( ).getContentPane( );
```

actuate.Viewer.getUIOptions

Syntax actuate.viewer.UIOptions Viewer.getUIOptions( )

Returns the UIOptions set in this viewer object.

Returns actuate.viewer.UIOptions object. This function returns null when no UIOptions object is set.

Example To retrieve and store the uiOptions for the viewer, use the following code:

```
var options = myViewer.getUIOptions( );
```

actuate.Viewer.getViewer

Syntax actuate.Viewer Viewer.getViewer(string bookmark)

actuate.Viewer Viewer.getViewer(object elementID)

Returns a viewer object containing the report element that is associated with a bookmark or contained in an HTML element.

Parameters **bookmark**
String. The bookmark of the report element to view.

elementID
Object. An HTML element that contains the viewer.

Returns actuate.Viewer object or null if the viewer is not found.

Example This example uses getViewer( ) to retrieve a report element and return the bookmark of the chart in that report:

```
function chartBookmark(myReport){
   var bviewer = myReport.getViewer("Chart");
   var bpagecontents = bviewer.getCurrentPageContent( );
   return bpagecontents.getChartByBookmark("ChartBookmark");
}
```

actuate.Viewer.getWidth

Syntax string Viewer.getWidth( )

Returns the width value of the viewer.

Returns String.

Example This example decreases the viewer's width by 10:

```
var width = myViewer.getWidth( );
myViewer.setWidth(width - 10);
```

actuate.Viewer.gotoBookmark

Syntax void Viewer.gotoBookmark(string bookmark)

Goes to the page position by the specified bookmark. The viewer displays to the first page when the bookmark is not found.

Parameters **bookmark**
String. The bookmark of a report element.

Example To move the viewer to the page position specified by the value of the 'bookmark' parameter, use this code:

```
viewer.gotoBookmark(document.getElementById('bookmark').value);
```

actuate.Viewer.gotoPage

Syntax void Viewer.gotoPage(integer pageNumber)

Goes to the specified page. The viewer throws an exception when the page is not found.

Parameters **pageNumber**
Integer. A page number in the report.

Example To go to the first page of a report, use the following code:

```
viewer.gotoPage(1);
```

actuate.Viewer.isInteractive

Syntax boolean Viewer.isInteractive( )

Returns the interactive viewing status of the viewer. Enables or disables the interactive viewing features with actuate.Viewer.enableIV( ).

Returns Boolean. True when interactive viewing features are enabled.

Example This example displays an alert box with the interactive status of the viewer:

```
alert("Interactivity of this viewer is set to " +
   myViewer.isInteractive( ));
```

actuate.Viewer.saveReportDesign

Syntax void Viewer.saveReportDesign(string filename, function callback)

Saves the current viewer content as a report design. The viewer must enable interactive viewing with enableIV( ) prior to saving a report design.

Parameters **filename**
String. Sets the name of the saved file. The current file name is used if null. The file name must be a path relative to the viewer's repository.

callback
Function. Optional. The function to execute after the asynchronous call processing is done. The callback takes the current actuate.Viewer object as an input parameter.

Example To save the content of the viewer as the report design called NewDesign, use the following code:

```
myViewer.saveReportDesign("NewDesign");
```

actuate.Viewer.saveReportDocument

Syntax void Viewer.saveReportDocument(string filename, function callback)

Saves the current viewer content as a report document. The viewer must enable interactive viewing with enableIV( ) prior to saving a report design.

Parameters **filename**
String. Sets the name of the saved file. The current file name is used if null. The file name must be a path relative to the viewer's repository.

callback
Function. Optional. The function to execute after the asynchronous call processing is done. The callback takes the current actuate.Viewer object as an input parameter.

Example To save the content of the viewer as the report document called NewDocument, use the following code:

```
myViewer.saveReportDocument("NewDocument");
```

actuate.Viewer.setContentMarg

Syntax void Viewer.setContentMargin(string[ ] margin)

void Viewer.setContentMargin(int margin)

Sets the viewer content margin.

Parameters **margin**
Array of strings or integer. Each member of the array is the margin for the top, left, right, and bottom internal margins for the viewer. An integer sets all margins to that value.

Example To set the internal margin of the viewer to a 10-pixel buffer, use the following code:

```
myViewer.setContentMargin(10);
```

actuate.Viewer.setFocus

Syntax void setFocus(boolean focus)

Sets the focus for the viewer.

Parameters **focus**
Boolean. The viewer's context menu is in focus when this parameter is set to true.

Example This example blurs the context menu for the viewer:

```
viewer.setFocus(false);
```

actuate.Viewer.setHeight

Syntax void Viewer.setHeight(integer height)

Sets the viewer height.

Parameters **height**
Integer. The height in pixels.

Example To set the height of the viewer to 600 pixels, use the following code:

```
viewer.setHeight(600);
```

actuate.Viewer.setHelpBase

Syntax void Viewer.setHelpBase(string helpBase)

Sets the URL of the help base. The help base is the base URL for the product help documentation.

Parameters **helpBase**
String. The URL where the help documentation is located.

Example This example sets the help base URL to http://www.actuate.com/documentation/R11:

```
myViewer.setHelpBase("http://www.actuate.com/documentation/R11");
myViewer.submit( );
```

actuate.Viewer.setParameters

Syntax void Viewer.setParameters(string[ ] params)

Sets parameters for executing report using literal string pairs.

Parameters **params**
Array of strings. Each string in the array is constructed of name:"value" pairs. Use a literal list, such as {param1:"value1", param2:"value2", … }.

Example To set the value of a parameter, city, to the value, New York, use the following object literal:

```
viewer.setParameters({ city:"New York"});
```

actuate.Viewer.setParameterValues

Syntax void Viewer.setParameterValues(actuate.parameter.ParameterValue[ ] parameters)

Sets parameter values for executing a report using ParameterValue objects.

Parameters **parameters**
Array of actuate.parameter.ParameterValue objects. An array of this kind is returned by actuate.Parameter.downloadParameterValues( ) and is the recommended function for creating the parameters input.

Example To set the parameter values for a report to the parameters in the pvs array, use this code:

```
viewer.setParameterValues(pvs);
```

actuate.Viewer.setReportletBookmark

Syntax void Viewer.setReportletBookmark(string bookmark)

Sets the bookmark for the Reportlet rendered.

Parameters **bookmark**
String. The bookmark ID used to render the Reportlet. Viewer requires a bookmark to render a Reportlet. Viewer does not support automatically generated generic bookmarks from a BIRT report.

Example To open the Top 5 Customers Reportlet of the Customer Dashboard, set the Reportlet bookmark by name and then call viewer.submit, as shown in the following example:

```
viewer.setReportName("/Public/BIRT and BIRT Studio Examples
   /Customer Dashboard.rptdocument");
viewer.setReportletBookmark("Top 5 Customers");
viewer.submit( );
```

actuate.Viewer.setReportName

Syntax void Viewer.setReportName(string reportFile)

Sets the report file, either a report design or report document, to render in this Viewer.

Parameters **reportFile**
String. The report file path for a report design file or report document file.

Example To open the Top 5 Sales Performers report, set the report by name and then call submit( ), as shown in the following example:

```
viewer.setReportName("/Public/BIRT and BIRT Studio Examples/Top 5
   Sales Performers.rptdesign");
viewer.submit( );
```

actuate.Viewer.setService

Syntax void Viewer.setService(string iPortalURL, actuate.RequestOptions requestOptions)

Sets the target service URL to which this Viewer links. When the service URL is not set, this Viewer links to the default service URL, which is set on the actuate object.

Parameters **iPortalURL**
String. The target Actuate web application URL, either a Java Component or Information Console.

requestOptions
actuate.RequestOptions object. Optional. requestOptions defines URL parameters to send with the authentication request, such as the iHub URL, Encyclopedia volume, or repository type. The URL can also include custom parameters.

Example This example sets the URL for the Actuate iPortal web application service:

```
myViewer.setService("http://localhost:8700/iportal",
   myRequestOptions);
```

actuate.Viewer.setSize

Syntax void Viewer.setSize(integer width, integer height)

Resizes the viewer's width and height.

Parameters **width**
Integer. The new width is specified in pixels.

height
Integer. The new height is specified in pixels.

Example To set the viewer's size to 300 pixels by 300 pixels, use code similar to the following:

```
myViewer.setSize(300, 300);
```

actuate.Viewer.setSupportSVG

Syntax void Viewer.setSupportSVG(boolean usvgFlag)

Controls Scalable Vector Graphics support for the viewer.

Parameters **svgFlag**
Boolean. True enables Scalable Vector Graphic support.

Example To disable Scalable Vector Graphic support for the myViewer viewer, use code similar to the following:

```
myViewer.setSupportSVG(false);
```

actuate.Viewer.setUIOptions

Syntax void Viewer.setUIOptions(actuate.viewer.UIOptions options)

Sets the UI options for the viewer using an actuate.viewer.UIOptions object.

Parameters **options**
actuate.viewer.UIOptions object. Enables or disables various controls and features.

Example To hide the toolbar for the viewer, use the following code:

```
uioptions.enableToolBar(false);
viewer.setUIOptions(uioptions);
viewer.submit( );
```

actuate.Viewer.setViewingMode

Syntax void Viewer.setViewingMode(string viewer)

Sets the dashboard viewing mode.

Parameters **viewer**
actuate.Constant.ViewingMode constant. Legal values are NON_DASHBOARD, DASHBOARD_NORMAL, and DASHBOARD_MAX.

Example To display content without dashboard features, use the following code:

```
viewer.setViewingMode(actuate.Constant.ViewingMode.NON_DASHBOARD);
```

actuate.Viewer.setWidth

Syntax void Viewer.setWidth(string width)

Sets the viewer width.

Parameters **width**
String.

Example To set the width of the viewer to 800 pixels, use the following code:

```
viewer.setWidth(800);
```

actuate.Viewer.showDownloadReportDialog

Syntax void Viewer.showDownloadReportDialog( )

Displays the export report dialog window.

Example Use this code to display the report dialog window:

```
myViewer.showDownloadReportDialog( );
```

actuate.Viewer.showDownloadResultSetDialog

Syntax void Viewer.showDownloadResultSetDialog( )

Displays the export data dialog window.

Example Use this code to display the result set download dialog window:

```
viewer.showDownloadResultSetDialog( );
```

actuate.Viewer.showFacebookCommentPanel

Syntax void Viewer.showFacebookCommentPanel( )

Displays the Facebook comments panel.

Example Use this code to display the Facebook comments panel:

```
viewer.showFacebookCommentPanel( );
```

actuate.Viewer.showParameterPanel

Syntax void Viewer.showParameterPanel( )

Displays the parameter panel.

Example Use this code to display the parameter panel:

```
viewer.showParameterPanel( );
```

actuate.Viewer.showPrintDialog

Syntax void Viewer.showPrintDialog( )

Displays the print dialog window.

Example Use this code to display the print dialog window:

```
viewer.showPrintDialog( );
```

actuate.Viewer.showTocPanel

Syntax void Viewer.showTocPanel( )

Displays the table of contents panel.

Example Use this code to display the table of contents panel:

```
viewer.showTocPanel( );
```

actuate.Viewer.submit

Syntax void Viewer.submit(function callback, boolean rerun)

Updates and reloads the viewer after submitting requests for the viewer. The submit( ) function triggers an AJAX request for all asynchronous operations. When the server finishes the processing, it returns a response and the results are rendered on the page in the viewer container. Calling submit( ) when a previous submit( ) is pending throws an exception.

Parameters **callback**
Function. Optional. The function to execute after the asynchronous call processing is done.

rerun
Boolean. Optional. Indicates whether to re-run the report design when refreshing. Default to true.

Example To open the Top 5 Sales Performers report, set the report by name and then call submit( ), as shown in the following example:

```
viewer.setReportName("/Public/BIRT and BIRT Studio Examples/Top 5
   Sales Performers.rptdesign");
viewer.submit( );
```

Class actuate.viewer.BrowserPanel

Description A container for a browser content panel in a viewer. This class defines the default scroll bars for a content panel.

Constructor

Syntax actuate.Viewer.BrowserPanel( )

Constructs a new BrowserPanel object for the parent viewer. The browser panel has vertical and horizontal scroll bars for navigation.

Class actuate.viewer.EventConstants

Description Defines the event constants supported by this API. Table 4-48 lists the viewer event constants.

Table 4-48 Actuate JavaScript API viewer event constants

Event	Description
ON_CONTENT_CHANGED	Calls a registered event handler when the report content is reloaded. The event handler must take the viewer instance that fired the event as an input argument.
ON_CONTENT_SELECTED	Calls a registered event handler when the relevant part of the report content is selected. Supported selected contents are: ■ Column ■ Table ■ Data ■ Label ■ Text When the content is selected, the corresponding object is passed into user's event handler function. For example, if the table area is selected in a viewer, actuate.Viewer.Table is passed into the event handler. The event handler must take the viewer instance that fired the event and an instance of actuate.viewer.SelectedContent as input arguments.
ON_DIALOG_OK	This event fires when the user clicks the OK button in a dialog. The event handler must take the viewer object that fired the event and a dialog.AdvancedFilterDialog object as input parameters.
ON_EXCEPTION	An exception event is broadcast when an error occurs. The event handler must take the viewer instance that fired the event and an instance of actuate.viewer.Exception as input arguments.
ON_SESSION_TIMEOUT	Calls a registered event handler when the session expires. The event handler must take the viewer instance that fired the event as an input argument.

Class actuate.viewer.PageContent

Description A container for the content of a report document file. actuate.Viewer.PageContent contains a comprehensive list of report elements, such as tables, charts, labels, and data items.

Constructor

The PageContent object is constructed by actuate.viewer.getCurrentPageContent( ).

Function summary

Table 4-49 lists actuate.viewer.PageContent functions.

Table 4-49 actuate.viewer.PageContent functions

Function	Description
getChartByBookmark( )	Returns a chart element specified by the given bookmark
getDataItemByBookmark( )	Returns a data element specified by the given bookmark
getFlashObjectByBookmark( )	Returns a Flash object specified by the given bookmark
getGadgetByBookmark( )	Returns a Flash gadget specified by the given bookmark
getLabelByBookmark( )	Returns a label element specified by the given bookmark
getTableByBookmark( )	Returns a table element specified by the given bookmark
getTextByBookmark( )	Returns a text element specified by the given bookmark
getViewerId( )	Returns the viewer ID

actuate.viewer.PageContent.getChartByBookmark

Syntax actuate.report.Chart PageContent.getChartByBookmark(string bookmark)

Returns the chart element specified by the given bookmark.

Parameters **bookmark**
String. A bookmark to identify a chart element. When the bookmark value is not given, this function returns the first chart element found in the report content.

Returns actuate.report.Chart object.

Example This example retrieves the Chart object and changes the chart title:

```
this.onclick = function(event){
   var bviewer = this.getViewer( );
   var bpagecontents = bviewer.getCurrentPageContent( );
   var bchart = bpagecontents.getChartByBookmark("ChartBookmark");
   bchart.setChartTitle("Orders By Country (Classic Cars)");
   bchart.submit( );
}
```

actuate.viewer.PageContent.getDataItemByBookmark

Syntax actuate.report.DataItem PageContent.getDataItemByBookmark(string bookmark)

Returns the data element specified by the given bookmark.

Parameters **bookmark**
String. A bookmark to identify a data element. When the bookmark value is not given, the first data element found in the report content is returned.

Returns actuate.report.DataItem object.

Example Use this function to access the bookmarks for specific elements in the page content. For example, to access the data element "myDataItem" on the page loaded in the myViewer viewer object, use the following code:

```
var element = myViewer.getCurrentPageContent( ).
   getDataItemByBookmark("myDataItem");
```

actuate.viewer.PageContent .getFlashObjectByBookmark

Syntax actuate.report.FlashObject PageContent.getFlashObjectByBookmark(string bookmark)

Returns the Flash object specified by the given bookmark.

Parameters **bookmark**
String. A bookmark to identify a Flash object. When the bookmark value is not given, the first data element found in the report content is returned.

Returns actuate.report.FlashObject object.

Example Use this function to access the bookmarks for specific elements in the page content. For example, to access the Flash object "myFlashObj" on the page loaded in the myViewer viewer object, use the following code:

```
var element = myViewer.getCurrentPageContent( ).
   getFlashObjectByBookmark("myFlashObj");
```

actuate.viewer.PageContent.getGadgetByBookmark

Syntax actuate.report.Gadget PageContent.getGadgetByBookmark(string bookmark)

Returns the gadget element specified by the given bookmark.

Parameters **bookmark**
String. A bookmark to identify a gadget element. When the bookmark value is not given, the first data element found in the report content is returned.

Returns actuate.report.Gadget object.

Example Use this function to access the bookmarks for specific elements in the page content. For example, to access the gadget "myGadget" on the page loaded in the myViewer viewer object, use the following code:

```
var element = myViewer.getCurrentPageContent( ).
   getGadgetByBookmark("myGadget");
```

actuate.viewer.PageContent.getLabelByBookmark

Syntax actuate.report.Label PageContent.getLabelByBookmark(string bookmark)

Returns the label element specified by the given bookmark.

Parameters **bookmark**
String. A bookmark to identify a label element. When the bookmark value is not given, the first label element found in the report content is returned.

Returns actuate.report.Label object.

Example Use this function to access the bookmarks for specific elements in the page content. For example, to access the label "LabelOne" on the page loaded in the myViewer viewer object, use the following code:

```
var element = myViewer.getCurrentPageContent( ).
   getLabelByBookmark("LabelOne");
```

actuate.viewer.PageContent.getTableByBookmark

Syntax actuate.report.Table PageContent.getTableByBookmark(string bookmark)

Returns the table element specified by the given bookmark.

Parameters **bookmark**
String. A bookmark to identify a table element. When the bookmark value is not given, the first table element found in the report content is returned.

Returns actuate.report.Table object.

Example Use this function to access the bookmarks for specific elements in the page content. For example, to access the table mytable on the page loaded in the myViewer viewer object, use the following code:

```
var element = myViewer.getCurrentPageContent( )
   .getTableByBookmark("mytable");
```

actuate.viewer.PageContent.getTextByBookmark

Syntax actuate.report.TextItem PageContent.getTextByBookmark(string bookmark)

Returns the text element specified by the given bookmark.

Parameters **bookmark**
String. A bookmark to identify a text element. If the bookmark value is not given, the first text element found in the report content is returned.

Returns actuate.report.TextItem object.

Example Use this function to access the bookmarks for specific elements in the page content. For example, to access the text item "myTextItem" on the page loaded in the myViewer viewer object, use the following code:

```
var element = myViewer.getCurrentPageContent( ).
   getTextByBookmark("myTextItem");
```

actuate.viewer.PageContent.getViewerId

Syntax string PageContent.getViewerId( )

Returns the viewer ID.

Returns String. The viewer ID.

Example This example displays the viewer ID in an alert box:

```
alert("The Viewer ID is " + myViewer.getViewerId( ));
```

Class actuate.viewer.ParameterValue

Description The ParameterValue class is a JavaScript version of the com.actuate.schemas.ParameterValue class.

Constructor

Syntax actuate.parameter.ParameterValue( )

Constructs a new ParameterValue object.

Function summary

Table 4-50 lists the actuate.viewer.ParameterValue functions.

Table 4-50 actuate.viewer.ParameterValue functions

Function	Description
getName( )	Returns the name value
getValue( )	Returns the value value
getValueIsNull( )	Returns the valueIsNull value
setName( )	Sets the name value
setValue( )	Sets the value value
setValueIsNull( )	Sets the valueIsNull value

actuate.viewer.ParameterValue.getName

Syntax string ParameterValue.getName( )

Returns the name value.

Returns String. The name value.

Example To store the name of a viewer.ParameterValue object in a variable called vPVname, use code similar to the following:

```
var vPVname = myParamValue.getName( );
```

actuate.viewer.ParameterValue.getValue

Syntax object ParameterValue.getValue( )

Returns the value value.

Returns Object. The value value, a string or array of strings.

Example To store a ParameterValue's value in vPVvalue, use the following code:

```
var vPVvalue = myParamValue.getValue( );
```

actuate.viewer.ParameterValue.getValueIsNull

Syntax boolean ParameterValue.getValueIsNull( )

Returns the valueIsNull value.

Returns Boolean. The valueIsNull value.

Example This example displays an alert with the valueIsNull of the ParameterValue object:

```
alert("Value is null: " + myParamValue.getValueIsNull( ));
```

actuate.viewer.ParameterValue.setColumnName

Syntax void ParameterValue.setColumnName(string columnName)

Sets the columnName value.

Parameters **columnName**
String. The column name.

Example To set the column name to "Motorcycles", use code similar to the following:

```
myParamValue.setColumnName("Motorcycles");
```

actuate.viewer.ParameterValue.setValue

Syntax void ParameterValue.setValue(object value)

Sets the value. A value can be a string or an array of strings.

Parameters **value**
Object. The value for this ParameterValue object, a string or an array of strings.

Example To set the value for a ParameterValue to myValues, use the following code:

```
var myValues = myParamValue.setValue(myValues);
```

actuate.viewer.ParameterValue.setValueIsNull

Syntax void ParameterValue.setValueIsNull(boolean valueIsNull)

Sets the valueIsNull value.

Parameters **valueIsNull**
Boolean. The valueIsNull value.

Example To set a ParameterValue's setValueIsNull to true, use the following code:

```
myParamValue.setValueIsNull(true);
```

Class actuate.viewer.RenderOptions

Description The RenderOptions class specifies render options for the actuate.Viewer.downloadReport( ) function. Currently, the only supported option is multisheet.

Constructor

Syntax actuate.Viewer.RenderOptions( )

Constructs a new RenderOptions object for the parent viewer.

Function summary

Table 4-51 lists actuate.viewer.RenderOptions functions.

Table 4-51 actuate.viewer.RenderOptions functions

Function	Description
getOptions( )	Returns whether mouse scrolling is enabled
setOption( )	Returns whether mouse panning is enabled

actuate.viewer.RenderOptions.getOptions

Syntax Object[ ] RenderOptions.getOptions( )

Returns the render options map.

Returns Array, arranged in string and object pairs corresponding to option names and option values.

Example This example displays an alert box with the options status of render options:

```
alert("Rendering Options: " + options.getOptions( ));
```

actuate.viewer.RenderOptions.setOption

Syntax void RenderOptions.setOption(string option, boolean value)

Specifies a render option and its setting.

Parameters **option**
String. The permitted value is actuate.viewer.RenderOptions.IS_MULTISHEET, which is used for xls format download only.

value
Boolean. Enabled value for IS_MULTISHEET. True indicates that the xls format file has multiple worksheets.

Example To disable multisheet for the options object, use code similar to the following:

```
options.setOption(actuate.viewer.RenderOptions.IS_MULTISHEET,
   false);
```

Class actuate.viewer.ScrollPanel

Description A container for a scrolling content panel in a viewer, which includes the scroll panel control, as shown in Figure 4-1.

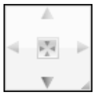


Figure 4-1 Scroll panel control

A ScrollPanel object enhances the viewer with scroll controls, such as mouse wheel scrolling.

Constructor

Syntax actuate.Viewer.ScrollPanel( )

Constructs a new ScrollPanel object for the parent viewer enabled scroll controls.

Function summary

Table 4-52 lists actuate.viewer.ScrollPanel functions.

Table 4-52 actuate.viewer.ScrollPanel functions

Function	Description
getMouseScrollingEnabled( )	Returns whether mouse scrolling is enabled
getPanInOutEnabled( )	Returns whether mouse panning is enabled
getScrollControlEnabled( )	Returns whether scrolling is enabled
setMouseScrollingEnabled( )	Enables mouse scrolling
setPanInOutEnabled( )	Enables panning
setScrollControlEnabled( )	Enables scrolling

actuate.viewer.ScrollPanel.getMouseScrollingEnabled

Syntax boolean ScrollPanel.getMouseScrollingEnabled( )

Returns true when mouse scrolling is enabled.

Returns Boolean.

Example This example displays an alert with the mouse scrolling status of a scroll panel:

```
alert("Mouse scrolling enabled: " +
   sPanel.getMouseScrollingEnabled( ));
```

actuate.viewer.ScrollPanel.getPanInOutEnabled

Syntax boolean ScrollPanel.getPanInOutEnabled( )

Returns true when panning in and out is enabled.

Returns Boolean.

Example This example displays an alert with the panning in and out status of a scroll panel:

```
alert("Panning enabled: " + scrollPanel.getPanInOutEnabled( ));
```

actuate.viewer.ScrollPanel.getScrollControlEnabled

Syntax boolean ScrollPanel.getScrollControlEnabled( )

Returns true when scrolling is enabled.

Returns Boolean.

Example This example displays an alert box with the scrolling status of a scroll panel:

```
alert("Scrolling enabled: " + sPanel.getScrollControlEnabled( ));
```

actuate.viewer.ScrollPanel.setMouseScrollingEnabled

Syntax void ScrollPanel.setMouseScrollingEnabled(boolean enabled)

Enables mouse scrolling for this scroll panel.

Parameters **enabled**
Boolean.

Example To disable mouse scrolling for sPanel, use code similar to the following:

```
sPanel.setMouseScrollingEnabled(false);
```

actuate.viewer.ScrollPanel.setPanInOutEnabled

Syntax void ScrollPanel.setPanInOutEnabled(boolean enabled)

Enables panning in and out for this scroll panel.

Parameters **enabled**
Boolean.

Example To disable panning for the sPanel object, use code similar to the following:

```
sPanel.setPanInOutEnabled(false);
```

actuate.viewer.ScrollPanel.setScrollControlEnabled

Syntax void ScrollPanel.setScrollControlEnabled(boolean enabled)

Enables scrolling for this scroll panel.

Parameters **enabled**
Boolean.

Example To disable scrolling for sPanel, use code similar to the following:

```
sPanel.setScrollControlEnabled(false);
```

Class actuate.viewer.SelectedContent

Description A container for content selected in the viewer. SelectedContent provides an object to pass to a handler when the user-defined ON_CONTENT_SELECTED event occurs. This object contains an instance of the element selected in the viewer.

Constructor

The SelectedContent object is constructed when an ON_CONTENT_SELECTED event occurs.

Function summary

Table 4-53 lists actuate.viewer.SelectedContent functions.

Table 4-53 actuate.viewer.SelectedContent functions

Function	Description
getColumnIndex( )	Returns the currently selected table column index number
getSelectedElement( )	Returns a copy of the currently selected element

actuate.viewer.SelectedContent.getColumnIndex

Syntax integer SelectedContent.getColumnIndex( )

Returns the numerical index for the currently selected column. Returns null when the user selects a non-table element.

Returns Integer.

Example To retrieve the index of a column selected, use the following code:

```
var index = selected.getColumnIndex( );
```

actuate.viewer.SelectedContent.getSelectedElement

Syntax object SelectedContent.getSelectedElement( )

Returns an instance of the currently selected element. The instance can be one of the following objects:

- actuate.report.Chart
- actuate.report.DataItem
- actuate.report.Label

- actuate.report.Table
- actuate.report.TextItem

To determine the object type, use the Object.getType( ) function. The type strings for the above objects are "Chart", "Data", "Label", "Table", or "Text", respectively.

Returns Object. An instance of the currently selected element.

Example To retrieve and store a label bookmark if a selected element is a label, use the following code:

```
var selected = selected.getColumnIndex( );
if (selected.getType( ) == "Label"){
   var bmark = Object.getBookmark( );
}
```

Class actuate.viewer.UIConfig

Description The UIConfig class specifies feature availability for the viewer.

Constructor

Syntax void actuate.viewer.UIConfig( )

Generates a new UIConfig object to manage the content panel for the viewer. By default, the content panel is an actuate.viewer.ScrollPanel object with ScrollControl, PanInOut, and MouseScrolling enabled.

Function summary

Table 4-54 lists actuate.viewer.UIConfig functions.

Table 4-54 actuate.viewer.UIConfig functions

Function	Description
getContentPanel( )	Returns the content panel configuration
getShowToc( )	Gets the showToc flag
setContentPanel( )	Sets the content panel configuration
setShowToc( )	Sets the showToc flag

actuate.viewer.UIConfig.getContentPanel

Syntax object UIConfig.getContentPanel( )

Returns the content panel object.

Returns Object. Valid objects are actuate.viewer.BrowserPanel, actuate.viewer.ScrollPanel, and null. A null value indicates a content panel configured with the browser scroll bar enabled.

Example To retrieve and store the content panel from the viewer, use the following code:

```
var contentpanel = viewer.getUIConfig( ).getContentPanel( );
```

actuate.viewer.UIConfig.getShowToc

Syntax boolean UIConfig.getShowToc( )

Returns the showToc flag.

Returns Boolean.

Example To determine if the showtoc flag is set to true, use the following code:

```
if (!viewer.getUIConfig( ).getShowToc( )){ ...}
```

actuate.viewer.UIConfig.setContentPanel

Syntax void UIConfig.setContentPanel(objectcontentPanel)

Sets the content panel for the viewer.

Parameters **contentPanel**
Object. Valid objects are actuate.viewer.BrowserPanel, actuate.viewer.ScrollPanel, and null. A null value sets a content panel configured with the browser scroll bar enabled.

Example To set the content panel to BrowserPanel if it is null, use the following code:

```
var contentpanel = viewer.getUIConfig( ).getContentPanel( );
if (contentpanel == null){
   var newconfig = viewer.getUIConfig( );
   newconfig.setContentPanel(new actuate.viewer.BrowserPanel( ));
   viewer.setUIConfig(newconfig);
}
```

actuate.viewer.UIConfig.setShowToc

Syntax void UIConfig.setShowToc(boolean showToc)

Sets the showToc flag.

Parameters **showToc**
Boolean.

Example To hide the Toc in the UI, use the following code:

```
var newconfig = viewer.getUIConfig( );
newconfig.setShowToc(false);
viewer.setUIConfig(newconfig);
```

Class actuate.viewer.UIOptions

Description The UIOptions class specifies feature availability for the viewer object.

Constructor

Syntax void actuate.viewer.UIOptions( )

Generates a new UIOptions object to manage the features of the viewer.

Function summary

Table 4-55 lists actuate.viewer.UIOptions functions.

Table 4-55 actuate.viewer.UIOptions functions

Function	Description
enableAdvancedSort( )	Enables the advanced sort feature
enableAggregation( )	Enables the aggregation feature
enableCalculatedColumn( )	Enables the calculated column feature
enableChartProperty( )	Enables the chart properties feature
enableChartSubType( )	Enables the chart subtype selection
enableCollapseExpand( )	Enables the collapse/expand feature
enableColumnEdit( )	Enables the column editing feature
enableColumnResize( )	Enables the column resizing feature
enableContentMargin( )	Enables the content margin feature
enableDataAnalyzer( )	Enables the Data Analyzer feature
enableDataExtraction( )	Enables the data extraction feature
enableEditReport( )	Enables the report editing feature
enableExportReport( )	Enables the export report feature
enableFilter( )	Enables the filter feature
enableFacebookComments( )	Enables the Facebook comments feature.
enableFlashGadgetType( )	Enables the Flash gadget type change feature
enableFormat( )	Enables the format editing feature
enableGroupEdit( )	Enables the group editing feature
enableHideShowItems( )	Enables the hide/show item feature
enableHighlight( )	Enables the highlight feature

(continues)

Table 4-55 actuate.viewer.UIOptions functions (continued)

Function	Description
enableHoverHighlight( )	Enables the hover highlight feature
enableLaunchViewer( )	Enables the launch viewer feature
enableLinkToThisPage( )	Enables the "link to this page" feature
enableMainMenu( )	Enables the main menu feature
enableMoveColumn( )	Enables column moving
enablePageBreak( )	Enables the page break editing feature
enablePageNavigation( )	Enables the page navigation feature
enableParameterPage( )	Enables the parameter page feature
enablePrint( )	Enables the print feature
enableReorderColumns( )	Enables the column reordering
enableRowResize( )	Enables row resizing
enableSaveDesign( )	Enables the report design save feature
enableSaveDocument( )	Enables the report document save feature
enableShowToolTip( )	Enables the show tooltip feature
enableSort( )	Enables the sort feature
enableSuppressDuplicate( )	Enables the duplication suppression feature
enableSwitchView( )	Enables the switch view feature
enableTextEdit( )	Enables the text editing feature
enableTOC( )	Enables the table of contents feature
enableToolBar( )	Enables the toolbar feature
enableToolbarContextMenu( )	Enables the toolbar context menu feature
enableToolbarHelp( )	Enables the toolbar help feature
enableTopBottomNFilter( )	Enables the top N and bottom N filter feature
enableUndoRedo( )	Enables the undo and redo feature
getFeatureMap( )	Returns a list of enabled and disabled features

actuate.viewer.UIOptions.enableAdvancedSort

Syntax void UIOptions.enableAdvancedSort(boolean enabled)

Enables or disables the advanced sort feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the advanced sort feature, use code similar to the following:

```
viewerOpts.enableAdvancedSort(false);
```

actuate.viewer.UIOptions.enableAggregation

Syntax void UIOptions.enableAggregation(boolean enabled)

Enables or disables the aggregation feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the aggregation feature, use code similar to the following:

```
viewerOpts.enableAggregation(false);
```

actuate.viewer.UIOptions.enableCalculatedColumn

Syntax void UIOptions.enableCalculatedColumn(boolean enabled)

Enables or disables the calculated column feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the calculated column feature, use code similar to the following:

```
viewerOpts.enableCalculatedColumn(false);
```

actuate.viewer.UIOptions.enableChartProperty

Syntax void UIOptions.enableChartProperty(boolean enabled)

Enables or disables the chart properties feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the chart properties feature, use code similar to the following:

```
viewerOpts.enableChartProperty(false);
```

actuate.viewer.UIOptions.enableChartSubType

Syntax void UIOptions.enableChartSubType(boolean enabled)

Enables or disables the chart subtype selection feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the chart subtype selection feature, use code similar to the following:

```
viewerOpts.enableChartSubType(false);
```

actuate.viewer.UIOptions.enableCollapseExpand

Syntax void UIOptions.enableCollapseExpand(boolean enabled)

Enables or disables the collapse/expand feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the collapse/expand feature, use code similar to the following:

```
viewerOpts.enableCollapseExpand(false);
```

actuate.viewer.UIOptions.enableColumnEdit

Syntax void UIOptions.enableColumnEdit(boolean enabled)

Enables or disables the column editing feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the column editing feature, use code similar to the following:

```
viewerOpts.enableColumnEdit(false);
```

actuate.viewer.UIOptions.enableColumnResize

Syntax void UIOptions.enableColumnResize(boolean enabled)

Enables or disables the column resizing feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the column resizing feature, use code similar to the following:

```
viewerOpts.enableColumnResize(false);
```

actuate.viewer.UIOptions.enableContentMargin

Syntax void UIOptions.enableContentMargin(boolean enabled)

Enables or disables the content margin feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the content margin feature, use code similar to the following:

```
viewerOpts.enableContentMargin(false);
```

actuate.viewer.UIOptions.enableDataAnalyzer

Syntax void UIOptions.enableDataAnalyzer(boolean enabled)

Enables or disables the Data Analyzer feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the Data Analyzer feature, use code similar to the following:

```
viewerOpts.enableDataAnalyzer(false);
```

actuate.viewer.UIOptions.enableDataExtraction

Syntax void UIOptions.enableDataExtraction(boolean enabled)

Enables or disables the data extraction feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the data extraction feature, use code similar to the following:

```
viewerOpts.enableDataExtraction(false);
```

actuate.viewer.UIOptions.enableEditReport

Syntax void UIOptions.enableEditReport(boolean enabled)

Enables or disables the report editing feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the report editing feature, use code similar to the following:

```
viewerOpts.enableEditReport(false);
```

actuate.viewer.UIOptions.enableExportReport

Syntax void UIOptions.enableExportReport(boolean enabled)

Enables or disables the export report feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the export report feature, use code similar to the following:

```
viewerOpts.enableExportReport(false);
```

actuate.viewer.UIOptions.enableFilter

Syntax void UIOptions.enableFilter(boolean enabled)

Enables or disables the filter feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the filter feature, use code similar to the following:

```
viewerOpts.enableFilter(false);
```

actuate.viewer.UIOptions.enableFacebookComments

Syntax void UIOptions.enableFacebookComments(boolean enabled)

Enables or disables the Facebook comments feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the Facebook comments feature, use code similar to the following:

```
viewerOpts.enableFacebookComments(false);
```

actuate.viewer.UIOptions.enableFlashGadgetType

Syntax void UIOptions.enableFlashGadgetType(boolean enabled)

Enables or disables the Flash gadget type change control.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the Flash gadget type change control, use code similar to the following:

```
viewerOpts.enableFlashGadgetType(false);
```

actuate.viewer.UIOptions.enableFormat

Syntax void UIOptions.enableFormat(boolean enabled)

Enables or disables the format editing feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the format editing feature, use code similar to the following:

```
viewerOpts.enableFormat(false);
```

actuate.viewer.UIOptions.enableGroupEdit

Syntax void UIOptions.enableGroupEdit(boolean enabled)

Enables or disables the group editing feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the group editing feature, use code similar to the following:

```
viewerOpts.enableGroupEdit(false);
```

actuate.viewer.UIOptions.enableHideShowItems

Syntax void UIOptions.enableHideShowItems(boolean enabled)

Enables or disables the hide/show item feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the hide/show feature, use code similar to the following:

```
viewerOpts.enableHideShowItems(false);
```

actuate.viewer.UIOptions.enableHighlight

Syntax void UIOptions.enableHighlight(boolean enabled)

Enables or disables the highlight feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the highlight feature, use code similar to the following:

```
viewerOpts.enableHighlight(false);
```

actuate.viewer.UIOptions.enableHoverHighlight

Syntax void UIOptions.enableHoverHighlight(boolean enabled)

Enables or disables the hover highlight feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the hover highlight feature, use code similar to the following:

```
viewerOpts.enableHoverHighlight(false);
```

actuate.viewer.UIOptions.enableLaunchViewer

Syntax void UIOptions.enableLaunchViewer(boolean enabled)

Enables or disables the launch viewer feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the launch viewer feature, use code similar to the following:

```
viewerOpts.enableLaunchViewer(false);
```

actuate.viewer.UIOptions.enableLinkToThisPage

Syntax void UIOptions.enableLinkToThisPage(boolean enabled)

Enables or disables the "link to this page" feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the "link to this page" feature, use code similar to the following:

```
viewerOpts.enableLinkToThisPage(false);
```

actuate.viewer.UIOptions.enableMainMenu

Syntax void UIOptions.enableMainMenu(boolean enabled)

Enables or disables the main menu feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the main menu feature, use code similar to the following:

```
viewerOpts.enableMainMenu(false);
```

actuate.viewer.UIOptions.enableMoveColumn

Syntax void UIOptions.enableMoveColumn(boolean enabled)

Enables or disables the option to move columns.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the option to move columns, use code similar to the following:

```
viewerOpts.enableMoveColumn(false);
```

actuate.viewer.UIOptions.enablePageBreak

Syntax void UIOptions.enablePageBreak(boolean enabled)

Enables or disables the page break editing feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the page break editing feature, use code similar to the following:

```
viewerOpts.enablePageBreak(false);
```

actuate.viewer.UIOptions.enablePageNavigation

Syntax void UIOptions.enablePageNavigation(boolean enabled)

Enables or disables the page navigation feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the page navigation feature, use code similar to the following:

```
viewerOpts.enablePageNavigation(false);
```

actuate.viewer.UIOptions.enableParameterPage

Syntax void UIOptions.enableParameterPage(boolean enabled)

Enables or disables the parameter page feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the parameter page feature, use code similar to the following:

```
viewerOpts.enableParameterPage(false);
```

actuate.viewer.UIOptions.enablePrint

Syntax void UIOptions.enablePrint(boolean enabled)

Enables or disables the print feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the print feature, use code similar to the following:

```
viewerOpts.enablePrint(false);
```

actuate.viewer.UIOptions.enableReorderColumns

Syntax void UIOptions.enableReorderColumns(boolean enabled)

Enables or disables the column reordering feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the column reordering feature, use code similar to the following:

```
viewerOpts.enableReorderColumns(false);
```

actuate.viewer.UIOptions.enableRowResize

Syntax void UIOptions.enableRowResize(boolean enabled)

Enables or disables row resizing.

Parameters **enabled**
Boolean. True enables this option.

Example To disable row resizing, use code similar to the following:

```
viewerOpts.enableRowResize(false);
```

actuate.viewer.UIOptions.enableSaveDesign

Syntax void UIOptions.enableSaveDesign(boolean enabled)

Enables or disables the report design save feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the report design save feature, use code similar to the following:

```
viewerOpts.enableSaveDesign(false);
```

actuate.viewer.UIOptions.enableSaveDocument

Syntax void UIOptions.enableSaveDocument(boolean enabled)

Enables or disables the report document save feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the report document save feature, use code similar to the following:

```
viewerOpts.enableSaveDocument(false);
```

actuate.viewer.UIOptions.enableShowToolTip

Syntax void UIOptions.enableShowToolTip(boolean enabled)

Enables or disables the showing of tooltips.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the showing of tooltips, use code similar to the following:

```
viewerOpts.enableShowToolTip(false);
```

actuate.viewer.UIOptions.enableSort

Syntax void UIOptions.enableSort(boolean enabled)

Enables or disables the sort feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the sort feature, use code similar to the following:

```
viewerOpts.enableSort(false);
```

actuate.viewer.UIOptions.enableSuppressDuplicate

Syntax void UIOptions.enableSuppressDuplicate(boolean enabled)

Enables or disables the duplication suppression feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the duplication suppression feature, use code similar to the following:

```
viewerOpts.enableSuppressDuplicate(false);
```

actuate.viewer.UIOptions.enableSwitchView

Syntax void UIOptions.enableSwitchView(boolean enabled)

Enables or disables the switch view feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the switch view feature, use code similar to the following:

```
viewerOpts.enableSwitchView(false);
```

actuate.viewer.UIOptions.enableTextEdit

Syntax void UIOptions.enableTextEdit(boolean enabled)

Enables or disables the text editing feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the text editing feature, use code similar to the following:

```
viewerOpts.enableTextEdit(false);
```

actuate.viewer.UIOptions.enableTOC

Syntax void UIOptions.enableTOC(boolean enabled)

Enables or disables the table of contents feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the table of contents feature, use code similar to the following:

```
viewerOpts.enableTOC(false);
```

actuate.viewer.UIOptions.enableToolBar

Syntax void UIOptions.enableToolBar(boolean enabled)

Enables or disables the toolbar feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the toolbar feature, use code similar to the following:

```
viewerOpts.enableToolBar(false);
```

Example This code initializes a new viewer display, using enableToolBar(false) to disable the toolbar:

```
function initDisplay( ){
   var uioptions = new actuate.viewer.UIOptions( );
   viewer = new actuate.Viewer("viewerpane");

   var viewerwidth = 800;
   var viewerheight = 600;

   viewer.setWidth(viewerwidth);
   viewer.setHeight(viewerheight);

   uioptions.enableToolBar(false);

   viewer.setUIOptions(uioptions);

   document.getElementById("display").disabled = false;
}
```

actuate.viewer.UIOptions.enableToolbarContextMenu

Syntax void UIOptions.enableToolbarContextMenu(boolean enabled)

Enables or disables the context menu feature.

Parameters **enabled**
Boolean. True enables this option.

Example This code initializes a new viewer display, using enableToolbarHelp(true) to enable the toolbar help feature:

```
function initDisplay( ){
   var uioptions = new actuate.viewer.UIOptions( );
   viewer = new actuate.Viewer("viewerpane");

   var viewerwidth = 800;
   var viewerheight = 600;

   viewer.setWidth(viewerwidth);
   viewer.setHeight(viewerheight);

   uioptions.enableToolBar(true);
   uioptions.enableToolbarHelp(true);

   viewer.setUIOptions(uioptions);

   document.getElementById("display").disabled = false;
}
```

actuate.viewer.UIOptions.enableToolbarHelp

Syntax void UIOptions.enableToolbarHelp(boolean enabled)

Enables or disables the toolbar help feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the toolbar help feature, use code similar to the following:

```
viewerOpts.enableToolbarHelp(false);
```

actuate.viewer.UIOptions.enableTopBottomNFilter

Syntax void UIOptions.enableTopBottomNFilter(boolean enabled)

Enables or disables the top N and bottom N filter feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the top N and bottom N filter feature, use code similar to the following:

```
viewerOpts.enableTopBottomNFilter(false);
```

actuate.viewer.UIOptions.enableUndoRedo

Syntax void UIOptions.enableUndoRedo(boolean enabled)

Enables or disables the undo and redo feature.

Parameters **enabled**
Boolean. True enables this option.

Example To disable the undo and redo feature, use code similar to the following:

```
viewerOpts.enableUndoRedo(false);
```

actuate.viewer.UIOptions.getFeatureMap

Syntax object UIOptions.getFeatureMap( )

Returns the features and their Boolean values as an associative array. This function makes the name of each feature an object property and sets the value of that property to the associated enabled Boolean value.

Returns Object.

Class actuate.viewer.ViewerException

Description A container for an exception. ViewerException provides an object to pass to a handler when the user-defined ON_EXCEPTION event occurs. It contains a reference to the element that generated the exception.

Constructor

The ViewerException object is constructed when an ON_EXCEPTION event occurs. The exceptions are divided into three types, which determine the contents of the Exception object. These types are:

- ERR_CLIENT: Exception type for a client-side error
- ERR_SERVER: Exception type for a server error
- ERR_USAGE: Exception type for a JSAPI usage error

Function summary

Table 4-56 lists actuate.viewer.ViewerException functions.

Table 4-56 actuate.viewer.ViewerException functions

Function	Description
getElement( )	Returns the element for which the exception occurred
getErrorMessage( )	Returns the exception message

actuate.viewer.ViewerException.getElement

Syntax object ViewerException.getElement( )

Returns an instance of the element that caused the exception, if applicable. The instance can be an object of one of following types:

- actuate.report.Chart
- actuate.report.DataItem
- actuate.report.Label
- actuate.report.Table
- actuate.report.TextItem

To determine the object type, use the Object.getType( ) function. The type strings for the above objects are "Chart", "Data", "Label", "Table", or "Text", respectively.

Returns Object. An instance of the element that generated the exception.

Example This example displays the type of element that generated the exception in an alert box:

```
alert("Exception in " + vException.getElement.getType( ));
```

actuate.viewer.ViewerException.getErrorMessage

Syntax string ViewerException.getErrorMessage( )

Returns the error message for the exception.

Returns String. A server error message.

Example This example displays the server error code in an alert box:

```
alert("Server error message: " + vException.getErrorMessage( ));
```

Chapter

5

BIRT Data Analyzer API classes

This chapter contains the following topics:

- About the BIRT Data Analyzer JavaScript API
- Data Analyzer API reference
- Data Analyzer JavaScript classes quick reference

About the BIRT Data Analyzer JavaScript API

The Data Analyzer portion of the Actuate JavaScript API is a set of JavaScript classes that modify, analyze, and display data within cross tab elements. These classes are available to users of BIRT iHub. The Actuate JavaScript API functions that are described in this chapter invoke and control the Data Analyzer viewer and elements that are associated with the viewer. The Data Analyzer JavaScript can be placed within a web page or any other location where the Actuate JavaScript API interfaces with a cross tab.

The actuate.xtabAnalyzer class represents the Data Analyzer viewer that contains cross tab information. Load the analyzer with actuate.load( ).

```
actuate.load("xtabAnalyzer");
```

Load support for dialog boxes from the Actuate JavaScript API with actuate.load( ), as shown in the following code:

```
actuate.load("dialog");
```

Load the XTabAnalyzer and dialog components to prepare the actuate.XTabAnalyzer component for use within a web page. Call actuate.XTabAnalyzer functions to create and prepare an analytics cross tab. Call the XTabAnalyzer's submit( ) function to display an existing cross tab in a specified HTML <div> element on a web page.

Use the following JavaScript code to create an instance of a Data Analyzer viewer:

```
var ctViewer = new actuate.XTabAnalyzer("cTab");
```

In this example, cTab is the name value for the <div> element that holds the cross tab content. The web page body must contain a <div> element with an ID value of cTab, as shown in the following code:

```
<DIV ID="cTab"></DIV>
```

When no <div> element with the correct ID value exists in the web page body, the Data Analyzer viewer launches in a pop-up window.

To load a cross tab or a data cube, use setReportName( ).

```
ctViewer.setReportName("/Public/BIRT and BIRT Studio Examples
   /Crosstab Sample Revenue.rptdocument");
```

The example code loads a report document that consists of a single data cube and cross tab. The report document can be loaded into the Data Analyzer viewer directly. This sample report document installs with iHub.

To access a cross tab element that is part of a larger report, use the cross tab element's bookmark after setting the report name. A bookmark is set in a report designer or by an external function. Retrieve a cross tab element with actuate.xtabanalyzer.PageContent.getCrosstabByBookmark( ). For example, the following code retrieves a cross tab with the bookmark SampleRevenue:

```
var content = ctViewer.getCurrentPageContent( );
var crosstab = content.getCrosstabByBookmark("SampleRevenue");
```

The code in this example retrieves the current page content and the cross tab element within that page, returning an actuate.xtabanalyzer.Crosstab object. This cross tab object supports the modification of the cross tab with the functions in the Xtabanalyzer subclasses.

To set the bookmark for a cross tab element, create a bookmark for the element within BIRT Designer Professional or call setXTabBookmark( ), as shown in the following code:

```
ctViewer.setXTabBookmark("SampleRevenue");
```

This example code assigns the bookmark SampleRevenue to the cross tab.

The XTabAnalyzer.submit( ) function triggers an AJAX request to display the report with all the asynchronous operations that previous viewer functions have prepared. Call submit( ) as shown in the following code:

```
ctViewer.submit( );
```

Upon executing submit( ), the Actuate web application returns the report with the cross tab in the assigned <div> element.

Data Analyzer API reference

This section provides an alphabetic listing of the Data Analyzer API classes.

The examples in this section consist of JavaScript functions usable by a typical web page. These examples use a sample report document called reportfile.rptdocument. The sample report document contains a cross tab that has been bookmarked within BIRT Designer Professional with the value of Sample Revenue. Use any equivalent file of that design. Place the Data Analyzer viewer in the acviewer container. The acviewer container is a <div> tag in the HTML page with the following code:

```
<DIV ID="acviewer" STYLE="border-width: 1px; border-style:
  solid;display:none;"></DIV>
```

The JavaScript setup for the examples includes the initialization of the Data Analytics module and the setup of variables for use by the examples, as shown in the following code:

```
<HTML>
...
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript">
<!-- Load the xtabAnalyzer viewer component-->
actuate.load("xtabAnalyzer");
actuate.load("dialog");
actuate.initialize("../../",null,null,null,run)

var content;
var crosstab;
var viewer;
var container;
function run( ){
   container = document.getElementById("acviewer");
   viewer = new actuate.XTabAnalyzer(container);
   viewer.setReportName("reportfile.rptdocument");
   viewer.setXTabBookmark("Sample Revenue");
   viewer.submit( );
   content = viewer.getCurrentPageContent( );
   crosstab = content.getCrosstabByBookmark( );
}

<!-- JavaScript application functions -->

</SCRIPT>

<!-- Other HTML code -->
...
</HTML>
```

The viewer variable points to the XTabAnalyzer object. The content variable points to the data within the web page. The crosstab variable points to the cross tab. These variables are used throughout the examples as needed.

Place example functions in the area marked "JavaScript application functions". The section marked "Other HTML code" contains <div> and other tags necessary for the web page.

Call the examples as any other JavaScript function. For example, the following HTML code creates a button with the label "Job 1" on it. When a user clicks that button, the page runs the JavaScript function Job1.

```
<INPUT TYPE="button" CLASS="btn" VALUE="Job 1" ONCLICK="Job1( );">
```

Data Analyzer JavaScript classes quick reference

Table 5-1 lists the Data Analyzer JavaScript classes.

Table 5-1 Actuate Data Analyzer JavaScript classes

JavaScript class	Description
actuate.XTabAnalyzer	A Data Analyzer viewer component that can be embedded in an HTML page
actuate.xtabanalyzer.Crosstab	A cross tab element
actuate.xtabanalyzer.Dimension	A data dimension
actuate.xtabanalyzer.Driller	A helper class for drilling down through cross tab data
actuate.xtabanalyzer.EventConstants	Global constants for Data Analyzer events class
actuate.xtabanalyzer.Exception	Exception object sent to calling function
actuate.xtabanalyzer.Filter	Filter conditions to filter data
actuate.xtabanalyzer.GrandTotal	A cross tab grand total
actuate.xtabanalyzer.Level	A cross tab level
actuate.xtabanalyzer.LevelAttribute	An attribute for a level
actuate.xtabanalyzer.Measure	A data measure
actuate.xtabanalyzer.MemberValue	Data as a member value
actuate.xtabanalyzer.Options	Options for the cross tab
actuate.xtabanalyzer.PageContent	The content shown in the Data Analyzer viewer
actuate.xtabanalyzer.ParameterValue	A cross tab parameter value
actuate.xtabanalyzer.Sorter	Conditions for sorting data
actuate.xtabanalyzer.SubTotal	A cross tab subtotal
actuate.xtabanalyzer.Total	A cross tab total
actuate.xtabanalyzer.UIOptions	Enables UI elements of the Data Analyzer

Class actuate.XTabAnalyzer

Description The XTabAnalyzer class represents a Data Analyzer viewer, used to view and operate a crosstab.

Constructor

Syntax actuate.XTabAnalyzer( )

Constructs a new Data Analyzer object.

actuate.XTabAnalyzer(object xtabContainer, actuate.xtabanalyzer.UIOptions uiOptions)

actuate.XTabAnalyzer(string xtabContainerId, actuate.xtabanalyzer.UIOptions uiOptions)

Constructs a new Data Analyzer object in the specified container.

Parameters **xtabContainer**
Object. A document object referencing the HTML <div> element that contains the xTabAnalyzer viewer.

xtabContainerId
String. The value of the ID parameter for an HTML <div> element to hold the xTabAnalyzer viewer. For example, with 'containerName' as the xtabContainer parameter, a <DIV ID='containerName' /> tag on the page displays the viewer at the location of the <div> element.

uiOptions
actuate.xtabanalyzer.UIOptions object. Optional. UIOptions references display options for the viewer.

Function summary

Table 5-2 lists actuate.XTabAnalyzer functions.

Table 5-2 actuate.XTabAnalyzer functions

Function	Description
commit( )	Commits all changes to the report design
forceSoftRestart( )	Forces the viewer to restart
getCurrentPageContent( )	Returns the Current Page Content object
getCurrentPageNum( )	Returns the current page number
getGadgetId	Returns the gadget ID of the shown cross tab
getHeight( )	Returns the viewer height

Table 5-2 actuate.XTabAnalyzer functions (continued)

Function	Description
getLeft( )	Returns the viewer left margin
getParameterValues( )	Returns the parameter values.
getPosition( )	Returns the CSS position attribute value
getTop( )	Returns the viewer top margin
getTotalPageCount( )	Returns the total page count
getUIOptions( )	Returns the actuate.xtabanalyzer.UIOptions object assigned to this viewer
getViewer( )	Gets a viewer within a container
getWidth( )	Returns the viewer width
getXTabBookmark( )	Returns the bookmark of the cross tab displayed in the viewer
getXTabIid( )	Returns the instance ID of the cross tab displayed in the viewer
isActive( )	Checks if current viewer pop-up is active
isDashboard( )	Checks if the current viewer pop-up is a dashboard
isInteractive( )	Checks if the current viewer is interactive
registerEventHandler( )	Registers an event handler
removeEventHandler( )	Removes an event handler
reset( )	Resets the viewer object
resizeTo( )	Resizes the viewer
rollback( )	Rolls back all changes in the viewer and refreshes its content
setGadgetId	Sets the gadget id of the cross tab
setHeight( )	Sets the viewer height
setIVMode( )	Sets whether the viewer is in IV mode
setLeft( )	Sets the viewer left margin
setOnClosed( )	Sets callback when the pop-up window is closed
setPageNum( )	Sets the page number
setPosition( )	Sets the CSS position attribute
setParameterValues( )	Sets the parameter values

(continues)

Table 5-2 actuate.XTabAnalyzer functions (continued)

Function	Description
setReportletDocument Mode( )	Sets a Reportlet to document mode
setService( )	Sets the Information Console and request options
setSupportSVG( )	Sets whether or not the client browser supports SVG
setTop( )	Sets the top margin
setUIOptions( )	Sets the user interface options for the viewer
setWidth( )	Sets the viewer width
setXTabBookmark( )	Sets a bookmark for the cross tab
setXTabIid( )	Sets the instance ID of the cross tab
submit( )	Submits asynchronous operations and renders the requested components

actuate.XTabAnalyzer.commit

Syntax void XTabAnalyzer.commit(function callback)

Commits all design changes to a generated document as a single operation. If ivMode is not set to true, call setIVMode( ) to set the value of ivMode to true before calling commit( ).

Parameter **callback**
Function. The callback function called after commit finishes.

Example This example opens a design with a cross tab and pivots the cross tab:

```
function pivot( ){
// make a change to the cross tab.
   crosstab.pivot( );
   crosstab.submit( );
   viewer.commit( );
}
```

actuate.XTabAnalyzer.forceSoftRestart

Syntax void XTabAnalyzer.forceSoftRestart( )

Forces the viewer to restart.

Example This example restarts the viewer:

```
this.onclick = function(event){
   forceSoftRestart( );
}
```

actuate.XTabAnalyzer.getCurrentPageContent

Syntax actuate.xtabanalyzer.PageContent XTabAnalyzer.getCurrentPageContent( )

Returns the Current Page Content object.

Returns actuate.xtabanalyzer.PageContent object. Content from the current page.

Example This example retrieves the cross tab from the current page:

```
function getCrosstab(analyzerViewer){
   var content = analyzerViewer.getCurrentPageContent( );
   return content.getCrosstabByBookmark( );
}
```

actuate.XTabAnalyzer.getCurrentPageNum

Syntax integer XTabAnalyzer.getCurrentPageNum( )

Returns the current page number.

Returns Integer. The current page number.

Example This example retrieves the page number:

```
function retrievePageNum( ){
   return analyzerViewer.getCurrentPageNum( );
}
```

actuate.XTabAnalyzer.getGadgetId

Syntax string XTabAnalyzer.getGadgetId( )

Returns the gadget ID of the shown cross tab. This function is used for dashboard integration.

Returns String. A gadget ID.

Example This example retrieves the gadget ID:

```
function retrieveGadgetID( ){
   return analyzerViewer.getGadgetId( );
}
```

actuate.XTabAnalyzer.getHeight

Syntax integer XTabAnalyzer.getHeight( )

Returns the height of the viewer.

Returns Integer. The height in pixels.

Example This example retrieves the current height of the viewer and doubles the height if the current height is lower than 630 pixels:

```
function doubleHeight( ){
   var height = viewer.getHeight( );
   if (height < 630){
      viewer.setHeight(height * 2);
      viewer.submit( );
   }
}
```

actuate.XTabAnalyzer.getLeft

Syntax integer XTabAnalyzer.getLeft( )

Returns the left margin of the viewer.

Returns Integer. The left margin in pixels.

Example This example retrieves the position of the viewer′s left margin and moves the margin 20 pixels to the right if the left margin is fewer than 45 pixels from the left edge of the screen:

```
function moveLeftMargin( ){
   var left = viewer.getLeft( );
   if (left < 45){
      viewer.setLeft(left + 20);
      viewer.submit( );
   }
}
```

actuate.XTabAnalyzer.getParameterValues

Syntax actuate.xtabanalyzer.ParameterValue[ ] XTabAnalyzer.getParameterValues( )

Returns the parameter values.

Returns actuate.xtabanalyzer.ParameterValue[ ] or actuate.parameter.ParameterValue[ ]. An array of parameter values.

actuate.XTabAnalyzer.getPosition

Syntax string XTabAnalyzer.getPosition( )

Returns the CSS position attribute for the viewer.

Returns String. The CSS position attribute.

Example This example changes the CSS positioning type from relative to absolute:

```
function changePosition( ){
   if (viewer.getPosition( ) == 'relative'){
      viewer.setPosition('absolute');
      viewer.submit( );
   }
}
```

actuate.XTabAnalyzer.getTop

Syntax integer XTabAnalyzer.getTop( )

Returns the top margin of the viewer.

Returns Integer. The top margin in pixels.

Example This example retrieves the value for the viewer's top margin and moves the margin 20 pixels down the screen if the margin was fewer than 45 pixels from the top of the screen:

```
function moveTopMargin( ){
   var top = viewer.getTop( );
   if (top < 45){
      viewer.setTop(top + 20);
      viewer.submit( );
   }
}
```

actuate.XTabAnalyzer.getTotalPageCount

Syntax integer XTabAnalyzer.getTotalPageCount( )

Returns the total page count.

Returns Integer. The total number of pages.

Example This example displays an alert with the total page count from viewer:

```
alert("Total pages: " + viewer.getTotalPageCount( ));
```

actuate.XTabAnalyzer.getUIOptions

Syntax actuate.xtabanalyzer.UIOptions getUIOptions( )

Returns the user interface options object for the cross tab analyzer. The UIOptions object specifies what features are used within the viewer.

Returns actuate.xtabanalyzer.UIOptions object. Data Analyzer user interface options.

Example This example retrieves the user interface options and sets one of the UIOptions values:

```
function resetUIOptions( ){
   var options = viewer.getUIOptions( );
   options.enableToolbar(false);
   viewer.setUIOptions(options);
}
```

actuate.XTabAnalyzer.getViewer

Syntax static XTabAnalyzer.getViewer(HTMLElement container)

Returns a viewer by container. To retrieve the viewer for the current object, do not specify a container. This function is useful to retrieve the instance ID for a specific viewer when there are multiple viewers on a page.

Parameters **container**
HTMLElement. The container instance ID from which to retrieve the viewer.

Returns XTabAnalyzer object. The Data Analyzer viewer.

Example This example retrieves the viewer:

```
function retrieveViewer( ){
   return viewer.getViewer( );
}
```

actuate.XTabAnalyzer.getWidth

Syntax string XTabAnalyzer.getWidth( )

Returns the width value of the viewer.

Returns String. The width in pixels.

Example This example retrieves the width of the viewer, then alters it based on the size:

```
function doubleWidth( ){
   var width = viewer.getWidth( );
   if (width < 630){
      viewer.setWidth(width * 2);
      viewer.submit( );
   }
}
```

actuate.XTabAnalyzer.getXTabBookmark

Syntax string XTabAnalyzer.getXTabBookmark( )

Returns the bookmark name for the cross tab set to render in the viewer.

Returns String. The bookmark for a cross tab.

Example This example retrieves the bookmark that the cross tab is associated with, changes the bookmark, and resets the bookmark. This functionality supports the use of multiple cross tab elements within a single design.

```
function changeBookmark( ){
   var oldBookMark = viewer.getXTabBookmark( );
   viewer.setXTabBookmark("crosstab2");
   viewer.submit( );
}
```

actuate.XTabAnalyzer.getXTabIid

Syntax string XTabAnalyzer.getXTabIid( )

Returns the current instance ID of the data analyzer. This function is useful in integration with Interactive Viewer and supports the ability of Interactive Viewer to obtain and use the data analyzer instance ID.

Returns String. A data analyzer instance ID.

Example This example retrieves the data analyzer instance ID:

```
function retrieveXTablid( myviewer ){
   return myviewer.getXTablid( );
}
```

actuate.XTabAnalyzer.isActive

Syntax boolean XTabAnalyzer.isActive( )

Returns true when a pop-up containing a data analyzer is active and false in all other cases.

Returns Boolean. True indicates an active data analyzer pop-up window.

Example This example checks if a viewer exists by checking two conditions: the viewer variable exists, or isActive( ) returns true. When both conditions fail, the example code creates a new viewer object within a container:

```
function checkViewer( ){
   if(!viewer || !viewer.isActive( )){
      viewer = new actuate.XTabAnalyzer(container);
   }
}
```

actuate.XTabAnalyzer.isDashboard

Syntax boolean XTabAnalyzer.isDashboard( )

Returns true when dashboard mode is active and false in all other cases.

Returns Boolean. True indicates dashboard mode.

actuate.XTabAnalyzer.isInteractive

Syntax boolean XTabAnalyzer.isInteractive( )

Returns whether this Data Analyzer is in Interactive mode.

Returns Boolean. True indicates dashboard mode.

Example This example displays whether myDataAnalyzer is interactive:

```
alert("Interactive mode: " + myDataAnalyzer.isInteractive( ));
```

actuate.XTabAnalyzer.registerEventHandler

Syntax void XTabAnalyzer.registerEventHandler(string viewerEvent, function handler)

Registers an event handler for the specified event. This function throws actuate.xtabanalyzer.Exception when invalid arguments are passed.

Parameters **viewerEvent**
String. Specifies the event that triggers the handler call. For a list of supported events, see actuate.xtabanalyzer.EventConstants.

handler
Function. Called when the event occurs.

Example This example changes an event handler from one function to another:

```
function changeEventHandler( event ){
  viewer.removeEventHandler(actuate.xtabanalyzer.EventConstants.
                            ON_CONTENT_CHANGED,
                            oldChangedHandler);
  viewer.registerEventHandler(actuate.xtabanalyzer.
                              EventConstants.ON_CONTENT_CHANGED,
                              newChangedHandler);
}
```

actuate.XTabAnalyzer.removeEventHandler

Syntax void XTabAnalyzer.removeEventHandler(string viewerEvent, function handler)

Removes an event handler from the specified event. This function throws actuate.xtabanalyzer.Exception when invalid arguments are passed.

Parameters **viewerEvent**
String. Specifies the event from which to remove the event handler. For a list of supported events see actuate.xtabanalyzer.EventConstants.

handler
Function. The function to deregister from the event.

Example This example changes an event handler from one function to another:

```
function changeEventHandler( event ){
  viewer.removeEventHandler(actuate.xtabanalyzer.EventConstants.
                            ON_CONTENT_CHANGED,
                            oldChangedHandler);
  viewer.registerEventHandler(actuate.xtabanalyzer.
                              EventConstants.ON_CONTENT_CHANGED,
                              newChangedHandler);
}
```

actuate.XTabAnalyzer.reset

Syntax void XTabAnalyzer.reset( )

Resets the viewer to its initial state.

Example This example resets the viewer. All changes to the viewer made prior to this call are lost:

```
function resetViewer( ){
  viewer.reset( );
}
```

actuate.XTabAnalyzer.resizeTo

Syntax void XTabAnalyzer.resizeTo(integer width, integer height)

Resizes the viewer to the specified height and width.

Parameters **width**
Integer. The width in pixels.

height
Integer. The height in pixels.

Example This example resizes the viewer when the new width is fewer than 1000 pixels and the new height is fewer than 650 pixels:

```
function resizeViewer(width,height){
  if ((width < 1000) && (height < 650)){
    viewer.resizeTo(width,height);
  }
}
```

actuate.XTabAnalyzer.rollback

Syntax void XTabAnalyzer.rollback(function callback)

Rolls back all changes in the viewer since the last commit( ) call and refreshes the viewer's content. The value of ivMode must be true for rollback( ) to function.

Parameter **callback**
Function. The callback function called after rollback finishes.

Example This example rolls back all changes to the viewer made since the last commit or submit function call:

```
function rollbackViewer( ){
   viewer.rollback( );
}
```

actuate.XTabAnalyzer.setGadgetId

Syntax void XTabAnalyzer.setGadgetId(string gadgetId)

Sets the cross tab gadget ID. This function is used for dashboard integration.

Parameters **gadgetId**
String. The gadget ID used to render the cross tab.

Example This example sets the gadget ID:

```
function setGadgetID(id){
   viewer.setGadgetId(id);
}
```

actuate.XTabAnalyzer.setHeight

Syntax void XTabAnalyzer.setHeight(integer height)

Changes the height of the viewer.

Parameters **height**
Integer. The height in pixels.

Example This example retrieves the viewer's current height. When the current height is fewer than 630 pixels, the example code doubles the viewer's height.

```
function doubleHeight( ){
   var height = viewer.getHeight( );
   if (height < 630){
      height = height * 2;
      viewer.setHeight(height);
      viewer.submit( );
   }
}
```

actuate.XTabAnalyzer.setIVMode

Syntax void XTabAnalyzer.setIVMode(boolean ivMode)

Sets IVMode for the viewer. Integrating a Data Analytics viewer with the Interactive Viewer affects the undo/redo feature. When set to true, all changes to the Data Analytics viewer must be committed as one transaction. The Interactive Viewer can undo or redo the entire batch.

Parameters **ivMode**
Boolean. Set to true if using IV mode.

Example This example sets IVMode for the viewer:

```
function setViewerMode(mode){
   viewer.setIVMode(mode);
}
```

actuate.XTabAnalyzer.setLeft

Syntax void XTabAnalyzer.setLeft(integer left)

Sets the position of the viewer's left margin.

Parameters **left**
Integer. The left margin for the viewer in pixels.

Example This example retrieves the left margin of the viewer and moves the margin 20 pixels to the right when the margin is less than 45 pixels from the edge of the screen:

```
function moveLeftMargin( ){
   var left = viewer.getLeft( );
   if (left < 45){
      viewer.setLeft(left + 20);
      viewer.submit( );
   }
}
```

actuate.XTabAnalyzer.setOnClosed

Syntax void XTabAnalyzer.setOnClosed(function callback)

Sets a callback function to call when a viewer pop-up closes.

Parameters **callback**
Function. The function to call when the pop-up closes.

Example This example checks to see if a pop-up window is active and sets a callback function to trigger when the pop-up closes:

```
function setPopupCloser( ){
   if(viewer.isActive( )){
      viewer.setOnClosed(closerCallbackFunctionName);
   }
}
```

actuate.XTabAnalyzer.setPageNum

Syntax void XTabAnalyzer.sePageNum(function pageNum)

Sets the page number.

Parameters **pageNum**
Integer. The page number.

Example This example sets the sets the page number to the first page:

```
function setPageNumberToFirst( ){
   if(viewer.isActive( )){
      viewer.setPageNum(1);
   }
}
```

actuate.XTabAnalyzer.setPosition

Syntax void XTabAnalyzer.setPosition(string position)

Sets the CSS position attribute.

Parameters **position**
String. The value for the CSS position attribute.

Example This example changes the type of CSS positioning in use:

```
function changePosition( ){
   var pos = viewer.getPosition( );
   if (pos == 'relative'){
      viewer.setPosition('absolute');
      viewer.submit( );
   }
}
```

actuate.XTabAnalyzer.setParameterValues

Syntax void XTabAnalyzer.setParameterValues(actuate.xtabanalyzer.ParameterValue[ ] parameterValues)

Sets the CSS position attribute.

Parameters **parameterValues**
actuate.xtabanalyzer.ParameterValue[ ] or actuate.parameter.ParameterValue[ ]. An array of parameter values.

actuate.XTabAnalyzer.setReportletDocumentMode

Syntax void XTabAnalyzer.setReportletDocumentMode(boolean reportletMode)

Sets whether the viewer displays documents as Reportlets.

Parameters **reportletMode**
Boolean. True indicates Reportlet display mode.

actuate.XTabAnalyzer.setReportName

Syntax void XTabAnalyzer.setReportName(string reportName)

Sets the report file name for the viewer. The file must be a report document file or report design file.

Parameters **reportName**
String. The name of the report file.

Example This example sets the report name to reportfile.rptdocument and reloads the Data Analyzer viewer with its content:

```
function run( ){
   container = document.getElementById("acviewer");
   viewer = new actuate.XTabAnalyzer(container);
   viewer.setReportName("reportfile.rptdocument");
   viewer.submit( );
```

}actuate.XTabAnalyzer.setService

Syntax void XTabAnalyzer.setService(string iPortalURL, actuate.RequestOptions requestOptions)

Sets the Actuate web application URL. This function can request options for that URL.

Parameters **iPortalURL**
String. The URL of the Actuate web application.

requestOptions
actuate.RequestOptions object. Request options for the web application. This parameter is optional.

Example This example sets the service and request options:

```
function setServerOptions(URL,options){
   viewer.setService(URL,options);
}
```

actuate.XTabAnalyzer.setSupportSVG

Syntax void XTabAnalyzer.setSupportSVG(boolean svgFlag)

Sets a flag indicating whether or not the browser supports SVG.

Parameters **svgFlag**
Boolean. Flag indicating SVG support in the browser. This parameter's value is true when the browser supports SVG and false in all other cases.

Example This example sets the browser's level of SVG support:

```
function setSVG(flag){
   viewer.setSupportSVG(flag);
}
```

actuate.XTabAnalyzer.setTop

Syntax void XTabAnalyzer.setTop(integer top)

Sets the top margin for the viewer.

Parameters **top**
Integer. The top margin for the viewer in pixels.

Example This example retrieves the current top margin for the viewer and moves the margin 20 pixels down the screen when the current position of the margin is fewer than 45 pixels from the top of the screen:

```
function moveTopMargin( ){
   var top = viewer.getTop( );
   if (top < 45){
      top = top + 20;
      viewer.setTop(top);
      viewer.submit( );
   }
}
```

actuate.XTabAnalyzer.setUIOptions

Syntax void XTabAnalyzer.setUIOptions(actuate.xtabanalyzer.uioptions options)

Sets the user interface options enabled for the viewer.

Parameters **options**
Actuate.xtabanalyzer.uioptions object. The options object for the viewer.

Example This example retrieves the user interface options and sets one of the UIOptions values:

```
function resetUIOptions( ){
   var options = viewer.getUIOptions( );
   options.enableToolbar(false);
   viewer.setUIOptions(options);
}
```

actuate.XTabAnalyzer.setWidth

Syntax void XTabAnalyzer.setWidth(integer width)

Sets the width for the viewer.

Parameters **width**
Integer. The width for the viewer in pixels.

Example This example retrieves the width of the viewer. When the viewer is fewer than 630 pixels wide, the example code doubles the viewer's width:

```
function doubleWidth( ){
   var width = viewer.getWidth( );
   if (width < 630){
      viewer.setWidth(width * 2);
      viewer.submit( );
   }
}
```

actuate.XTabAnalyzer.setXTabBookmark

Syntax void XTabAnalyzer.setXTabBookmark(string bookmark)

Sets the bookmark for a cross tab to render in the viewer.

Parameters **bookmark**
String. The bookmark for a cross tab.

Example This example retrieves the bookmark for the cross tab the viewer is associated with, changes the bookmark, and reloads the bookmark. This functionality enables the use of multiple cross tab elements within a single design.

```
function changeBookmark( ){
   var oldBookMark = viewer.getXTabBookmark( );
   viewer.setXTabBookmark("crosstab2");
   viewer.submit( );
}
```

actuate.XTabAnalyzer.setXTabIid

Syntax void XTabAnalyzer.setXTabIid(string iid)

Sets the instance ID for viewer rendering. This function is useful in integration with Interactive Viewer, and supports the ability of Interactive Viewer to obtain and use the cross tab instance ID.

Parameters **iid**
String. The instance ID.

Example This example sets the cross tab instance ID:

```
function setxtabInstance(id){
   viewer.setXTablid(id);
}
```

actuate.XTabAnalyzer.submit

Syntax void XTabAnalyzer.submit(function callback, boolean rerun)

Submits requests to the server for the Data Analyzer viewer. This method triggers an AJAX request to submit all pending operations for this object. The server returns a response after processing the pending operations. The results render on the page in the Data Analyzer container. The submit( ) function throws an exception when another submit( ) operation is pending. A CONTENT_CHANGED event fires when the Data Analyzer content changes.

Parameters **callback**
Function. Optional. A function called when submit completes. This function receives the current XTabAnalyzer object as an input parameter.

rerun
Boolean. Optional. Indicates whether re-run the report design when it refreshes. Default to true.

Example This example retrieves the left margin of the viewer and expands the margin. The change does not take effect until submit( ) executes. The submit( ) function calls the function in the submitCallback parameter when submit( ) finishes executing. The callback function contains any processing that must occur after submit( ) finishes. Do not place code after the submit( ) call in the same function because submit( ) is asynchronous.

```
function moveLeftMargin( ){
   var left = viewer.getLeft( );
   if (left < 45){
      viewer.setLeft(left + 20);
      viewer.submit(submitCallback);
   }
}
```

Class actuate.xtabanalyzer.Crosstab

Description The actuate.xtabanalyzer.Crosstab class represents a cross tab report element.

Constructor

Syntax actuate.xtabanalyzer.Crosstab( )

Constructs a new Crosstab object.

Function summary

Table 5-3 lists actuate.xtabanalyzer.Crosstab functions.

Table 5-3 actuate.xtabanalyzer.Crosstab functions

Function	Description
addDimension( )	Adds a dimension to the cross tab
addMeasure( )	Adds a measure to the cross tab
applyOptions( )	Sets options for the cross tab
changeMeasureDirection( )	Switches measure direction
clearFilters( )	Clears cross tab filters
drill( )	Drills up or down measure levels, replacing drill and filter conditions
drillDown( )	Drills down a measure level, updating drill conditions
drillUp( )	Drills up a measure level, updating drill conditions
editMeasure( )	Edits a measure
getBookmark( )	Retrieves the cross tab element bookmark
getColumn( )	Retrieves table data by column index
getData( )	Returns the data from a cross tab
getHtmlDom( )	Retrieves the HTML DOM object
getPageContent( )	Retrieves the content of the page the cross tab belongs to
getRow( )	Retrieves table data by row index
getType( )	Retrieves the report element type
hideDetail( )	Hides the detail of a specified level

(continues)

Table 5-3 actuate.xtabanalyzer.Crosstab functions (continued)

Function	Description
pivot( )	Pivots the cross tab
removeDimension( )	Removes a dimension from the cross tab
removeMeasure( )	Removes a measure from the cross tab
reorderDimension( )	Reorders a dimension
reorderMeasure( )	Reorders a measure
setFilters( )	Sets the cross tab's filters
setSorters( )	Sets the cross tab's sorters
setTotals( )	Sets the cross tab's totals
showDetail( )	Shows details to the lower level
submit( )	Applies changes made to the cross tab

actuate.xtabanalyzer.Crosstab.addDimension

Syntax void Crosstab.addDimension(actuate.xtabanalyzer.Dimension dimension)

Adds a dimension to the cross tab object.

Parameters **dimension**

actuate.xtabanalyzer.Dimension object. The dimension to add.

Example This example adds a date-based, multi-level dimension to a cross tab:

```
function addDimension( ){
// Create a dimension for dates in the first column
   var dimension = new actuate.xtabanalyzer.Dimension( );
   dimension.setIndex(0);
   dimension.setAxisType(actuate.xtabanalyzer.Dimension.
                         COLUMN_AXIS_TYPE);
   dimension.setDimensionName("dates");

// Create levels using levels from the data cube.
   var level = new actuate.xtabanalyzer.Level( );
   level.setLevelName("year");
   dimension.addLevel(level);
   var level = new actuate.xtabanalyzer.Level( );
   level.setLevelName("quarter");
   dimension.addLevel(level);

// Add the dimension to the cross tab.
   crosstab.addDimension(dimension);
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.addMeasure

Syntax void Crosstab.addMeasure(actuate.xtabanalyzer.Measure measure, integer options)

Adds a measure to the cross tab object.

Parameters

measure
actuate.xtabanalyzer.Measure object. The measure to add.

options
Integer. The options for the add measure operation. These options distinguish the function call's origin, which can be from another dialog or directly from the Actuate JavaScript API.

Example This example adds a measure to a cross tab:

```
function addMeasure( ){
//Create a measure for revenue organized by date and product line.
   var measure = new actuate.xtabanalyzer.Measure( );
   measure.setIndex(1);
   measure.setMeasureName("Quarter Rate");
   measure.setExpression("[revenue]/[revenue_SalesDate
                         /year_Product/PRODUCTLINE]");

// Apply the measure to the cross tab
   crosstab.addMeasure(measure);
   crosstab.submit( );
}
```

In this example, the expression set with setExpression( ) is in easyscript, which is described in *Using Actuate BIRT Designer Professional*.

actuate.xtabanalyzer.Crosstab.applyOptions

Syntax void Crosstab.applyOptions(string | actuate.xtabanalyzer.Options measureDirection, string rowMirrorStartingLevel, string columnMirrorStartingLevel, string emptyCellValue)

Sets measure direction, empty settings, row mirror starting level, column mirror starting level, and empty cell value.

Parameters

measureDirection
String or actuate.xtabanalyzer.Options object. When measureDirection is a string, measureDirection is set to horizontal or vertical and the other parameters set options individually. When an actuate.xtabanalyzer.Options object is specified, all the options are set using settings from this object and applyOptions ignores all subsequent parameters.

rowMirrorStartingLevel
String. Sets the mirror starting level empty setting for a row.

columnMirrorStartingLevel
String. Sets the mirror starting level empty setting for a column.

emptyCellValue
String. Sets the value of an empty cell.

actuate.xtabanalyzer.Crosstab.changeMeasureDirection

Syntax void Crosstab.changeMeasureDirection( )

Switches the measure direction between horizontal and vertical.

Example This example changes the measure direction:

```
function changeMeasureDirection( ){
   if( crosstab ){
      crosstab.changeMeasureDirection( );
      crosstab.submit( );
   }
}
```

actuate.xtabanalyzer.Crosstab.clearFilters

Syntax void Crosstab.clearFilters(actuate.xtabanalyzer.Level level, String filterType)

Clears the filters from a level.

Parameters **level**
actuate.xtabanalyzer.Level object. Optional. The level from which to clear the filters. To clear all filters, do not specify a level.

filterType
String. Optional. The filter type. To clear all filter types, do not specify a filter type.

Example This example clears the filters from the level filterLevel:

```
function clearLevelFilters( ){
   if( crosstab ){
      crosstab.clearFilters("filterLevel");
      crosstab.submit( );
   }
}
```

actuate.xtabanalyzer.Crosstab.drill

Syntax void Crosstab.drill(actuate.xtabanalyzer.Driller driller)

Drills up or down a dimension level. Removes all drill/filter conditions defined on specified dimension first, then adds new drill/filter conditions.

Parameters **driller**
actuate.xtabanalyzer.Driller object. The driller object specifies drill conditions on a dimension.

Example This example drills to a level within a dimension. Any existing drill conditions are replaced.

```
function drillToDimension(memberVal){
   var driller = new actuate.xtabanalyzer.Driller( );
   driller.setAxisType(actuate.xtabanalyzer.Dimension.
                        ROW_AXIS_TYPE);
   driller.addMember(memberVal);
   myCrosstab.drill(driller);
   myCrosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.drillDown

Syntax void Crosstab.drillDown(actuate.xtabanalyzer.Driller driller)

Drills down a dimension level. This method updates the drill conditions specified in the Driller object and leaves all other conditions in place.

Parameters **driller**
actuate.xtabanalyzer.Driller object. A drill condition object.

Example This example drills down a level within a dimension. Any existing drill conditions are unchanged.

```
function drillToDimension(memberVal){
   var driller = new actuate.xtabanalyzer.Driller( );
   driller.setAxisType(actuate.xtabanalyzer.Dimension.
                        ROW_AXIS_TYPE);
   driller.addMember(memberVal);
   myCrosstab.drillDown(driller);
   myCrosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.drillUp

Syntax void Crosstab.drillUp(actuate.xtabanalyzer.Driller driller)

Drills up a dimension level. This method updates the drill conditions specified in the Driller object and leaves all other conditions in place.

Parameters **driller**
A drill condition object.

Example This example drills up a level within a dimension. Any existing drill conditions are unchanged.

```
function drillToDimension( ){
   var driller = new actuate.xtabanalyzer.Driller( );
   driller.setAxisType(actuate.xtabanalyzer.Dimension.
                         ROW_AXIS_TYPE);
// Add the member list to the Driller. Add the Driller to the
// crosstab.
   driller.addMember(memberVal);
   myCrosstab.drillUp(driller);
   myCrosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.editMeasure

Syntax void Crosstab.editMeasure(actuate.xtabanalyzer.Meaure Measure, integer opts)

Edits a measure in the Computed Measure view.

Parameters **Measure**
actuate.xtabanalyzer.Measure object. A measure to change.

opts
Integer. Optional. Options for the editMeasure function. These options distinguish the function call's origin, which can be from another dialog or directly from the Actuate JavaScript API.

Example This example edits a measure:

```
function editComputedMeasure( ){
   if( crosstab ){
      var measure = new actuate.xtabanalyzer.Measure( );
      measure.setMeasureName("measureName");
      measure.setExpression("measureExpression");
      crosstab.editMeasure(measure);
      crosstab.submit( );
   }
}
```

actuate.xtabanalyzer.Crosstab.getBookmark

Syntax string Crosstab.getBookmark( )

Returns the bookmark that is associated with the cross tab element.

Returns String. The cross tab bookmark.

Example The following code retrieves the bookmark that is associated with the cross tab object:

```
function getCrosstabBookmark( ){
   var crosstabBookmark = crosstab.getBookmark( );
   if( !crosstabBookmark ){
      alert( "No cross tab bookmark found!" )
      return null;
   }
   return crosstabBookmark;
}
```

actuate.xtabanalyzer.Crosstab.getColumn

Syntax string[ ] Crosstab.getColumn(integer columnIndex)

Returns the table data by column index.

Parameters **columnIndex**
Integer. The column index, starting with 1.

Returns String[ ]. The column data as an array of strings. This function returns null when the value of columnIndex is out of range. This function only returns data from the current visible page.

Example The following code retrieves data from a data column:

```
function getColumnData(index,value){
   var columnData = crosstab.getColumn(index);
   if( !columnData ){
      alert( "Invalid column index!" )
      return null;
   }
   return columnData[value];
}
```

actuate.xtabanalyzer.Crosstab.getData

Syntax String[ ] Crosstab.getData(boolean forceReparse)

Returns the data in a cross tab.

Parameters **forceReparse**
Boolean. Forces a cache refresh when true.

Returns String[ ]. The data from the cross tab as an array of strings.

actuate.xtabanalyzer.Crosstab.getHtmlDom

Syntax HTMLElement Crosstab.getHtmlDom( )

Returns the HTML element DOM object.

Returns HTMLElement. The DOM element containing the cross tab.

Example The following code retrieves the DOM object and uses the DOM object to retrieve an element within the document:

```
function getContainer(containerName){
   var HTMLDom = crosstab.getHtmlDom( );
   var container = HTMLDom.getElementById(containerName);
   return container;
}
```

actuate.xtabanalyzer.Crosstab.getPageContent

Syntax actuate.xtabanalyzer.PageContent Crosstab.getPageContent( )

Returns the page content from the current page to which this cross tab belongs. This function returns the same information as XTabAnalyzer.getCurrentPageContent( ).

Returns actuate.xtabanalyzer.PageContent. The report content.

Example This example retrieves the page content:

```
function retrievePageContent( ){
   return crosstab.getPageContent( );
}
```

actuate.xtabanalyzer.Crosstab.getRow

Syntax string[ ] Crosstab.getRow(integer rowIndex)

Returns table data based on row index.

Parameters **rowIndex**
Integer. The row index, starting with 1.

Returns String[ ]. The row data as an array of string values. This function returns null when the value of rowIndex is out of range. This function only returns data from the current visible page.

Example The following code retrieves data from a data row:

```
function getRowData(index,value){
   var rowData = crosstab.getRow(index);
   if( !rowData ){
      alert( "Invalid row index!" )
      return null;
   }

   return rowData[value];
}
```

actuate.xtabanalyzer.Crosstab.getType

Syntax string Crosstab.getType( )

Returns the report element type.

Returns String containing the value "Crosstab".

actuate.xtabanalyzer.Crosstab.hideDetail

Syntax void Crosstab.hideDetail(string levelName)

Hides details of the specified level.

Parameters **levelName**
String. The full name of a dimension level to hide.

Example This example hides lower level details in a level:

```
function hideDetail( ){
   if(crosstab){
      var levelName = "rollLevelName";
      crosstab.hideDetail(levelName);
      crosstab.submit( );
   }
}
```

actuate.xtabanalyzer.Crosstab.pivot

Syntax void Crosstab.pivot( )

Pivots the cross tab.

Example This example pivots a cross tab:

```
function pivot(crosstab){
   crosstab.pivot( );
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.removeDimension

Syntax void Crosstab.removeDimension(object dimension, integer axisType,
    integer[ ] levels)

Removes a dimension from the cross tab.

Parameters **dimension**
actuate.xtabanalyzer.dimension object, a dimension index, or a dimension name.
The dimension to remove.

axisType
Integer. The dimension axis type. Axis type can be one of the following values:

- actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE
- actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE

levels
The levels assigned in the dimension, as an array of actuate.xtabanalyzer.Level objects, a level index array, or a level name array.

Example This example removes a dimension with several layers. The level names are in a text control named levelNames and are separated by semicolons.

```
function removeDimension( ){
   if(crosstab){
      crosstab.removeDimension("dimensionName",null,"levelName";);
      crosstab.submit( );
   }
}
```

actuate.xtabanalyzer.Crosstab.reorderDimension

Syntax void Crosstab.reorderDimension(actuate.xtabanalyzer.Dimension dimension, integer axisType, integer newIndex, integer newAxisType)

Reorders a dimension within a cross tab. This function can change a dimension's index or axis type.

Parameters **dimension**
actuate.xtabanalyzer.dimension object, or a dimension index or a dimension name. The dimension to reorder.

axisType
Integer. The dimension axis type. Axis type can be one of the following values:

- actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE
- actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE

newIndex
The new index for the dimension.

newAxisType
The new axis type.

Example This example changes the index and axis type of a dimension:

```
function changeDimensionOrder( ){
   var dimensionIndex = 5;
   var newDimensionIndex = 2;

   var axisType = actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE;
   var newAxisType = actuate.xtabanalyzer.Dimension.
                     COLUMN_AXIS_TYPE;
   crosstab.reorderDimension(dimensionIndex, axisType,
                              newDimensionIndex, newAxisType);
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.removeMeasure

Syntax void Crosstab.removeMeasure(actuate.xtabanalyzer.Measure measure)

void Crosstab.removeMeasure(integer measure)

void Crosstab.removeMeasure(string measure)

Removes a measure from the cross tab.

Parameters **measure**
actuate.xtabanalyzer.measure object, index, or name. The measure to remove.

Example This example removes a measure from a cross tab:

```
function removeMeasure( ){
   crosstab.removeMeasure("measureName");
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.reorderMeasure

Syntax void Crosstab.reorderMeasure(actuate.xtabanalyzerMeasure measure,
   integer newIndex)

void Crosstab.reorderMeasure(integer measure,integer newIndex)

void Crosstab.reorderMeasure(string measure,integer newIndex)

Reorders a measure within a cross tab.

Parameters **measure**
actuate.xtabanalyzer.Measure object, or a measure index or a measure name. The measure to reorder.

newIndex
The new index for the measure.

Example This example reorders a measure:

```
function changeMeasureOrder( ){
   var index = 6;
   var newIndex = 3;
   crosstab.reorderMeasure(index,newIndex);
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.setFilters

Syntax void Crosstab.setFilters(actuate.xtabanalyzer.Filter[ ] filters)

Sets an array of filters for the cross tab.

Parameters **filters**
Array of actuate.xtabanalyzer.Filter objects. The filter conditions.

Example This example creates a Filter object and then places it into the cross tab:

```
function filterLevel( ){
   var levelName = "levelName";
   var operator = "BETWEEN";
   var filterValue = "20000;50000";
   var filter = new actuate.xtabanalyzer.Filter(levelName,
                operator);
   filter.setValues(filterValue.split(";"));
   crosstab.setFilters(filter);
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.setSorters

Syntax void Crosstab.setSorters(actuate.xtabanalyzer.Sorter[ ] sorters)

Sets an array of sorters for the cross tab.

Parameters **sorters**
Array of actuate.xtabanalyzer.Sorter objects. The sort settings.

Example This example creates a sorter and adds it to the cross tab:

```
function sortLevel( ){
   var levelName = "levelName";
   var sortAscending = true;
   var sorter = new actuate.xtabanalyzer.Sorter(levelName);
   sorter.setAscending(sortAscending);
   crosstab.setSorters(sorter);
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.setTotals

Syntax void Crosstab.setTotals(actuate.xtabanalyzer.GrandTotal[ ] grandTotals, actuate.xtabanalyzer.SubTotal[ ] subTotals)

Sets totals for the cross tab.

Parameters **grandTotals**
Array of actuate.xtabanalyzer.GrandTotal objects. Grand totals. To set a subtotal, set this parameter to null.

subTotals
Array of actuate.xtabanalyzer.SubTotal objects. Subtotals.

Example This example adds a grand total to a cross tab:

```
function addGrandTotal( ){
   var grandTotal = new actuate.xtabanalyzer.GrandTotal( );
   grandTotal.setAxisType(
      actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE);

   var total = new actuate.xtabanalyzer.Total( );
   total.setMeasureIndex(1);
   total.setAggregationFunction("SUM");
   total.setEnabled(true);
   grandTotal.addTotal(total);

   crosstab.setTotals(grandTotal);
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.showDetail

Syntax void Crosstab.showDetail(string axisType)

Shows a level of detail within a cross tab.

Parameters **axisType**
String. The dimension axis type. Axis type can be one of the following values:

- actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE
- actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE

Example This example uses showDetail to expose extra detail on a level:

```
function showDetail( ){
   var axisType = actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE;
   crosstab.showDetail(axisType);
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Crosstab.submit

Syntax void Crosstab.submit(function callback)

Applies the changes made to this element. This is an asynchronous operation.

Parameters **callback**
Function. Optional. The function called when submit( ) completes. This function receives the current XTabAnalyzer object as an input parameter.

Example This example uses submit( ) to confirm changes to the cross tab:

```
function showDetail(crosstab){
   var axisType = actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE;
   crosstab.showDetail(axisType);
   crosstab.submit( );
}
```

Class actuate.xtabanalyzer.Dimension

Description The Dimension class specifies a cross tab Dimension object.

Constructor

Syntax actuate.xtabanalyzer.Dimension( )

The Dimension class is used to specify a Dimension object.

Function summary

Table 5-4 lists actuate.xtabanalyzer.Dimension functions.

Table 5-4 actuate.xtabanalyzer.Dimension functions

Function	Description
addLevel( )	Adds the level to the dimension
getAxisType( )	Returns the axis type
getDimensionName( )	Returns the dimension name
getIndex( )	Returns the index of the dimension
getLevels( )	Returns cross tab levels
getNewAxisType( )	Returns the new axis type
getNewIndex( )	Returns the new index
setAxisType( )	Sets the axis type
setDimensionName( )	Sets the dimension name
setIndex( )	Sets the index
setLevels( )	Sets the levels
setNewAxisType( )	Sets the new axis type
setNewIndex( )	Sets the new index axis type

actuate.xtabanalyzer.Dimension.addLevel

Syntax void Dimension.addLevel(actuate.xtabanalyzer.Level level)

Adds a level to the dimension.

Parameters **level**
actuate.xtabanalyzer.Level object. A level to add to the dimension.

Example This example adds a level to a dimension:

```
function addLvl(dimension,levelName){
   var level = new actuate.xtabanalyzer.Level( );
   level.setLevelName(levelName);
   dimension.addLevel(level);
}
```

actuate.xtabanalyzer.Dimension.getAxisType

Syntax integer Dimension.getAxisType( )

Returns the axis type for the dimension.

Returns Integer. The axis type can be one of the following values:

- actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE
- actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE

Example This example retrieves and sets the axis type:

```
function swapAxis(dimension){
   if (dimension.getAxisType( ) ==
         actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE){
      dimension.setNewAxisType(
      actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE);
   } else {
      dimension.setNewAxisType(
      actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE);
   }
}
```

actuate.xtabanalyzer.Dimension.getDimensionName

Syntax string Dimension.getDimensionName( )

Returns the name of this dimension.

Returns String. The dimension name.

Example This example retrieves the dimension name:

```
function getDimName(dimension){
   if(dimension){
      return dimension.getDimensionName( );
   }
   return null;
}
```

actuate.xtabanalyzer.Dimension.getIndex

Syntax integer Dimension.getIndex( )

Returns the dimension index.

Returns Integer. The dimension index.

Example This example retrieves and increments the index:

```
function incrementIndex(dimension){
   var newIndex = dimension.getIndex( ) + 1;
   dimension.setNewIndex(newIndex);
}
```

actuate.xtabanalyzer.Dimension.getLevels

Syntax actuate.xtabanalyzer.Level[ ] Dimension.getLevels( )

Returns the dimension levels.

Returns actuate.xtabanalyzer.Level[ ]. Array of dimension levels.

Example This example retrieves the dimension levels:

```
function getDimLevels(dimension){
   if(dimension){
      return dimension.getLevels( );
   }
   return null;
}
```

actuate.xtabanalyzer.Dimension.getNewAxisType

Syntax integer Dimension.getNewAxisType( )

Returns the new axis type.

Returns Integer containing the new axis type.

Example This example retrieves the new axis type:

```
function getNewDimAxis(dimension){
   if(dimension){
      return dimension.getNewAxisType( );
   }
   return null;
}
```

actuate.xtabanalyzer.Dimension.getNewIndex

Syntax integer Dimension.getNewIndex( )

Returns the new index.

Returns Integer. The new index.

Example This example retrieves the new index:

```
function getNewDimIndex(dimension){
   if(dimension){
      return dimension.getNewIndex( );
   }
   return null;
}
```

actuate.xtabanalyzer.Dimension.setAxisType

Syntax void Dimension.setAxisType(integer axisType)

Sets the axis type when creating a new dimension. Use setNewAxisType( ) to change a dimension that already exists.

Parameters **axisType**
The axis type for the dimension. The axis type has the following legal values:

- actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE
- actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE

Example This example sets the axis type for a new dimension:

```
function setRowAxis(dimension){
      dimension.setAxisType(
      actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE);
}
```

actuate.xtabanalyzer.Dimension.setDimensionName

Syntax void Dimension.setDimensionName(string dimensionName)

Sets the name for a dimension during its creation.

Parameters **dimensionName**
String. The name of the dimension.

Example This example sets the dimension name to a value taken from a page element:

```
function setDimensionName(dimension){
   var dimensionName = document.getElementById("dimensionName").
                         value;
   dimension.setDimensionName(dimensionName);
}
```

actuate.xtabanalyzer.Dimension.setIndex

Syntax void Dimension.setIndex(integer index)

Sets the index for the dimension.

Parameters **index**
The index of the dimension.

Example This example sets the dimension index to a value taken from a page element:

```
function setDimensionIndex(dimension){
   var dimensionIndex = document.getElementById("dimensionIndex").
                        value;
   dimension.setIndex(dimensionIndex);
}
```

actuate.xtabanalyzer.Dimension.setLevels

Syntax void Dimension.setLevels(xtabanalyzer.Level[ ] levels)

Sets levels for the dimension.

Parameters **levels**
Array of xtabanalyzer.Level objects representing the levels for the dimension.

Example This example sets the dimension levels:

```
function setDimensionLevels(dimension,levels){
   if (dimension && levels){
      dimension.setLevels(levels);
   }
}
```

actuate.xtabanalyzer.Dimension.setNewAxisType

Syntax void Dimension.setNewAxisType(integer newAxisType)

Sets the new axis type.

Parameters **newAxisType**
Integer. The new axis type.

Example This example retrieves and changes the axis type:

```
function swapAxis(dimension){
   if (dimension.getAxisType( ) ==
         actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE){
      dimension.setNewAxisType(
         actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE);
   } else {
      dimension.setNewAxisType(
         actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE);
   }
}
```

actuate.xtabanalyzer.Dimension.setNewIndex

Syntax void Dimension.setNewtIndex(integer newIndex)

Sets the new index.

Parameters **newIndex**
Integer. The new index.

Example This example retrieves and increments the index:

```
function incrementIndex(dimension){
   var newIndex = dimension.getIndex( ) + 1;
   dimension.setNewIndex(newIndex);
}
```

Class actuate.xtabanalyzer.Driller

Description The Driller class enables an application to drill down or up levels on a member within a dimension.

Constructor

Syntax actuate.xtabanalyzer.Driller( )

Creates a Driller object.

Function summary

Table 5-5 lists actuate.xtabanalyzer.Driller functions.

Table 5-5 actuate.xtabanalyzer.Driller functions

Function	Description
addMember( )	Adds a member to the drill condition
getDimension( )	Retrieves the driller dimension
getMembers( )	Retrieves the members used by the drill
setDimension( )	Sets the driller dimension
setMembers( )	Adds an array of members to the drill condition

actuate.xtabanalyzer.Driller.addMember

Syntax void Driller.addMember(actuate.xtabanalyzer.MemberValue member)

Adds a dimension member to the drill condition. Functional candidates are Dimension members with levels.

Parameters **member**
actuate.xtabanalyzer.MemberValue object. A member value to add.

Example This example adds a member to a Driller object:

```
function drillDownDimension( ){
   var driller = new actuate.xtabanalyzer.Driller( );
   driller.setDimension(actuate.xtabanalyzer.Dimension.
                        ROW_AXIS_TYPE);
   var memberValue = new actuate.xtabanalyzer.
                     MemberValue("drillLevelName");
   memberValue.setValue("drillLevelValue");
   driller.addMember(memberValue);
   crosstab.drill( driller );
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Driller.getDimension

Syntax string Driller.getDimension( )

Returns the dimension name for the drill condition.

Returns String. A dimension name.

Example This example retrieves the dimension of the driller:

```
function getDrillerAxis(driller){
   if (driller){
      return driller.getDimension( );
   }
   return null;
}
```

actuate.xtabanalyzer.Driller.getMembers

Syntax actuate.xtabanalyzer.MemberValue[ ] Driller.getMembers( )

returns the list of members assigned to the driller.

Returns Array of actuate.xtabanalyzer.MemberValue. A dimension member.

Example This example retrieves the members that a driller uses:

```
function getDrillerMembers(driller){
   if (driller){
      return driller.getMembers( );
   }
   return null;
}
```

actuate.xtabanalyzer.Driller.setDimension

Syntax void Driller.setDimension(string dimension)

Sets the dimension for the driller by name.

Parameters **dimension**
String. A dimension name.

Example This example sets the dimension name for the driller:

```
function setRowAxis(driller){
   if (driller){
      dimension.setDimension("Row");
   }
}
```

actuate.xtabanalyzer.Driller.setMembers

Syntax void Driller.setMembers(actuate.xtabanalyzer.MemberValue[ ] member)

Sets an array of members to the drill condition.

Parameters **member**
Array of actuate.xtabanalyzer.MemberValue objects. An array of members.

Example This example sets the axis type for the driller:

```
function setDrillerMembers(driller,members){
   if (driller && members){
      driller.setMembers(members);
   }
}
```

Class actuate.xtabanalyzer.EventConstants

Description Defines constants for xtabanalyzer events. Table 5-6 lists the cross tab analyzer event constants.

Table 5-6 actuate.xtabanalyzer.Dimension constants

Constant	Description
ON_CONTENT_CHANGED	Content changed event. Triggers when the displayed content has changed, for example when changing cross tab report content. The event handler takes an actuate.XTabAnalyzer object that represents the viewer for which the event occurred, as the only parameter.
ON_CONTENT_SELECTED	Content selected event. Triggers when a user clicks on report elements. The event handler takes the following parameters: ■ actuate.XTabAnalyzer: object viewer for which event occurred ■ actuate.xtabanalyzer.SelectedContent: the SelectedContent object
ON_EXCEPTION	Exception event. Triggers when an exception occurs during an asynchronous operation. The event handler takes the following arguments: ■ actuate.XTabAnalyzer: viewer for which the event occurred ■ actuate.Exception: Exception object
ON_SESSION_TIMEOUT	Session time-out event. When a session time-out event occurs and the user tries to perform any operation on a viewer, a prompt dialog appears asking the user whether or not to log in again. When the user chooses to log in again, the ON_SESSION_TIMEOUT event triggers. When no handler is registered for this event, a default built-in login dialog will be displayed. This event is only available if the SessionReloginTimeout flag is set to true in the server configuration from which the API was initialized. The event handler takes one parameter: an actuate.XTabAnalyzer object, representing the viewer where the event occurred.

Class actuate.xtabanalyzer.Exception

Description A container for an XTabAnalyzer exception that supports specific exceptions. The Exception class provides an object to pass to a callback function or event handler when an exception occurs. The Exception class contains references to the exception's origin, description, and messages.

Constructor

The Exception object is constructed when unspecified exceptions occur. The exceptions are divided into three types, which determine the contents of the Exception object. These types are:

- ERR_CLIENT: Exception type for a client-side error
- ERR_SERVER: Exception type for a server error
- ERR_USAGE: Exception type for a JSAPI usage error

Function summary

Table 5-7 lists actuate.xtabanalyzer.Exception functions.

Table 5-7 actuate.xtabanalyzer.Exception functions

Function	Description
getDescription( )	Returns details of the exception
getElement( )	Returns the report element for which the exception occurred, if available
getErrCode( )	Returns the error code for ERR_SERVER
getMessage( )	Returns a short message about the error
getType( )	Returns the type of error exception
isExceptionType( )	Returns Boolean indicating whether exception is of certain type

actuate.xtabanalyzer.Exception.getDescription

Syntax string Exception.getDescription( )

Returns exception details as provided by the Server, Client, and User objects.

Returns String. A detailed description of the error. Information is provided according to the type of exception generated, as shown below:

- ERR_SERVER: The SOAP string

- ERR_CLIENT: For the Firefox browser, a list comprised of fileName+number+stack
- ERR_USAGE: Any value set when the object was created

Example This example consists of a function that registerEventHandler( ) set as a callback. The callback function takes an instance of the Exception class. Each of the functions for the Exception class can be called with the results formatted to create a message or for some other use.

```
function errorHandler(viewerInstance, exception){
   alert(exception.getDescription( ));
}
```

actuate.xtabanalyzer.Exception.getElement

Syntax string Exception.getElement( )

Returns the report element for which the exception occurred, if available.

Returns String. The report element for which the exception occurred.

Example This example uses getElement( ):

```
function errorHandler(viewerInstance, exception){
   alert("Error in " + exception.getElement( ));
}
```

actuate.xtabanalyzer.Exception.getErrCode

Syntax string Exception.getErrCode( )

Returns the error code for ERR_SERVER.

Returns String. The error code for ERR_SERVER.

Example This example uses getErrCode( ):

```
function errorHandler(viewerInstance, exception){
   alert(exception.getErrCode( ));
}
```

actuate.xtabanalyzer.Exception.getMessage

Syntax string Exception.getMessage( )

Returns a short message about the error.

Returns String. A short message about the exception.

Example This example uses getMessage( ):

```
function errorHandler(viewerInstance, exception){
   alert(exception.getMessage( ));
}
```

actuate.xtabanalyzer.Exception.getType

Syntax string Exception.getType( )

Returns the type of exception error.

Returns String. The errType exception type.

Example This example uses getType( ):

```
function errorHandler(viewerInstance, exception){
   alert(exception.getType( ));
}
```

actuate.xtabanalyzer.Exception.isExceptionType

Syntax boolean Exception.isExceptionType(object exceptionType)

Checks an exception's type for a match against a specified type.

Parameters **exceptionType**
An exception type as string, or exception class. For example, "actuate.viewer.ViewerException" or actuate.viewer.ViewerException.

Returns True if the exception is of the stated type, false otherwise.

Example This example checks to see if the exception is a client error type:

```
function errorHandler(viewerInstance, exception){
   if (exception.isExceptionType(ERR_CLIENT){
      alert("CLIENT ERROR");
   }
}
```

Class actuate.xtabanalyzer.Filter

Description The Filter class creates a filter condition on a cross tab dimension level. The condition is expressed as value1 operator value2. The values can either be a single value, or an array of values, depending on the operator. For example, IN can be expressed as value1 IN value2 value3 ... valueN.

Constructor

Syntax

```
actuate.data.Filter(string levelName, string levelAttributeName, string operator,
    string value, string filterType)
```

```
actuate.data.Filter(string levelName, string levelAttributeName, string operator,
    string value1, string value2, string filterType)
```

```
actuate.data.Filter(string levelName, string levelAttributeName, string operator,
    string[ ] values, string filterType)
```

Constructs a cross tab Filter object.

Parameters

levelName
String. The dimension level full name.

levelAttributeName
String. The dimension level attribute name.

operator
String. The operator can be any operator. Table 5-8 lists the valid filter operators and the number of arguments to pass to the constructor or setValues( ).

Table 5-8 Filter operators

Operator	Description	Number of arguments
BETWEEN	Between an inclusive range	2
BOTTOM_N	Matches the bottom n values	1
BOTTOM_PERCENT	Matches the bottom percent of the values	1
EQ	Equal	1
FALSE	Matches false Boolean values	0
GREATER_THAN	Greater than	1
GREATER_THAN_OR_EQUAL	Greater than or equal	1

Table 5-8 Filter operators

Operator	Description	Number of arguments
IN	Matches any value in a set of values	1+
LESS_THAN	Less than	1
LESS_THAN_OR_EQUAL	Less than or equal	1
LIKE	Search for a pattern	1
MATCH	Equal	1
NOT_BETWEEN	Not between an inclusive range	2
NOT_EQ	Not equal	1
NOT_IN	Does not match any value in a set of values	1+
NOT_LIKE	Searches for values that do not match a pattern	1
NOT_MATCH	Not equal	1
NOT_NULL	Is not null	0
NULL	Is null	0
TOP_N	Matches the top n values	1
TOP_PERCENT	Matches the top percent of the values	1
TRUE	Matches true Boolean values	0

value
String. The value to compare to the column value.

value1
String. The first value to compare to the column value for the BETWEEN or NOT_BETWEEN operators.

value2
String. The second value to compare to the column value for the BETWEEN or NOT_BETWEEN operators.

values
Array of strings. The values to compare to the column value for the IN and NOT_IN operators.

filterType
String. The filter type.

Function summary

Table 5-9 lists actuate.xtabanalyzer.Filter functions.

Table 5-9 actuate.xtabanalyzer.Filter functions

Function	Description
getFilterType( )	Returns the filter type
getLevelAttributeName( )	Returns the dimension level attribute name
getLevelName( )	Returns the name of the filtered level
getOperator( )	Returns the filter operator
getValues( )	Returns the set of values the filter is using
setFilterType( )	Sets the filter type
setLevelAttributeName( )	Sets the dimension level attribute name
setLevelName( )	Sets the dimension level name
setOperator( )	Sets the filter operator
setValues( )	Sets the values for the filter

actuate.xtabanalyzer.Filter.getFilterType

Syntax string Filter.getFilterType( )

Returns the filter type.

Returns String. The filter type.

Example This example retrieves the filter type for a filter:

```
function getType(filter){
   if(filter){
      return filter.getFilterType( );
   }
   else{
      return null;
   }
}
```

actuate.xtabanalyzer.Filter.getLevelAttributeName

Syntax string Filter.getLevelAttribute Name( )

Returns the name of the dimension level attribute to which this filter applies.

Returns String. The level attribute name.

Example This example retrieves the filter level attribute name for a filter:

```
function getLevelAttribute(filter){
   if(filter){
      return filter.getLevelAttributeName( );
   }
   else{
      return null;
   }
}
```

actuate.xtabanalyzer.Filter.getLevelName

Syntax string Filter.getLevelName( )

Returns the name of the dimension level to which this filter applies.

Returns String. A level name.

Example This example retrieves the filter level name for a filter:

```
function getLevel(filter){
   if(filter){
      return filter.getLevelName( );
   }
   else{
      return null;
   }
}
```

actuate.xtabanalyzer.Filter.getOperator

Syntax string Filter.getOperator( )

Returns the filter operator.

Returns String. The filter operator.

Example This example retrieves the filter operator:

```
function getFilterOp(filter){
   if(filter){
      return filter.getOperator( );
   }else{
      return null;
   }
}
```

actuate.xtabanalyzer.Filter.getValues

Syntax string[ ] Filter.getValues( )

Returns an array containing the values used in the filter.

Returns Array of strings. The values for the filter.

Example This example retrieves the filter level name for a filter:

```
function getFilterOp(filter){
   if(filter){
      return filter.getValues( );
   }else{
      return null;
   }
}
```

actuate.xtabanalyzer.Filter.setFilterType

Syntax void Filter.setFilterType(string filterType)

Sets the filter type to filter.

Parameters **filterType**
String. The type of filter.

Example This example sets the filter type to equality:

```
function filterLevel( ){
   var filterType = "equality";
   var filter = new actuate.xtabanalyzer.Filter("levelName",
      "attributeName","EQ","2000","blank");
   filter.setFilterType(filterType);
   crosstab.setFilters( filter );
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Filter.setLevelAttributeName

Syntax void Filter.setLevelAttributeName(string levelAttributeName)

Sets the dimension level attribute to filter on by name.

Parameters **levelAttributeName**
String. The name of the level attribute to filter.

Example This example sets the level attribute name to attributeName:

```
function filterLevel( ){
   var attributeName = "attributeName";
   var filter = new actuate.xtabanalyzer.Filter("levelName",
                    "blank","EQ","2000","equality");
   filter.setLevelAttributeName(attributeName);
   crosstab.setFilters( filter );
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Filter.setLevelName

Syntax void Filter.setLevelName(string level)

Sets the level to filter by name.

Parameters **level**
String. The name of the level to filter.

Example This example sets the filter level name to levelName:

```
function filterLevel( ){
   var levelName = "levelName";
   var filter = new actuate.xtabanalyzer.Filter("blank",
                    "attributeName",EQ","2000","equality");
   filter.setLevelName(levelName);
   crosstab.setFilters( filter );
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Filter.setOperator

Syntax void Filter.setOperator(string operator)

Sets the filter operator.

Parameters **operator**
String. The filter operator.

Example This example sets the filter operator to EQ:

```
function filterLevel( ){
   var operator = "EQ";
   var filter = new actuate.xtabanalyzer.Filter("levelName",
                    "attributeName","NOT","2000","equality");
   filter.setOperator(operator);
   crosstab.setFilters( filter );
   crosstab.submit( );
}
```

actuate.xtabanalyzer.Filter.setValues

Syntax void Filter.setValues(string[ ] value1, string[ ] value2)

Sets the values for the filter.

Parameters

value1
String or array of strings. The first value of the filter.

value2
String or array of strings. Optional. The second value of the filter.

Example This example sets the filter values to 2000 and 2004:

```
function filterLevel( ){
   if(crosstab){
      var filterValue = "2000;2004";
      var filter = new actuate.xtabanalyzer.Filter
                     ("levelName","attributeName","BETWEEN");
      filter.setValues(filterValue.split(";") );
      crosstab.setFilters( filter );
      crosstab.submit( );
   }
}
```

Class actuate.xtabanalyzer.GrandTotal

Description The GrandTotal class specifies a cross tab GrandTotal object.

Constructor

Syntax actuate.xtabanalyzer.GrandTotal( )

Constructs a new GrandTotal object.

Function summary

Table 5-10 lists actuate.xtabanalyzer.GrandTotal functions.

Table 5-10 actuate.xtabanalyzer.GrandTotal functions

Function	Description
addTotal( )	Adds a total
getAxisType( )	Returns the axis type
getTotals( )	Returns the totals array
getType( )	Returns the grand total type
setAxisType( )	Sets the axis type
setTotals( )	Sets the totals array

actuate.xtabanalyzer.GrandTotal.addTotal

Syntax void GrandTotal.addTotal(object total)

Adds a total to the cross tab.

Parameters **total**
actuate.xtabanalyzer.total. The total to add to the cross tab.

Example This example adds totals to a grand total:

```
function addTotal(grandTotal){
// The indexStr can be set from a web page or other source as
// necessary.
   var indexStr = "0;1;2;3;4";
   var indexs = indexsStr.split(";");
   var count = indexs.length;
   var measureIndexs = [ ];
   for(var i = 0;i < count;i++){
      measureIndexs.push(parseInt(indexs[i]));
   }
   for( var i = 0; i < measureIndexs.length; i++){
      var total = new actuate.xtabanalyzer.Total( );
      total.setMeasureIndex(measureIndexs[i]);
      total.setAggregationFunction("SUM");
      total.setEnabled(true);
      grandTotal.addTotal(total);
   }
}
```

actuate.xtabanalyzer.GrandTotal.getAxisType

Syntax integer GrandTotal.getAxisType( )

Returns the axis type for the total.

Returns Integer. The following values are legal axis types:

- actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE
- actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE

Example This example retrieves and sets the axis type:

```
function swapAxis(grandtotal){
   if (grandtotal.getAxisType( ) ==
         actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE){
      grandtotal.setNewAxisType(
      actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE);
   } else {
      grandtotal.setNewAxisType(
      actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE);
   }
}
```

actuate.xtabanalyzer.GrandTotal.getTotals

Syntax object[ ] GrandTotal.getTotals( )

Returns an array containing the totals.

Returns Array of Total objects. The totals.

Example This example retrieves totals from a GrandTotal object:

```
var totalsArray = [ ];
function getTotals(grandTotal,totalsArray){
   totalsArray = grandTotal.getTotals( );
}
```

actuate.xtabanalyzer.GrandTotal.getType

Syntax string GrandTotal.getType( )

Returns the type for the total.

Returns String. The total type.

actuate.xtabanalyzer.GrandTotal.setAxisType

Syntax void GrandTotal.setAxisType(integer axisType)

Sets the axis type for the total.

Parameters **axisType**
Integer. Axis type for the total.

Example This example retrieves and sets the axis type:

```
function swapAxis(grandtotal){
   if (grandtotal.getAxisType( ) ==
         actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE){
      grandtotal.setNewAxisType(
      actuate.xtabanalyzer.Dimension.COLUMN_AXIS_TYPE);
   } else {
      grandtotal.setNewAxisType(
      actuate.xtabanalyzer.Dimension.ROW_AXIS_TYPE);
   }
}
```

actuate.xtabanalyzer.GrandTotal.setTotals

Syntax void GrandTotal.setTotals(actuate.xtabanalyzer.Total[ ] totals)

Sets totals as an array.

Parameters **totals**
Array of actuate.xtabanalyzer.Total objects to add to the grand total.

Example This example copies the totals from grandtotal1 into grandtotal2:

```
grandtotal2.setTotals(grandtotal1.getTotals( ));
```

Class actuate.xtabanalyzer.Level

Description Defines a cross tab dimension level, its controls, and content.

Constructor

Syntax actuate.xtabanalyzer.Level( )

Creates a cross tab Level object.

Function summary

Table 5-11 lists actuate.xtabanalyzer.Level functions.

Table 5-11 actuate.xtabanalyzer.Level functions

Function	Description
addAttribute( )	Adds the level attribute
getAttributes( )	Returns the level attributes
getIndex( )	Returns the index of the level
getLevelName( )	Returns the level name
setIndex( )	Sets the index level
setLevelName( )	Sets the level name

actuate.xtabanalyzer.Level.addAttribute

Syntax void Level.addAttribute(actuate.xtabanalyzer.LevelAttribute attr)

Adds the level attribute.

Parameters **index**
actuate.xtabanalyzer.LevelAttribute object. A level attribute.

Example This example sets a name for newly created level attribute and assigns the attribute to a level:

```
var attribute = new actuate.xtabanalyzer.LevelAttribute( );
attribute.setName("pounds");
level.addLevelAttribute( attribute );
```

actuate.xtabanalyzer.Level.getAttributes

Syntax actuate.xtabanalyzer.LevelAttribute[ ] Level.getAttributes( )

Returns the level attributes.

Returns Array of actuate.xtabanalyzer.LevelAttribute objects. The level attributes.

Example This example retrieves the level index and stores it in a variable called lattributes:

```
var lattributes = new actuate,xtabanalyzer.LevelAttribute[ ];
lattributes = level.getAttributes( );
```

actuate.xtabanalyzer.Level.getIndex

Syntax integer Level.getIndex( )

Returns the level index.

Returns Integer. The level index.

Example This example retrieves the level index:

```
function levelIndex(level){
   if (level){
      return level.getIndex( );
   }
   return null;
}
```

actuate.xtabanalyzer.Level.getLevelName

Syntax string Level.getLevelName( )

Returns the level name.

Returns String. The level name.

Example This example retrieves the level name:

```
function levelName(level){
   if (level){
      return level.getLevelName( );
   }
   return null;
}
```

actuate.xtabanalyzer.Level.setIndex

Syntax void Level.setIndex(integer index)

Sets the level index.

Parameters **index**
Integer. The level index.

Example This example sets the level index:

```
function assignIndex(level,index){
   if (level){
      return level.setIndex(index);
   }
}
```

actuate.xtabanalyzer.Level.setLevelName

Syntax void Level.setLevelName(string levelName)

Sets the level name.

Parameters **levelName**
String. The level name.

Example This example sets level names for newly created levels:

```
var levelNames ="year;month;day";
...
function addLevels(dimension,levelNames);{
   var levelNamesArray = levelNames.split(";");
   for( var i = 0; i < levelNames.length; i++ ){
      var level = new actuate.xtabanalyzer.Level( );
      level.setLevelName(levelNames[i]);
      dimension.addLevel( level );
   }
}
```

Class actuate.xtabanalyzer.LevelAttribute

Description Defines an attribute for a level.

Constructor

Syntax actuate.xtabanalyzer.LevelAttribute( )

Creates a cross tab level attribute object.

Function summary

Table 5-12 lists actuate.xtabanalyzer.LevelAttribute functions.

Table 5-12 actuate.xtabanalyzer.Level functions

Function	Description
getName( )	Returns the level attribute name
setName( )	Sets the level attribute name

actuate.xtabanalyzer.LevelAttribute.getName

Syntax string LevelAttribute.getName( )

Returns the level attribute name.

Returns String. A level attribute name.

Example This example retrieves the level attribute name and stores it in a variable attname:

```
var attname = levelattribute.getName( );
```

actuate.xtabanalyzer.LevelAttribute.setName

Syntax void LevelAttribute.setName(string attributeName)

Sets the level attribute name.

Parameters **attributeName**
String. The level attribute name.

Example This example sets a name for newly created level attribute and assigns the attribute to a level:

```
var attribute = new actuate.xtabanalyzer.LevelAttribute( );
attribute.setName("pounds");
level.addLevelAttribute( attribute );
```

Class actuate.xtabanalyzer.Measure

Description Defines a cross tab measure.

Constructor

Syntax actuate.xtabanalyzer.Measure( )

Creates a cross tab measure object.

Function summary

Table 5-13 lists actuate.xtabanalyzer.Measure functions.

Table 5-13 actuate.xtabanalyzer.Measure functions

Function	Description
getAggregationFunction( )	Returns the aggregation function name
getDataType( )	Returns the computed column data type
getExpression( )	Returns the computed measure expression
getIndex( )	Returns the measure index
getMeasureName( )	Returns the measure name
getNewIndex( )	Returns the new index
setAggregationFunction( )	Sets the aggregation function name
setDataType( )	Sets the computed column data type
setExpression( )	Sets the computed measure expression
setIndex( )	Sets the measure index
setMeasureName( )	Sets the measure name
setNewIndex( )	Sets the new index

actuate.xtabanalyzer.Measure .getAggregationFunction

Syntax string Measure.getAggregationFunction( )

Returns the aggregation function name.

Returns String. An aggregation function name.

Example This example changes the aggregation function:

```
function swapMeasureAggregation(measure){
   if (measure.getAggregation( ) == "EQ"){
      measure.setAggregation("NE");
   }else{
      measure.setAggregation("EQ");
   }
}
```

actuate.xtabanalyzer.Measure.getDataType

Syntax string Measure.getDataType( )

Returns the computed column data type.

Returns String. The data type.

Example This example retrieves the computed column data type:

```
function getColumnDataType(measure){
   if (measure){
      return measure.getDataType( );
   }
   return null;
}
```

actuate.xtabanalyzer.Measure.getExpression

Syntax string Measure.getExpression( )

Returns the computed measure expression.

Returns String. An expression.

Example This example retrieves the computed measure expression:

```
function getMeasureExpression(measure){
   if (measure){
      return measure.getExpression( );
   }
   return null;
}
```

actuate.xtabanalyzer.Measure.getIndex

Syntax integer Measure.getIndex( )

Returns the measure index.

Returns Integer. The measure index.

Example This example retrieves the measure index:

```
function getMeasureIndex(measure){
   if (measure){
      return measure.getIndex( );
   }
   return null;
}
```

actuate.xtabanalyzer.Measure.getMeasureName

Syntax string Measure.getMeasureName( )

Returns the measure name.

Returns String. The name of the measure.

Example This example retrieves the measure name:

```
function getMeasureName(measure){
   if (measure){
      return measure.getMeasureName( );
   }
   return null;
}
```

actuate.xtabanalyzer.Measure.getNewIndex

Syntax integer Measure.getNewIndex( )

Retrieves the new index. The new index is set by setNewIndex and represents the index value the measure has after submit( ) finishes executing.

Returns Integer. The new index.

Example This example retrieves the new measure index:

```
function getNewMeasureIndex(measure){
   if (measure){
      return measure.getNewIndex( );
   }
   return null;
}
```

actuate.xtabanalyzer.Measure .setAggregationFunction

Syntax void Measure.setAggregationFunction(string aggregationFunction)

Sets the aggregation function name.

Parameters **aggregationFunction**
String. The aggregation function name.

Example This example changes the aggregation function:

```
function swapMeasureAggregation(measure){
   if (measure.getAggregation( ) == "EQ"){
      measure.setAggregation("NE");
   }else{
      measure.setAggregation("EQ");
   }
}
```

actuate.xtabanalyzer.Measure.setDataType

Syntax void Measure.setDataType(string dataType)

Sets the computed column data type name.

Parameters **dataType**
String. The data type.

actuate.xtabanalyzer.Measure.setExpression

Syntax void Measure.setExpression(string expression)

Sets the computed measure expression.

Parameters **expression**
String. The computed measure expression.

Example This example uses setExpression:

```
function addMeasure(viewer){
   var crosstab = getCrosstab(viewer);
   if(crosstab){
      var measureName  = "measureName";
      var measureExpression  =
         "[revenue]/[revenue_SalesDate/year_Product/PRODUCTLINE]";

      var measure = new actuate.xtabanalyzer.Measure( );
      measure.setIndex(1);
      measure.setMeasureName(measureName);
      measure.setExpression(measureExpression);

      crosstab.addMeasure(measure);
      crosstab.submit( );
   }
}
```

actuate.xtabanalyzer.Measure.setIndex

Syntax void Measure.setIndex(integer index)

Sets the index.

Parameters **index**
Integer. The index of this measure.

Example This example uses setIndex to add a new measure to a cross tab:

```
function setIndex(measure, index){
   measure.setIndex(index);
}
```

actuate.xtabanalyzer.Measure.setMeasureName

Syntax void Measure.setMeasureName(string measureName)

Sets the measure name.

Parameters **measureName**
String. The measureName.

Example This example sets the measure name which is taken from a page element:

```
function renameMeasure(measure){
   var measureName = document.getElementById("measureName").value;
   measure.setMeasureName(measureName);
}
```

actuate.xtabanalyzer.Measure.setNewIndex

Syntax void Measure.setNewIndex(integer newIndex)

Sets a new measure index.

Parameters **newIndex**
Integer. The new measure index.

Example This example changes the index for the measure:

```
function changeIndex(measure,index){
   if (measure){
      measure.setNewIndex(index);
   }
}
```

Class actuate.xtabanalyzer.MemberValue

Description Defines a member value used for sort, filter, or drill functionality.

Constructor

Syntax actuate.xtabanalyzer.MemberValue(levelName, value, (MemberValue))

Creates a MemberValue object for a given level and value. The object can contain multiple member values.

Parameters **levelName**
String. Dimension level name of member.

value
String. Value for the member to contain.

MemberValue
Optional actuate.xtabanalyzer.MemberValue object. MemberValue object to add during construction.

Function summary

Table 5-14 lists actuate.xtabanalyzer.MemberValue functions.

Table 5-14 actuate.xtabanalyzer.MemberValue functions

Function	Description
addMember( )	Adds a member value object
getLevelName( )	Retrieves the level name
getMembers( )	Retrieves an array of members
getValue( )	Returns the level value
setLevelName( )	Sets the level name
setValue( )	Sets the member value

actuate.xtabanalyzer.MemberValue.addMember

Syntax void MemberValue.addMember(actuate.xtabanalyzer.MemberValue member)

Adds a member value.

Parameters **member**
actuate.xtabanalyzer.MemberValue object. A member value.

Example MemberValue is an embedded class that can be a single value or an array of values. This example has a single member that contains four members:

```
function addMembers(memberData){
   var mv1 = new MemberValue('dim/state','CA');
   var mv2 = new MemberValue('dim/state','CN');
   var mv3 = new MemberValue(memberData);
   var mv = new MemberValue('dim/country','USA');
   mv.addMember(mv1);
   mv.addMember(mv2);
   mv.addMember(mv3);
   return mv;
}
```

actuate.xtabanalyzer.MemberValue.getLevelName

Syntax string MemberValue.getLevelName( )

Returns the level name of the member.

Returns String. The level name.

Example This example retrieves the level name for the member value:

```
function getLevelName(level){
   if (level){
      return level.getLevelName( );
   }
   return null;
}
```

actuate.xtabanalyzer.MemberValue.getMembers

Syntax actuate.xtabanalyzer.MemberValue[ ] MemberValue.getMembers( )

Returns all the member value objects contained in this member value object.

Returns Array of actuate.xtabanalyzer.MemberValue. An array of MemberValue objects.

Example This example returns the number of members in a member object:

```
function getMemberCount(members){
   if (members){
      var membersArray[] = members.getMembers( );
      return membersArray.length;
   }
   return null;
}
```

actuate.xtabanalyzer.MemberValue.getValue

Syntax string MemberValue.getValue( )

Returns the level value.

Returns String. The level value.

Example This example returns the value for the level:

```
function getMemberValue(members){
   if (members){
      return members.getValue( );
   }
   return null;
}
```

actuate.xtabanalyzer.MemberValue.setLevelName

Syntax void MemberValue.setLevelName(string level)

Sets the level name.

Parameters **level**
String. The name of the level.

Example This example sets the level name:

```
function setMemberLevelName(member,lvlName){
   if (member){
      member.setLevelName(lvlName);
   }
}
```

actuate.xtabanalyzer.MemberValue.setValue

Syntax void MemberValue.setValue(string level)

Sets the level value.

Parameters **level**
String. The value for the level.

Example This example sets the level value:

```
function setMemberLevelValue(member,lvlValue){
   if (member){
      member.setValue(lvlValue);
   }
}
```

Class actuate.xtabanalyzer.Options

Description The Options class specifies options for the cross tab.

Constructor

Syntax actuate.xtabanalyzer.Options(string measureDirection, string rowMirrorStartingLevel, string columnMirrorStartingLevel, string emptyCellValue, boolean enablePageBreak, integer rowPageBreakInterval, integer columnPageBreakInterval)

Creates an options object that contains options for how the cross tab displays data.

Parameters **measureDirection**
String. The measure direction. Legal values for measure direction are:

- DIRECTION_HORIZONTAL
- DIRECTION_VERTICAL

rowMirrorStartingLevel
String. Row mirror starting level name.

columnMirrorStartingLevel
String. Column mirror starting level name.

emptyCellValue
String. Value to display for an empty cell.

enablePageBreak
Boolean. Enables page breaks when true.

rowPageBreakInterval
Integer. Row page break interval.

columnPageBreakInterval
Integer. Column page break interval.

grandTotalsDisplayOption
String. Grand totals display option. Legal values for total display options are:

- DIRECTION_HORIZONTAL
- DIRECTION_VERTICAL

subtotalsDisplayOption
String. Subtotals display option. Legal values for total display options are:

- DIRECTION_HORIZONTAL
- DIRECTION_VERTICAL

Function summary

Table 5-15 lists actuate.xtabanalyzer.Options functions.

Table 5-15 actuate.xtabanalyzer.Options functions

Function	Description
getColumnMirrorStartingLevel( )	Returns the column mirror starting level full name
getColumnPageBreakInterval( )	Returns the column page break interval
getEmptyCellValue( )	Returns the empty cell value
getEnablePageBreak( )	Returns the page break enabled or disabled status
getMeasureDirection( )	Returns the measure direction
getRowMirrorStartingLevel( )	Returns the row mirror starting level full name
getRowPageBreakInterval( )	Returns the row page break interval
setColumnMirrorStartingLevel( )	Sets the column mirror starting level full name
setColumnPageBreakInterval( )	Sets the column page break interval
setEmptyCellValue( )	Sets the empty cell value
setEnablePageBreak( )	Sets the flag to enable page breaks
setMeasureDirection( )	Sets the measure direction
setRowMirrorStartingLevel( )	Sets the row mirror starting level full name
setRowPageBreakInterval( )	Sets the row page break interval

actuate.xtabanalyzer.Options .getColumnMirrorStartingLevel

Syntax string Options.getColumnMirrorStartingLevel( )

Returns the column mirror starting level name.

Returns String. Column mirror starting level name.

Example This example retrieves the column mirror starting level:

```
function getColumnMirrorStart(options){
   if (options){
      return options.getColumnMirrorStartinglevel( );
   }
   return null;
}
```

actuate.xtabanalyzer.Options .getColumnPageBreakInterval

Syntax integer Options.getColumnPageBreakInterval( )

Returns the column page break interval.

Returns Integer. The column page break interval.

Example This example retrieves the column page break interval:

```
function getColumnPBInterval(options){
   if (options){
      return options.getColumnPageBreakInterval( );
   }
   return null;
}
```

actuate.xtabanalyzer.Options.getEmptyCellValue

Syntax string Options.getEmptyCellValue( )

Returns the empty cell value.

Returns String. Value to display for an empty cell.

Example This example retrieves the empty cell:

```
function getEmptyCell(options){
   if (options){
      return options.getEmptyCellValue( );
   }
   return null;
}
```

actuate.xtabanalyzer.Options.getEnablePageBreak

Syntax boolean Options.getEnablePageBreak( )

Returns the page break status.

Returns Boolean. Page breaks are enabled when the value is true.

Example This example retrieves the column page break interval when page breaks are enabled:

```
function getColumnPBEnabled(options){
   if (options.getEnablePageBreak( ))
      return options.getColumnPageBreakInterval( );
   else
      alert ("Page Breaks Not Enabled.");
   return null;
}
```

actuate.xtabanalyzer.Options.getMeasureDirection

Syntax string Options.getMeasureDirection( )

Returns the measure direction.

Returns String. The measure direction. Legal values for measure direction are:

- DIRECTION_HORIZONTAL
- DIRECTION_VERTICAL

Example This example retrieves the measure direction:

```
function getMeasureDirection(options){
   if (options){
      return options.getMeasureDirection( );
   }
   return null;
}
```

actuate.xtabanalyzer.Options .getRowMirrorStartingLevel

Syntax string Options.getRowMirrorStartingLevel( )

Returns the row mirror starting level name.

Returns String. Row mirror starting level name.

Example This example retrieves the row mirror starting level:

```
function getRowMirrorStart(options){
   if (options){
      return options.getRowMirrorStartinglevel( );
   }
   return null;
}
```

actuate.xtabanalyzer.Options .getRowPageBreakInterval

Syntax integer Options.getRowPageBreakInterval( )

Returns the row page break interval.

Returns Integer. The row page break interval.

Example This example retrieves the row page break interval:

```
function getRowPBInterval(options){
   if (options){
      return options.getRowPageBreakInterval( );
   }
   return null;
}
```

actuate.xtabanalyzer.Options .setColumnMirrorStartingLevel

Syntax void Options.setColumnMirrorStartingLevel(string levelName)

Sets the column mirror starting level name.

Parameters **levelName**
String. The column mirror starting level name.

Example This example sets the column mirror starting level:

```
function setColumnMirrorLevel(options,level)(
   if (options){
      options.setColumnMirrorStartingLevel(level);
   }
}
```

actuate.xtabanalyzer.Options .setColumnPageBreakInterval

Syntax void Options.setColumnPageBreakInterval(integer columnPageBreakInterval)

Sets the column page break interval.

Parameters **columnPageBreakInterval**
Integer. The column page break interval.

Example This example sets the column page break interval:

```
function setColumnPBInterval(options,interval)(
   if (options){
      options.setColumnPageBreakInterval(interval);
   }
}
```

actuate.xtabanalyzer.Options.setEmptyCellValue

Syntax void Options.setEmptyCellValue(string emptyCellValue)

Sets the empty cell value.

Parameters **emptyCellValue**
String. The empty cell value.

Example This example sets the empty cell value:

```
function setEmptyCell(options, cellValue)(
   if (options){
      options.setEmptyCellValue(cellValue);
   }
}
```

actuate.xtabanalyzer.Options.setEnablePageBreak

Syntax void Options.setEnablePageBreak(boolean enablePageBreak)

Enables or disables page breaks.

Parameters **enablePageBreak**
Boolean. Enables page breaks when true.

Example This example enables page breaks and sets the row page break interval:

```
function enablesetRowPBInterval(options,interval)(
   if (options){
      options.setEnablePageBreak(true);
      options.setRowPageBreakInterval(interval);
   }
}
```

actuate.xtabanalyzer.Options.setMeasureDirection

Syntax void Options.setMeasureDirection(string measureDirection)

Sets the measure direction.

Parameters **measureDirection**
String. The measure direction. The measure direction. Legal values for measure direction are:

- DIRECTION_HORIZONTAL
- DIRECTION_VERTICAL

Example This example sets the measure direction:

```
function setMeasureDirection(options,direction){
   if (options){
      options.setMeasureDirection(direction);
   }
}
```

actuate.xtabanalyzer.Options .setRowMirrorStartingLevel

Syntax void Options.setRowMirrorStartingLevel(string levelName)

Sets the row mirror starting level.

Parameters **levelName**
String. Row mirror starting level name.

Example This example sets the row mirror starting level:

```
function setRowMirrorLevel(options,level){
   if (options){
      options.setRowMirrorStartingLevel(level);
   }
}
```

actuate.xtabanalyzer.Options .setRowPageBreakInterval

Syntax void Options.setRowPageBreakInterval(integer rowPageBreakInterval)

Sets the row page break interval.

Parameters **rowPageBreakInterval**
Integer. The row page break interval.

Example This example sets the row page break interval:

```
function setRowPBInterval(options,interval)(
   if (options){
      options.setRowPageBreakInterval(interval);
   }
}
```

Class actuate.xtabanalyzer.PageContent

Description A container for the content of a cross tab page. It contains a comprehensive list of report elements, such as tables, charts, labels, and data items.

Constructor

Syntax actuate.xtabanalyzer.PageContent( )

Creates a PageContent object that represents the report content that is generated by a report design file or document file.

Function summary

Table 5-16 lists actuate.xtabanalyzer.PageContent functions.

Table 5-16 actuate.xtabanalyzer.PageContent functions

Function	Description
getCrosstabByBookmark( )	Returns a report cross tab object
getViewerId( )	Returns the cross tab viewer ID

actuate.xtabanalyzer.PageContent .getCrosstabByBookmark

Syntax actuate.xtabanalyzer.crosstab PageContent.getCrosstabByBookmark(string bookmark)

Returns a cross tab object associated with a bookmark.

Parameters **bookmark**
The bookmark name of the item requested.

Returns actuate.xtabanalyzer.crosstab object.

Example This example retrieves the viewer ID, then retrieves the cross tab:

```
function getCrosstab( ){
   var viewer = PageContent.getViewerId( );
   var content = viewer.getCurrentPageContent( );
   var crosstab = content.getCrosstabByBookmark( );
   return crosstab;
}
```

actuate.xtabanalyzer.PageContent.getViewerId

Syntax string PageContent.getViewerId( )

Returns the XTabAnalyzer ID. The XTabAnalyzer is the cross tab viewer element.

Returns String. The XTabAnalyzer ID.

Example This example retrieves the viewer ID, then retrieves the cross tab:

```
function getCrosstab( ){
   var viewer = PageContent.getViewerId( );
   var content = viewer.getCurrentPageContent( );
   var crosstab = content.getCrosstabByBookmark( );
   return crosstab;
}
```

Class actuate.xtabanalyzer.ParameterValue

Description A container for the ParameterValue in the xtabanalyzer.

Constructor

Syntax

```
actuate.xtabanalyzer.ParameterValue(string name, string value, boolean
    valueIsNull)
```

The ParameterValue class is used to specify a cross tab ParameterValue object.

Parameters

name
String. The parameter name.

value
String. The parameter value.

valueIsNull
Boolean. Whether the value is null.

Function summary

Table 5-17 lists actuate.xtabanalyzer.ParameterValue functions.

Table 5-17 actuate.xtabanalyzer.ParameterValue functions

Function	Description
getName( )	Returns the parameter name
getValue( )	Returns the parameter value
getValueIsNull( )	Returns whether the parameter has a null value
setName( )	Sets the parameter name
setEnabled( )	Sets the parameter value
setMeasureIndex( )	Sets whether the parameter has a null value

actuate.xtabanalyzer.ParameterValue.getName

Syntax

```
string ParameterValue.getName( )
```

Returns the name for the parameter.

Returns String. The parameter name.

Example This example retrieves the parameter name:

```
function getParameterName(parametervalue){
   if (parametervalue){
      return parametervalue.getName( );
   }
   return null;
}
```

actuate.xtabanalyzer.ParameterValue.getValue

Syntax String[ ] Dimension.getValue( )

Returns the name for the ParameterValue.

Returns String or array of strings. The parameter value or values.

Example This example retrieves the parameter value:

```
function getParameterValue(parametervalue){
   if (parametervalue){
      return parametervalue.getValue( );
   }
   return null;
}
```

actuate.xtabanalyzer.ParameterValue.getValueIsNull

Syntax boolean ParameterValue.getValueIsNull( )

Returns whether the parameter value is null.

Returns Boolean. True indicates the parameter value is null.

Example This example switches whether the parameter value is null:

```
if (parametervalue){
   if (parametervalue.getValueIsNull){
      parametervalue.setValueIsNull(false);
   } else {
      parametervalue.setValueIsNull(true);
   }
}
```

actuate.xtabanalyzer.ParameterValue.setName

Syntax void ParameterValue.setName(string name)

Sets the parameter name.

Parameters **name**
String. The parameter name.

Example This example sets the parameter name:

```
function setParameterName(parametervalue, name){
   parametervalue.setName(name);
}
```

actuate.xtabanalyzer.ParameterValue.setValue

Syntax void ParameterValue.setValue(string[ ] value)

Sets the parameter value.

Parameters **value**
String. The parameter value.

Example This example sets the parameter value:

```
function setParameterValue(parametervalue, value){
   parametervalue.setValue(value);
}
```

actuate.xtabanalyzer.ParameterValue.setValueIsNull

Syntax void ParameterValue.setValueIsNull(boolean valueIsNull)

Sets the valueIsNull for the ParameterValue.

Parameters **valueIsNull**
Boolean. True switches the value to null. False disables the null value setting.

Example This example switches whether the parameter value is null:

```
if (parametervalue){
   if (parametervalue.getValueIsNull){
      parametervalue.setValueIsNull(false);
   } else {
      parametervalue.setValueIsNull(true);
   }
}
```

Class actuate.xtabanalyzer.Sorter

Description Defines a sort condition used to sort on a dimension level or measure.

Constructor

Syntax actuate.xtabanalyzer.Sorter(string levelName)

Constructs a new sorter object.

Function summary

Table 5-18 lists actuate.xtabanalyzer.Sorter functions.

Table 5-18 actuate.xtabanalyzer.Sorter functions

Function	Description
getKey( )	Returns the sort key
getLevelName( )	Returns the level name
getMember( )	Returns the sort member
isAscending( )	Returns the sort direction
setAscending( )	Sets ascending or descending sort
setKey( )	Sets the sort key
setLevelName( )	Sets the level name
setMember( )	Sets the sort member

actuate.xtabanalyzer.Sorter.getKey

Syntax string Sorter.getKey( )

Returns the sort key. This is the name of the measure or dimension level to sort the cross tab on.

Returns String. The key to sort on.

Example This example retrieves the sort key:

```
function getSortKey(sorter){
   if (sorter){
      return sorter.getKey( );
   }
   return null;
}
```

actuate.xtabanalyzer.Sorter.getLevelName

Syntax string Sorter.getLevelName( )

Returns dimension level to sort on.

Returns String. The name of a dimension level.

Example This example retrieves the level name associated with the sorter:

```
function getSortLevel(sorter){
   if (sorter){
      return sorter.getLevelName( );
   }
   return null;
}
```

actuate.xtabanalyzer.Sorter.getMember

Syntax actuate.xtabanalyzer.MemberValue Sorter.getMember( )

Returns the member value to sort on.

Returns actuate.xtabanalyzer.MemberValue object. A member value.

Example This example retrieves the sort member:

```
function getSortMember(sorter){
   if (sorter){
      return sorter.getMember( );
   }
   return null;
}
```

actuate.xtabanalyzer.Sorter.isAscending

Syntax boolean Sorter.isAscending( )

Returns the sort order.

Returns Boolean. True when the sorter is ascending and false in all other cases.

Example This example retrieves the level name that is associated with the sorter:

```
function ascending(sorter){
   if (sorter){
      return sorter.isAscending( );
   }
   return null;
}
```

actuate.xtabanalyzer.Sorter.setAscending

Syntax void Sorter.setAscending(boolean ascending)

Sets the sort order to ascending or descending.

Parameters **ascending**
Boolean. Set to true for ascending, set to false for descending.

Example This example swaps the sort direction:

```
sorter.setAscending(!(sorter.isAscending));
```

actuate.xtabanalyzer.Sorter.setKey

Syntax void Sorter.setSortKey(string sortKey)

Sets the key to sort on.

Parameters **sortKey**
String. The sort key.

Example This example sets the sort key:

```
function setSortKey(sorter,key){
   sorter.setKey(key);
}
```

actuate.xtabanalyzer.Sorter.setLevelName

Syntax void Sorter.setLevelName(string levelName)

Sets the dimension level name to sort on.

Parameters **levelName**
String. A dimension level name.

Example This example sets the level name to sort:

```
function setSortLevel(sorter,level){
   sorter.setLevelName(level);
}
```

actuate.xtabanalyzer.Sorter.setMember

Syntax void Sorter.setMember(actuate.xtabanalyzer.MemberValue member)

Sets the member value to sort on.

Parameters **member**
actuate.xtabanalyzer.MemberValue object. A member value.

Example This example sets the sort member:

```
function setSortMember(sorter,member){
   sorter.setMember(member);
}
```

Class actuate.xtabanalyzer.SubTotal

Description A SubTotal object.

Constructor

Syntax actuate.xtabanalyzer.SubTotal( )

Constructs a new SubTotal object.

Function summary

Table 5-19 lists actuate.xtabanalyzer.SubTotal functions.

Table 5-19 actuate.xtabanalyzer.SubTotal functions

Function	Description
addTotal( )	Add a total
getLevelName( )	Returns the full level name
getLocation( )	Returns the location
getTotals( )	Returns the totals array
getType( )	Returns the type string
setLevelName( )	Sets the full level name
setLocation( )	Sets the location
setTotals( )	Sets the totals array

actuate.xtabanalyzer.SubTotal.addTotal

Syntax void SubTotal.addTotal(actuate.xtabanalyzer.Total total)

Adds a total to the subtotal.

Parameters actuate.xtabanalyzer.Total. The total object being added.

Example This example uses addTotal( ) to create a subtotal:

```
function addSubTotal( ){
   var subTotal = new actuate.xtabanalyzer.SubTotal( );
   subTotal.setLevelName("year");
   subTotal.setLocation("after");
   var indexStr = "0;1;2;3;4";
   var indexs = indexsStr.split(";");
   var measureIndexs = [ ];
   for(var i = 0;i < indexs.length;i++){
      measureIndexs.push(parseInt(indexs[i]));
   }
   for( var i = 0; i < measureIndexs.length; i++){
      var total = new actuate.xtabanalyzer.Total( );
      total.setMeasureIndex(measureIndexs[i]);
      total.setAggregationFunction("SUM");
      total.setEnabled(true);
      subTotal.addTotal(total);
   }
      crosstab.setTotals(null,subTotal);
      crosstab.submit( );
}
```

actuate.xtabanalyzer.SubTotal.getLevelName

Syntax string SubTotal.getLevelName( )

Returns the level for the subtotal.

Returns String. The level name for the subtotal.

Example This example retrieves the level name from the subtotal:

```
function getLevelName(subTotal){
   if (subTotal){
      return subTotal.getLevelName( );
   }
   return null;
}
```

actuate.xtabanalyzer.SubTotal.getLocation

Syntax string SubTotal.getLocation( )

Returns the location name for the subtotal.

Returns String. The location name.

Example This example retrieves the level name from the subtotal:

```
function getLocation(subTotal){
   if (subTotal){
      return subTotal.getLocation( );
   }
   return null;
}
```

actuate.xtabanalyzer.SubTotal.getTotals

Syntax object[ ] SubTotal.getTotals( )

Returns the totals used to calculate the subtotal.

Returns actuate.xtabanalyzer.Total[ ]. An array of total objects.

Example This example retrieves the totals from a SubTotal object:

```
var totalsArray = [ ];
function getTotals(subTotal,totalsArray){
   totalsArray = subTotal.getTotals( );
}
```

actuate.xtabanalyzer.SubTotal.getType

Syntax string SubTotal.getType( )

Returns the type for the subtotal.

Returns String. The type for the subtotal.

Example This example retrieves the type from the subtotal:

```
function getLevelName(subTotal){
   if (subTotal){
      return subTotal.getType( );
   }
   return null;
}
```

actuate.xtabanalyzer.SubTotal.setLevelName

Syntax void SubTotal.setLevelName(string levelName)

Sets the level for the subtotal by name.

Parameters **levelName**
String. The level name.

Example This example sets the level name for a subtotal:

```
function subTotalLevel(subTotal,levelName){
   if(subTotal){
      subTotal.setLevelName(levelName);
   }
}
```

actuate.xtabanalyzer.SubTotal.setLocation

Syntax void SubTotal.setLocation(string location)

Sets the location for the subtotal.

Parameters **location**
String. The location. Value can be either before or after.

Example This example sets the location for a subtotal:

```
function subTotalLocation(subTotal,location){
   if(subTotal){
      subTotal.setLocation(location);
   }
}
```

actuate.xtabanalyzer.SubTotal.setTotals

Syntax void SubTotal.setTotals(actuate.xtabanalyzer.Total[ ] totals)

Sets the totals using an array.

Parameters **totals**
Array of actuate.xtabanalyzer.Total objects to add to the subtotal.

Example This example uses setTotals( ) to create a subtotal:

```
function addSubTotal( ){
   var subTotal = new actuate.xtabanalyzer.SubTotal( );
   subTotal.setLevelName("year");
   subTotal.setLocation("after");
   var indexStr = "0;1;2;3;4";
   var indexs = indexsStr.split(";");
   var count = indexs.length;
   var measureIndexs = [ ];
   for(var i = 0;i < count;i++){
      measureIndexs.push(parseInt(indexs[i]));
   }
   var totals = Array(count);
   for( var i = 0; i < measureIndexs.length; i++){
      var total = new actuate.xtabanalyzer.Total( );
      total.setMeasureIndex( measureIndexs[i] );
      total.setAggregationFunction( "SUM" );
      total.setEnabled(true);
      totals[i] = total;
   }
   subTotal.setTotals(totals);

   crosstab.setTotals( null, subTotal );
   crosstab.submit( );
}
```

Class actuate.xtabanalyzer.Total

Description A container for a total in the xtabanalyzer. Total handles numeric aggregation functions for a measure.

Constructor

Syntax actuate.xtabanalyzer.Total( )

The Total class is used to specify a cross tab total object.

Function summary

Table 5-20 lists actuate.xtabanalyzer.Total functions.

Table 5-20 actuate.xtabanalyzer.Total functions

Function	Description
getAggregationFunction( )	Returns the aggregation function name
getMeasureIndex( )	Returns the measure index
isEnabled( )	Returns whether or not the total is enabled
setAggregationFunction( )	Sets the aggregation function name
setEnabled( )	Sets the enabled flag
setMeasureIndex( )	Sets the index for the total

actuate.xtabanalyzer.Total.getAggregationFunction

Syntax string Total.getAggregationFunction( )

Returns the aggregation function for the total.

Returns String. An aggregation function.

Example This example changes the aggregation function:

```
function swapTotalAggregation(total){
   if (total.getAggregationFunction( ) == "SUM"){
      total.setAggregationFunction("COUNT");
   } else {
      total.setAggregationFunction("SUM");
   }
}
```

actuate.xtabanalyzer.Total.getMeasureIndex

Syntax integer Dimension.getMeasureIndex( )

Retrieves the measure index for the total.

Returns Integer. The measure index.

Example This example retrieves the measure index:

```
function getMeasureIndex(total){
   if (total){
      return total.getIndex( );
   }
   return null;
}
```

actuate.xtabanalyzer.Total.isEnabled

Syntax boolean Total.isEnabled( )

Returns whether the total is enabled.

Returns Boolean. True indicates this total is enabled.

Example This example enables and disables a total:

```
if (total){
   if (total.isEnabled){
      total.setEnabled(false);
   } else {
      total.setEnabled(true);
   }
}
```

actuate.xtabanalyzer.Total.setAggregationFunction

Syntax void Total.setAggregationFunction(string aggregationFunction)

Sets the aggregation function name.

Parameters **aggregationFunction**
String. The aggregation function name.

Example This example changes the aggregation function:

```
function swapTotalAggregation(total){
   if (total.getAggregationFunction( ) == "SUM"){
      total.setAggregationFunction("COUNT");
   } else {
      total.setAggregationFunction("SUM");
   }
}
```

actuate.xtabanalyzer.Total.setEnabled

Syntax void Total.setEnabled(boolean enabled)

Sets whether total is enabled or disabled.

Parameters **enabled**
Boolean. True if the total is enabled. False for disabled.

Example This example enables and disables a total:

```
if (total){
   if (total.isEnabled){
      total.setEnabled(false);
   } else {
      total.setEnabled(true);
   }
}
```

actuate.xtabanalyzer.Total.setMeasureIndex

Syntax void Total.setMeasureIndex(integer measureIndex)

Sets the measure index for the total.

Parameters **measureIndex**
Integer. The measure index for the total.

Example This example uses setMeasureIndex( ) to create a subtotal:

```
function addSubTotal( ){
   var subTotal = new actuate.xtabanalyzer.SubTotal( );
   subTotal.setLevelName("year");
   subTotal.setLocation("after");
   var indexStr = "0;1;2;3;4";
   var indexs = indexsStr.split(";");
   var count = indexs.length;
   var measureIndexs = [];
   for(var i = 0;i < count;i++){
      measureIndexs.push(parseInt(indexs[i]));
   }
   for( var i = 0; i < measureIndexs.length; i++) {
      var total = new actuate.xtabanalyzer.Total( );
      total.setMeasureIndex(measureIndexs[i]);
      total.setAggregationFunction("SUM");
      total.setEnabled(true);
      subTotal.addTotal(total);
   }
   crosstab.setTotals(null,subTotal);
   crosstab.submit( );
}
```

Class actuate.xtabanalyzer.UIOptions

Description Specifies feature availability for the Data Analyzer viewer.

Constructor

Syntax void actuate.xtabanalyzer.UIOptions( )

Generates a new UIOptions object to manage the features of the xtabanalyzer.

Function summary

Table 5-21 lists actuate.xtabanalyzer.UIOptions functions.

Table 5-21 actuate.xtabanalyzer.UIOptions functions

Function	Description
enableCrosstabView( )	Enables the cross tab layout view feature
enableCubeView( )	Enables the cube view feature
enableFilterSummaryView( )	Enables the filter summary view
enableToolBar( )	Enables the toolbar feature
enableToolbarHelp( )	Enables the toolbar help feature
enableToolbarSave( )	Enables the toolbar save feature
enableToolbarSaveDesign( )	Enables the toolbar save design feature
enableToolbarSaveDocument( )	Enables the toolbar save document feature
getFeatureMap( )	Returns a list of enabled and disabled features

actuate.xtabanalyzer.UIOptions.enableCrosstabView

Syntax void UIOptions.enableCrosstabView(boolean enabled)

Enables or disables the cross tab layout view.

Parameters **enabled**
Boolean. True enables this option.

Example This example enables or disables the cross tab view:

```
function setCrosstabView(flag){
   var uiOptions = new actuate.xtabanalyzer.UIOptions( );
   uiOptions.enableCrosstabView(flag);
   myXTabAnalyzer.setUIOptions(uiOptions);
}
```

actuate.xtabanalyzer.UIOptions.enableCubeView

Syntax void UIOptions.enableCubeView(boolean enabled)

Enables or disables the cube view.

Parameters **enabled**
Boolean. A value of true enables this option.

Example This example enables or disables the cube view:

```
function setCubeView(flag){
   var uiOptions = new actuate.xtabanalyzer.UIOptions( );
   uiOptions.enableCubeView(flag);
   myXTabAnalyzer.setUIOptions(uiOptions);
}
```

actuate.xtabanalyzer.UIOptions .enableFilterSummaryView

Syntax void UIOptions.enableFilterSummaryView(boolean enabled)

Enables or disables the filter summary view.

Parameters **enabled**
Boolean. A value of true enables this option.

Example This example enables or disables the filter summary view:

```
function setFilterSummary(flag) {
   var uiOptions = new actuate.xtabanalyzer.UIOptions( );
   uiOptions.enableFilterSummaryView(enabled);
   myXTabAnalyzer.setUIOptions(uiOptions);
}
```

actuate.xtabanalyzer.UIOptions.enableToolBar

Syntax void UIOptions.enableToolBar(boolean enabled)

Enables or disables the toolbar feature.

Parameters **enabled**
Boolean. A value of true enables this option.

Example This example enables or disables the toolbar:

```
function setToolbar(flag){
   var uiOptions = new actuate.xtabanalyzer.UIOptions( );
   uiOptions.enableToolBar(flag);
   myXTabAnalyzer.setUIOptions(uiOptions);
}
```

actuate.xtabanalyzer.UIOptions.enableToolbarHelp

Syntax void UIOptions.enableToolbarHelp(boolean enabled)

Enables or disables the toolbar help feature.

Parameters **enabled**
Boolean. A value of true enables this option.

Example This example enables or disables toolbar help:

```
function setToolbarHelp(flag){
   var uiOptions = new actuate.xtabanalyzer.UIOptions( );
   uiOptions.enableToolbarHelp(flag);
   myXTabAnalyzer.setUIOptions(uiOptions);
}
```

actuate.xtabanalyzer.UIOptions.enableToolbarSave

Syntax void UIOptions.enableToolbarSave(boolean enabled)

Enables or disables the toolbar save feature.

Parameters **enabled**
Boolean. A value of true enables this option.

Example This example enables or disables toolbar save:

```
function setToolbarSave(flag){
   var uiOptions = new actuate.xtabanalyzer.UIOptions( );
   uiOptions.enableToolbarSave(flag);
   myXTabAnalyzer.setUIOptions(uiOptions);
}
```

actuate.xtabanalyzer.UIOptions .enableToolbarSaveDesign

Syntax void UIOptions.enableToolbarSaveDesign(boolean enabled)

Enables or disables the toolbar save design feature.

Parameters **enabled**
Boolean. A value of true enables this option.

Example This example enables or disables toolbar save design:

```
function setToolbarSave(flag){
   var uiOptions = new actuate.xtabanalyzer.UIOptions( );
   uiOptions.enableToolbarSaveDesign(flag);
   myXTabAnalyzer.setUIOptions(uiOptions);
}
```

actuate.xtabanalyzer.UIOptions .enableToolbarSaveDocument

Syntax void UIOptions.enableToolbarSaveDocument(boolean enabled)

Enables or disables the toolbar save document feature.

Parameters **enabled**
Boolean. A value of true enables this option.

Example This example enables or disables toolbar save document:

```
function setToolbarSave(flag){
   var uiOptions = new actuate.xtabanalyzer.UIOptions( );
   uiOptions.enableToolbarSaveDocument(flag);
   myXTabAnalyzer.setUIOptions(uiOptions);
}
```

actuate.xtabanalyzer.UIOptions.getFeatureMap

Syntax Object UIOptions.getFeatureMap( )

Returns the features and their Boolean values as an associative array. This function makes the name of each feature an object property and sets the value of that property to the associated enabled Boolean value.

Returns Object. An associative array of string name and Boolean value pairs.

Example This example retrieves the feature map:

```
function retrieveFeatureMap( ){
   var uiOptions = new actuate.xtabanalyzer.UIOptions( );
   var features = uiOptions.getFeatureMap( );
   return features;
}
```

Index