SAP BusinessObjects Business Intelligence platform
Document Version: 4.2 Support Package 1 – 2015-12-14

SAP BusinessObjects BI Developer’s Guide for Web Intelligence and the BI Semantic Layer
# Content

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Document Version History. .......................................................... 4</td>
</tr>
<tr>
<td>2</td>
<td>About This Guide. ........................................................................... 6</td>
</tr>
<tr>
<td>3</td>
<td>Audience. ......................................................................................... 7</td>
</tr>
<tr>
<td>4</td>
<td>Conventions in This Guide. ............................................................. 8</td>
</tr>
<tr>
<td>5</td>
<td>Setting Up Web Intelligence as an Embedded Applet. ......................... 9</td>
</tr>
<tr>
<td>5.1</td>
<td>Prerequisites. .................................................................................. 9</td>
</tr>
<tr>
<td>5.2</td>
<td>Where to Find the Embedded Applet. ................................................ 10</td>
</tr>
<tr>
<td>5.3</td>
<td>Package Content. .............................................................................. 10</td>
</tr>
<tr>
<td>5.3.1</td>
<td>The Embedded Applet Utility. ............................................................ 10</td>
</tr>
<tr>
<td>5.3.2</td>
<td>The Embedded Applet Properties File. ............................................... 10</td>
</tr>
<tr>
<td>5.3.3</td>
<td>The webiApplet Folder Content. ....................................................... 11</td>
</tr>
<tr>
<td>5.3.4</td>
<td>The JSP and Image Sample Files. ..................................................... 11</td>
</tr>
<tr>
<td>5.4</td>
<td>Managing the Locale. ....................................................................... 12</td>
</tr>
<tr>
<td>5.5</td>
<td>Managing the Session. ..................................................................... 13</td>
</tr>
<tr>
<td>5.6</td>
<td>Deploying the Embedded Applet Sample. .......................................... 13</td>
</tr>
<tr>
<td>5.6.1</td>
<td>To Create the Deployment Root Directory. ......................................... 13</td>
</tr>
<tr>
<td>5.6.2</td>
<td>To Deploy the Embedded Applet. ..................................................... 13</td>
</tr>
<tr>
<td>5.6.3</td>
<td>To Test the Deployment. .................................................................. 14</td>
</tr>
<tr>
<td>6</td>
<td>Customizing Web Intelligence User Interfaces. ...................................... 15</td>
</tr>
<tr>
<td>6.1</td>
<td>Customizing Web Intelligence interface elements by user group and folders. 15</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Customization Interface. .................................................................. 15</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Customization Folders section. .......................................................... 16</td>
</tr>
<tr>
<td>6.1.3</td>
<td>User Interface Elements tab. ............................................................. 16</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Features tab. .................................................................................... 26</td>
</tr>
<tr>
<td>6.1.5</td>
<td>Customization rules. ......................................................................... 26</td>
</tr>
<tr>
<td>6.1.6</td>
<td>To customize the Web Intelligence interface appearance. .................. 27</td>
</tr>
<tr>
<td>6.2</td>
<td>Web Intelligence content alignment. .................................................. 27</td>
</tr>
<tr>
<td>6.2.1</td>
<td>To set content alignment for the Web Intelligence Applet interface. ...... 28</td>
</tr>
<tr>
<td>7</td>
<td>Customizing Web Intelligence with UI Extension Points. .................... 29</td>
</tr>
<tr>
<td>7.1</td>
<td>About the JavaScript APIs. ................................................................. 29</td>
</tr>
<tr>
<td>7.2</td>
<td>UI Extension Points Task Sequence. ................................................ 30</td>
</tr>
<tr>
<td>7.3</td>
<td>About the Extension Bundle. .............................................................. 30</td>
</tr>
<tr>
<td>7.4</td>
<td>Where to Find the Bundle Host. ........................................................ 31</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>7.5</td>
<td>Prerequisites</td>
</tr>
<tr>
<td>7.6</td>
<td>To Import the Bundle Host.</td>
</tr>
<tr>
<td>7.7</td>
<td>To Create an Extension Bundle.</td>
</tr>
<tr>
<td>7.8</td>
<td>To Declare the Extension Bundle Contribution.</td>
</tr>
<tr>
<td>7.9</td>
<td>To Implement the IExtension Interface.</td>
</tr>
<tr>
<td></td>
<td>getExtensionProperties.</td>
</tr>
<tr>
<td></td>
<td>getContribution.</td>
</tr>
<tr>
<td>7.10</td>
<td>To Append Contribution Files.</td>
</tr>
<tr>
<td>7.11</td>
<td>To Develop with the JavaScript APIs.</td>
</tr>
<tr>
<td>7.12</td>
<td>To Make the Extension Visible on the Interface.</td>
</tr>
<tr>
<td>7.13</td>
<td>To Test the Extension Bundle.</td>
</tr>
<tr>
<td>7.14</td>
<td>To Build the Extension Bundle.</td>
</tr>
<tr>
<td>7.15</td>
<td>To Deploy the Extension Bundle in Production.</td>
</tr>
<tr>
<td>7.16</td>
<td>About the Web Intelligence UI Extension Point Sample.</td>
</tr>
<tr>
<td></td>
<td>To Use the Extension Sample.</td>
</tr>
<tr>
<td>8</td>
<td>Creating Visualizations with a Custom Element Service.</td>
</tr>
<tr>
<td>8.1</td>
<td>About the Custom Element Service APIs.</td>
</tr>
<tr>
<td>8.2</td>
<td>Getting the Supported Formats.</td>
</tr>
<tr>
<td>8.3</td>
<td>Getting the Visualization Types.</td>
</tr>
<tr>
<td>8.4</td>
<td>Getting the Visualization Feed Definitions.</td>
</tr>
<tr>
<td>8.5</td>
<td>Rendering the Visualization.</td>
</tr>
<tr>
<td>9</td>
<td>Exposing Web Intelligence Features with REST Web Services.</td>
</tr>
<tr>
<td>10</td>
<td>Consuming BI Semantic Layer Universes with REST Web Services.</td>
</tr>
<tr>
<td>11</td>
<td>Developing Applications to Design and Administrate Universes.</td>
</tr>
<tr>
<td>12</td>
<td>Creating a Data Access Driver.</td>
</tr>
</tbody>
</table>
# Document Version History

The following table provides an overview of the most important document changes.

## Table 1:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP BusinessObjects BI Developer’s Guide for Web Intelligence and the BI Semantic Layer 4.2 Support Package 1</td>
<td>December 2015</td>
<td>The following updates have been made to the chapter Customizing Web Intelligence with UI Extension Points:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Completed prerequisites, see [Pre-requisites](page 31)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- JDK version 1.8.0 supported, see [To Import the Bundle Host](page 31)</td>
</tr>
<tr>
<td>SAP BusinessObjects BI Developer’s Guide for Web Intelligence and the BI Semantic Layer 4.2</td>
<td>November 2015</td>
<td>You can hide the Comments and Share Elements interface items from the Side Panel of SAP BusinessObjects Web Intelligence through customization in the CMC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You can develop your own visualizations for Web Intelligence through the Custom Element Service APIs.</td>
</tr>
<tr>
<td>SAP BusinessObjects BI Developer’s Guide for Web Intelligence and the BI Semantic Layer 4.1 Support Package 6</td>
<td>June 2015</td>
<td>The Embedded Applet has been modified to follow the SAP branding standard. A new image folder must be deployed along with the applet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You can hide the SAP Marketplace button from the Status Bar of SAP BusinessObjects Web Intelligence through customization in the CMC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This document now provides sections on the JavaScript APIs used to develop an extension, and a link to the API reference.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The chapter “Developing Applications to Design and Administrate Universes” now mentions the Universe Design Tool COM SDK.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This document now provides a chapter about creating drivers with the Driver Development Kit.</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Change</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
The SAP BusinessObjects BI platform 4.2 is the key foundation for your analytics applications. It comes with a comprehensive set of tools from which you can pick up the one that suits the technologies you are ready to use and your business objectives. To this end, the *SAP BusinessObjects BI Developer’s Guide for Web Intelligence and the BI Semantic Layer* is your new entry point to learn how to develop applications, using SDKs, samples, and extension framework, to enforce and take advantage of the Web Intelligence and BI Semantic Layer capabilities.

This guide provides information and references about:

- The Embedded Applet for Web Intelligence and how to deploy its sample in your portal
- How to customize the DHTML or Java Web Intelligence user interface via the Central Management Console
- How to extend Web Intelligence features using UI extension points
- How to create your visualizations with Web Intelligence custom elements
- How to use REST APIs to work with Web Intelligence documents and reports in non-SAP client tools
- How to use REST APIs to access universes and run queries in non-SAP client tools
- How to create, edit, secure and deploy universes with the BI Semantic Layer Java SDK
- How to create a JavaBean driver or an Open driver with the Driver Development Kit

This guide relates to the SAP BusinessObjects Business Intelligence platform 4.2 Support Package 1 release.
3 Audience

As it serves as an entry point to the Web Intelligence and BI Semantic Layer customization area, the BI Developer’s Guide is intended for various readers.

This guide is for you if:

- You are an SAP consultant who wants to help SAP partners and customers in their BI platform customization project
- You are an SAP partner who would like to provide customizations and extensions of Web Intelligence to your customer
- You are an SAP BusinessObjects administrator who wants to use Web Intelligence in their corporate portal
- You are a JavaScript developer responsible for developing extensions to Web Intelligence user interfaces
- You are a Java developer responsible for developing applications that perform creation, editing, and publication tasks on UNX and UNV universes
- You are a developer responsible for writing programs that access and consume the BI platform web services
- You are a developer responsible for developing data access drivers to help the BI platform to communicate with your company’s data sources
Conventions in This Guide

In this guide, the placeholder `<bip-install-dir>` is the install root path of the SAP BusinessObjects Bi platform. On Microsoft Windows, the default `<bip-install-dir>` stands for the `C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0` directory.

The placeholder `<tomcat-dir>` stands for the `C:\Program Files (x86)\SAP BusinessObjects\tomcat` directory.
5 Setting Up Web Intelligence as an Embedded Applet

You can set up the Web Intelligence Java applet to run as an applet embedded in your own portal, rather than running it from the BI launch pad. The Embedded Applet provides the same functionality as the Web Intelligence applet that is launched through the BI launch pad.

End-users will be able to open Web Intelligence documents stored in your BI platform repository directly from your portal. Using this applet, documents can be viewed, refreshed, printed, and saved locally as snapshots. No refresh will be available for documents saved locally.

A JSP sample demonstrating how to include the Embedded Applet in your own web application is also provided.

⚠️ Restriction
- With the Embedded Applet, you cannot use the following services:
  - Scheduling reports
  - Checking the Document History
  - Using the ‘Send To’ function
- You cannot change the drill options managed via the BI launch pad Web Intelligence preferences.
- You cannot set a language different to your portal language. The Embedded Applet accepts the Language parameter, but your deployment will have to pass it to the applet. This is not exposed to end-users.

5.1 Prerequisites

Software Requirements

- The SAP BusinessObjects BI platform servers
- Apache Tomcat web application server
- Java SE Runtime Environment 7

User Rights

In the Embedded Applet, the user rights are managed in the same way as in the BI launch pad version of the applet: the CMC administrator assigns user rights and authorizations, and these settings are applied when the user launches the applet and logins to the session. If you use your own login method, then all users will be able to perform all operations.

To set the user rights for the Embedded Applet, you must have:
• Access to the Central Management Console (CMC)
• The authorization to edit the settings of the Web Intelligence Adaptive Processing Server
• The rights to manage the portal

5.2 Where to Find the Embedded Applet

The Web Intelligence Embedded Applet is a ZIP file installed by default with the SAP BusinessObjects BI platform servers. It is located in `<bip-install-dir>\Samples\webi\EmbeddedApplet.zip`.

5.3 Package Content

The `EmbeddedApplet.zip` archive file contains the following series of folders:

- js
- jsp
- lib
- sample
- webiApplet

Table 2:

<table>
<thead>
<tr>
<th>Folder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>js</td>
<td>The Embedded Applet utility</td>
</tr>
<tr>
<td>jsp</td>
<td>The configuration files of the Embedded Applet</td>
</tr>
<tr>
<td>lib</td>
<td>The list of mandatory JAR files to make the Embedded Applet work</td>
</tr>
<tr>
<td>sample</td>
<td>The list of JSP and image sample files to create and manage a user session to the CMS</td>
</tr>
<tr>
<td>webiApplet</td>
<td>The Embedded Applet resources</td>
</tr>
</tbody>
</table>

5.3.1 The Embedded Applet Utility

The `applet_util.js` file helps the embedded applet to detect the web browser in which the applet is launched, the operating system of the machine, and the Java version.

5.3.2 The Embedded Applet Properties File

The `embeddedapplet.properties` file provides the necessary values for the configuration of the applet, which are used in the `appletpopup.jsp` file.
The following table lists the property description and values that configure the applet sample.

Table 3:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>portalroot</td>
<td>The end of the portal URL</td>
<td>/BOE/portal/1303180624</td>
</tr>
<tr>
<td>portal_port</td>
<td>The port of the portal URL</td>
<td>8080</td>
</tr>
<tr>
<td>help_url</td>
<td>The end of the help URL</td>
<td>/AnalyticalReporting</td>
</tr>
<tr>
<td>gateway_url</td>
<td>The end of the gateway URL</td>
<td>/rebean3ws/services/Gateway</td>
</tr>
<tr>
<td>applet_url</td>
<td>URL of the applet on the web application server</td>
<td>/webiApplet</td>
</tr>
</tbody>
</table>

5.3.3 The webiApplet Folder Content

The webiApplet folder mainly contains the Embedded Applet resources:

- The Embedded Applet JAR file (webiapplet.jar)
- JAR files to manage applet localization in different languages
- Splash screens used when the applet is loading

5.3.4 The JSP and Image Sample Files

The Embedded Applet provides a series of JSP and image sample files that allow you to login to the applet in your portal. The image files are stored in the img subfolder.

Note

These files are only samples. You do not need to use them to create your own application.

Table 4:

<table>
<thead>
<tr>
<th>JSP File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index.jsp</td>
<td>The sample index file that represents the portal in which you want to use the Embedded Applet</td>
</tr>
<tr>
<td>loginForm.jsp</td>
<td>The form that you use to get login information. The form calls the login.jsp file.</td>
</tr>
<tr>
<td>login.jsp</td>
<td>The JSP file used to login the end-user to the portal and create an IEnterpriseSession. It directs to the appletpopup.jsp file.</td>
</tr>
<tr>
<td>appletpopup.jsp</td>
<td>The JSP file used to load the Embedded Applet in the portal with the session created in the login.jsp file.</td>
</tr>
<tr>
<td>closeSession.jsp</td>
<td>The file used to close the session. The login form page is displayed again after the session is closed.</td>
</tr>
</tbody>
</table>
Table 5:

<table>
<thead>
<tr>
<th>Image File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>center_normal_logo.gif</td>
<td>Background image for the Logon button. It is used in the LoginForm.jsp file.</td>
</tr>
<tr>
<td>uhBodyTextureTop.png</td>
<td>Portal background image used in the index.jsp file</td>
</tr>
<tr>
<td>uhLogo2.png</td>
<td>SAP logo used in the index.jsp file</td>
</tr>
</tbody>
</table>

**Embedded Applet CallBack**

The `appletpopup.jsp` file calls the `AppletCallBack_updateDocumentTitle` function when the end-user is performing any of the following actions on a Web Intelligence document that could lead to a change to the current document name:

- Create
- Open
- Save as
- Close

This JavaScript function must be present in the parent of the iFrame which has the `appletpopup.jsp` file on it. It is implemented in the `index.jsp` file.

**IEnterpriseSession Attributes**

One of the roles of the `login.jsp` file is to provide the attributes of the IEnterpriseSession object with correct values. This enables the creation of the user session on the CMS.

Table 6:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebIEmbeddedApplet_EnterpriseSession</td>
<td>The IEnterpriseSession object</td>
</tr>
<tr>
<td>WebIEmbeddedApplet_CMSName</td>
<td>The name of the CMS which the applet will connect to</td>
</tr>
<tr>
<td>WebIEmbeddedApplet_PortalPort</td>
<td>The sample gateway port used to create the gateway URL</td>
</tr>
<tr>
<td>WebIEmbeddedApplet_ProductLanguage</td>
<td>The locale to use for the applet localization</td>
</tr>
</tbody>
</table>

**5.4 Managing the Locale**

The locale is not the one managed through the end-user properties in the BI launch pad. You must set the locale through the `WebIEmbeddedApplet_ProductLanguage` attribute.
5.5 Managing the Session

The Embedded Applet does not manage the lifecycle of the IEnterpriseSession object. So your portal implementation should take care of the creation and deletion of the session.

5.6 Deploying the Embedded Applet Sample

This section describes the deployment of the applet sample provided in the embeddedapplet.zip file.

5.6.1 To Create the Deployment Root Directory

Before you deploy the Embedded Applet, you should configure the web application server where you want to run the Embedded Applet, for example Apache Tomcat.

1. Stop Apache Tomcat.
2. Create the EmbeddedAppletTest folder as the root directory for the applet sample deployment on the web application server, under <tomcat-dir>/webapps.

   If your Apache Tomcat is brand new, the directories under <tomcat-dir>/webapps should look like the following:
   ○ docs
   ○ EmbeddedAppletTest
   ○ examples
   ○ host-manager
   ○ manager
   ○ ROOT

5.6.2 To Deploy the Embedded Applet

You must make sure that JAVA_HOME environment variable is set correctly, for example:

JAVA_HOME= C:\Program Files (x86)\Java\jre7

1. Copy to <tomcat-dir>/webapps/EmbeddedAppletTest:
   ○ The webiApplet folder
   ○ The js folder
   ○ The sample folder content, including the img subfolder
   ○ The appletpopup.jsp file from the jsp folder of the ZIP file
2. Create the <tomcat-dir>/webapps/EmbeddedAppletTest/WEB-INF/classes directory.
3. Copy all the properties files from the jsp folder of the ZIP file to this folder (embeddedapplet.properties, webi_applet_jars.properties, and webi_applet_lang_jars.properties).

4. Copy the lib folder to <tomcat-dir>/webapps/EmbeddedAppletTest/WEB-INF.

5. Start Apache Tomcat.

The final folder hierarchy should contain the following folders and files:

- img
- js
- webApplet
- WEB-INF
- appletpopup.jsp
- closesession.jsp
- index.jsp
- login.jsp
- loginForm.jsp

The WEB-INF folder should contain the following subfolders:

- classes
- lib

5.6.3 To Test the Deployment

1. Open an Internet Explorer browser window on the same machine where you have deployed the Embedded Applet.

2. Go to http://localhost:8080/EmbeddedAppletTest/.
You should see the login form displayed on a web page called Web Intelligence Embedded Applet.

3. Login to the portal as an end-user can do.
This starts SAP BusinessObjects Web Intelligence as an applet embedded in the portal.

4. Perform any action on a Web Intelligence document, save your modifications and close the document.

5. Click Logoff in the upper-right corner of the web page to close SAP BusinessObjects Web Intelligence and the user session.
6 Customizing Web Intelligence User Interfaces

You can simplify the appearance of the DHTML and Java interfaces of SAP BusinessObjects Web Intelligence by hiding some functionalities through the CMC.

6.1 Customizing Web Intelligence interface elements by user group and folders

In the CMC, you can customize the appearance of Web Intelligence interface elements for a user, depending on the user group they belong to and the folders containing Web Intelligence documents. For example, the entire toolbar or specific items in a toolbar, and customize access to specific document modes. You can also customize Web Intelligence by enabling extensions.

All interface elements appear by default. If you do not want specific elements to appear, you deselect them in the CMC. All extension points are disabled by default. If you want to make them available to users, you enable them in the CMC.

Note

- The customization and the enabled extension points are applicable to all Web Intelligence application clients: HTML, Java Applet, and Rich Client.
- It might happen that the customization and enabled extension points do not work on Web Intelligence Rich Client because of proxy or DNS configuration. To solve this problem, log in to the CMC with the IP address of the server instead of the server name when you customize Web Intelligence. This IP address will be used as a reference during customization.

6.1.1 Customization Interface

The Customization section contains the following section and tabs:

- Customized folders section
  On this section, you can select folders containing Web Intelligence documents for which you want to customize user interface and enable extensions.

- User Interface Elements tab
  On this tab, you can select individual interface elements to hide, such as a toolbar or tab, or their sub elements; for example a button command.

- Features tab
  On this tab, you can choose to hide all user interface elements related to a function; for example, Refresh.

- Extensions tab
On this tab, you can enable Web Intelligence user interface extensions that you have created and deployed in your installation.

### 6.1.2 Customization Folders section

The *Customization folders* section contains a folder named Default Folders, which is used to define default customization. You can choose folders for which you want to apply customization by clicking the *Add Folder* button. To avoid redefining the same customization for other folders, you can copy customization from one folder to another by using *Duplicate Customization* and *Paste Customization* options from the drop-down list. If you want to remove customization for a specific folder, you need to remove the folder that you have added by selecting the *Remove folder* option from the drop-down list.

**Note**
You cannot remove Default Folders from the *Customization folders* section.

### 6.1.3 User Interface Elements tab

Some of the interface elements that you can customize are identified in the diagrams in the subsequent subsections. Use the following table to identify the element items in the diagrams.

<table>
<thead>
<tr>
<th>User Interface Element Item</th>
<th>Sub-element item</th>
<th>Description</th>
<th>Number in diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splash screen</td>
<td></td>
<td>The screen that appears when a user opens Web Intelligence.</td>
<td></td>
</tr>
<tr>
<td>Application Contextual Menu</td>
<td></td>
<td>The menu that appears when a user right-clicks in the Web Intelligence screen.</td>
<td>1</td>
</tr>
<tr>
<td>Application mode</td>
<td></td>
<td>The application mode change option in the Application Contextual Menu.</td>
<td>1a</td>
</tr>
<tr>
<td>Filter Bar</td>
<td></td>
<td>The Filter Bar option in the Application Contextual Menu.</td>
<td>1b</td>
</tr>
<tr>
<td>Outline</td>
<td></td>
<td>The Outline option in the Application Contextual Menu.</td>
<td>1c</td>
</tr>
<tr>
<td>Formula Bar</td>
<td></td>
<td>The Formula Bar option in the Application Contextual Menu.</td>
<td>1d</td>
</tr>
<tr>
<td>Side Panel</td>
<td></td>
<td>The Side Panel option in the Application Contextual Menu.</td>
<td>1e</td>
</tr>
<tr>
<td>Report Tabs</td>
<td></td>
<td>The Report Tabs option in the Application Contextual Menu.</td>
<td>1f</td>
</tr>
<tr>
<td>Status Bar</td>
<td></td>
<td>The Status Bar option in the Application Contextual Menu.</td>
<td>1g</td>
</tr>
<tr>
<td>Side Panel</td>
<td></td>
<td>The Side Panel next to the report panel that allows users access to various information tabs.</td>
<td>2</td>
</tr>
<tr>
<td>Document Summary</td>
<td></td>
<td>The Document Summary tab in the Side Panel.</td>
<td>2a</td>
</tr>
<tr>
<td>User Interface</td>
<td>Sub-element item</td>
<td>Description</td>
<td>Number in diagram</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
<td>-------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Navigation Map</td>
<td>The Navigation Map (called Report Map in the HTML interface) tab in the Side Panel.</td>
<td>2b</td>
<td></td>
</tr>
<tr>
<td>Input Controls</td>
<td>The Input Controls tab in the Side Panel.</td>
<td>2c</td>
<td></td>
</tr>
<tr>
<td>User Prompt Input</td>
<td>The User Prompt Input tab in the Side Panel.</td>
<td>2d</td>
<td></td>
</tr>
<tr>
<td>Available Objects</td>
<td>The Available Objects tab in the Side Panel.</td>
<td>2e</td>
<td></td>
</tr>
<tr>
<td>Document Structure and Filters</td>
<td>The Document Structure and Filters tab in the Side Panel.</td>
<td>2f</td>
<td></td>
</tr>
<tr>
<td>Shared Elements</td>
<td>The Shared Elements tab in the Side Panel</td>
<td>Not shown</td>
<td></td>
</tr>
<tr>
<td>Comments</td>
<td>The Comments tabs ( ) in the Side Panel</td>
<td>Not shown</td>
<td></td>
</tr>
<tr>
<td>Web Service Publisher</td>
<td>The Web Service Publisher tab in the Side Panel.</td>
<td>2i</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>The Data tab in the Side Panel.</td>
<td>2k</td>
<td></td>
</tr>
<tr>
<td>Status Bar</td>
<td>The Status Bar, where the user sees information on document action statuses and can perform zoom, page navigation, and formula bar activation tasks.</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SAP Marketplace</td>
<td>The SAP Marketplace button ( ) in the Status Bar.</td>
<td>Not shown</td>
<td></td>
</tr>
<tr>
<td>Report dropdown list</td>
<td>The Report dropdown list in the Status Bar.</td>
<td>3a</td>
<td></td>
</tr>
<tr>
<td>Printing status icon</td>
<td>The Printing status icon list in the Status Bar.</td>
<td>3b</td>
<td></td>
</tr>
<tr>
<td>Shared Elements</td>
<td>The Shared Elements selection icon ( ) in the Status Bar.</td>
<td>Not shown</td>
<td></td>
</tr>
<tr>
<td>Track data changes</td>
<td>The Track data changes status in the Status Bar.</td>
<td>3c</td>
<td></td>
</tr>
<tr>
<td>Page Navigation</td>
<td>The Page Navigation bar in the Status Bar.</td>
<td>3d</td>
<td></td>
</tr>
<tr>
<td>Pagination Mode</td>
<td>The Pagination Mode buttons in the Status Bar.</td>
<td>3e</td>
<td></td>
</tr>
<tr>
<td>Zoom List</td>
<td>The Zoom percentage dropdown list in the Status Bar.</td>
<td>3f</td>
<td></td>
</tr>
<tr>
<td>Zoom Slider</td>
<td>The Zoom Slider bar in the Status Bar.</td>
<td>3g</td>
<td></td>
</tr>
<tr>
<td>Workspace Status</td>
<td>The Workspace Status indicator in the Status Bar.</td>
<td>Not shown</td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td>The Workspace Status indicator ( ) appears between the Zoom Slider and the Last Refresh Date if a problem occurs in the workspace.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last Refresh Date</td>
<td>The document refresh date in the Status Bar.</td>
<td>3i</td>
<td></td>
</tr>
<tr>
<td>Connection Status</td>
<td>The Web Intelligence Rich Client Connection Status in the Status Bar.</td>
<td>3j</td>
<td></td>
</tr>
<tr>
<td>Report Zone</td>
<td>The report zone in Web Intelligence.</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Report Tabs</td>
<td>The report tabs in the report zone.</td>
<td>4a</td>
<td></td>
</tr>
<tr>
<td>User Interface Element Item</td>
<td>Sub-element item</td>
<td>Description</td>
<td>Number in diagram</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------</td>
<td>-------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Bi-directional Page Scrolling</td>
<td></td>
<td>The bi-directional page scrolling feature in the lower corner of the report zone page.</td>
<td>4b</td>
</tr>
<tr>
<td>Formula Bar</td>
<td></td>
<td>The formula bar at the top of the report zone.</td>
<td>4c</td>
</tr>
<tr>
<td>Reading Mode Toolbar</td>
<td></td>
<td>The toolbars displayed in Reading mode.</td>
<td>5</td>
</tr>
<tr>
<td>Web Intelligence dropdown list</td>
<td></td>
<td>The Web Intelligence dropdown list in the Reading mode.</td>
<td>5a</td>
</tr>
<tr>
<td>File Group</td>
<td></td>
<td>The File Group toolbar in the Reading mode.</td>
<td>5b</td>
</tr>
<tr>
<td>Standard Actions Group</td>
<td></td>
<td>The Standard Actions Group toolbar in the Reading mode.</td>
<td>5c</td>
</tr>
<tr>
<td>Analysis Group</td>
<td></td>
<td>The Analysis Group toolbar in the Reading mode.</td>
<td>5d</td>
</tr>
<tr>
<td>Design Mode Toolbar</td>
<td></td>
<td>The toolbars and tabs displayed in the Design mode.</td>
<td>6</td>
</tr>
<tr>
<td>File tab</td>
<td></td>
<td>The File tab in the Reading mode.</td>
<td>6a</td>
</tr>
<tr>
<td>Properties tab</td>
<td></td>
<td>The Properties tab in the Reading mode.</td>
<td>6b</td>
</tr>
<tr>
<td>Standard Actions Group</td>
<td></td>
<td>The Standard Actions Group toolbar in the Reading mode.</td>
<td>6c</td>
</tr>
<tr>
<td>Report Elements tab</td>
<td></td>
<td>The Report Elements tab in the Reading mode.</td>
<td>6d</td>
</tr>
<tr>
<td>Format tab</td>
<td></td>
<td>The Format tab in the Reading mode.</td>
<td>6e</td>
</tr>
<tr>
<td>Data Access tab</td>
<td></td>
<td>The Data Access tab in the Reading mode.</td>
<td>6f</td>
</tr>
<tr>
<td>Analysis tab</td>
<td></td>
<td>The Analysis tab in the Reading mode.</td>
<td>6g</td>
</tr>
<tr>
<td>Page Setup tab</td>
<td></td>
<td>The Page Setup tab in the Reading mode.</td>
<td>6h</td>
</tr>
<tr>
<td>Initial Toolbar</td>
<td></td>
<td>The initial toolbars that appear when a user opens the Web Intelligence application.</td>
<td>7</td>
</tr>
<tr>
<td>Web Intelligence dropdown list</td>
<td></td>
<td>The Web Intelligence dropdown list in the initial toolbar.</td>
<td>7a</td>
</tr>
<tr>
<td>File Group</td>
<td></td>
<td>The File Group toolbar in the initial toolbar.</td>
<td>7b</td>
</tr>
<tr>
<td>Application Control Toolbar</td>
<td></td>
<td>The Application Control Toolbar that appears in the upper toolbar of Web Intelligence.</td>
<td>8</td>
</tr>
<tr>
<td>Application mode buttons</td>
<td></td>
<td>The application mode buttons (Reading, Design, and Data) in the upper toolbar of Web Intelligence.</td>
<td>8a</td>
</tr>
<tr>
<td>Tools</td>
<td></td>
<td>The Tools icon in the upper toolbar of Web Intelligence.</td>
<td>8b</td>
</tr>
<tr>
<td>Help</td>
<td></td>
<td>The Help icon in the upper toolbar of Web Intelligence.</td>
<td>8c</td>
</tr>
<tr>
<td>Close</td>
<td></td>
<td>The Close icon in the upper toolbar of Web Intelligence.</td>
<td>8d</td>
</tr>
<tr>
<td>Shortcuts</td>
<td></td>
<td>Keyboard shortcuts; for example, ( \text{CTRL + N} ) or ( \text{CTRL + S} ). Not shown</td>
<td></td>
</tr>
</tbody>
</table>
Splash Screen

In the CMC Users and Groups Customization, you can choose to hide the Web Intelligence splash screen that appears. The following image shows the splash screen that appears by default when a user opens Web Intelligence.

Application Contextual Menu

The following diagrams show the items that can be hidden in the right-click contextual menu.
Initial Toolbar

The following diagrams show the items that can be hidden in the toolbars that appear in Web Intelligence when no document is open.

**Web Intelligence Applet interface**

**Web Intelligence Rich Client**

![Initial Toolbar (English example)](image)

![Web Intelligence HTML interface](image)

Reading Mode

The following diagrams show the items that can be hidden in the Web Intelligence Reading mode.
Web Intelligence Applet interface

Web Intelligence Rich Client
Design Mode

The following diagrams show the items that can be hidden in the Web Intelligence Design mode.
Data Mode

The following diagrams show the items that can be hidden in the Web Intelligence Data mode.
Figure 6: Data Mode (English example)
6.1.4 Features tab

You can choose on this Feature tab to disable some functions at once, without having to disable them one after the other.

Table 8:

<table>
<thead>
<tr>
<th>Feature Item</th>
<th>Description</th>
<th>Affects the following interface items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh</td>
<td>Users can refresh documents to update the data from the data source.</td>
<td>The Refresh button in the Standard Action Group toolbar used in the Reading and Design modes.</td>
</tr>
</tbody>
</table>
| Drill        | Users can drill up and down on data in a document. | The Drill button available in the following locations:  
  ● The Analysis Group toolbar in the Reading mode.  
  ● The Interact subtab under the Analysis tab in Design mode. |
| Reading Mode | Users can view a document in Reading mode. | The Reading button in the following locations:  
  ● Application Contextual Menu  
  ● Application Control Toolbar |
| Design Mode  | Users can view a document in Design mode. | The Design button in the following locations:  
  ● Application Contextual Menu  
  ● Application Control Toolbar |
| Data Mode    | Users can view a document in Data mode. | The Data button in the following locations:  
  ● Application Contextual Menu  
  ● Application Control Toolbar |
| Comments     | Users can add comments to reports. | The Comments interface items |
| Shared Elements | Users can reuse reports parts multiple times. | The Shared Elements interface items |

6.1.5 Customization rules

The following rules are used to define customizations to apply to a user:

- If the user belongs to different groups, only the customization defined to the group whose ID is lower applies. The customization defined for the other groups containing the user does not apply.
- For nested folder structure, the immediate parent folder of the document that has been added in the list of customized folders define customizations for the document for user interface elements, features, and extensions.
- The customization defined for Default Folders apply for the documents stored in Personal Documents and Inboxes, and for documents for which the parent folder is not customized.
- The customization defined for user interface elements have priority over customization defined for features as features is only a shortcut to enable all user interface elements.
6.1.6 To customize the Web Intelligence interface appearance

You can customize the appearance of the Web Intelligence user interface by hiding menu items, sub-items, and features for a selected User Group and document folder.

1. Log into the CMC as an Administrator.
2. From the Organize list, select Users and Groups.
3. In the Group Hierarchy list, select a user group.
4. In the Actions list, select Customization.
5. In the Customization folders section, click Add Folder to add the folder containing Web Intelligence documents for which you want to apply customization for the selected user group.
6. Do one of the following:
   ○ To have items hidden in Web Intelligence, deselect them in the User Interface Elements or Features tab.
   ○ To have hidden items appear in Web Intelligence, select them in the User Interface Elements or Features tab.
7. Click Save & Close.

When you save the customization, all users of the selected group will see these changes the next time they log on to BI launch pad and open Web Intelligence.

i Note
We recommend that you log on to BI launch pad as a user from the group you have just customized, start Web Intelligence, and verify that the interface corresponds to your customization settings.

6.2 Web Intelligence content alignment

Choose the way document content will be aligned (left-to-right or right-to-left) when users create Web Intelligence documents.

For the Web Intelligence Applet interface, you can set the content alignment in the CMC. Choose from these options:

- Right-to-Left only when both the Preferred viewing and Product locales are set to Right-to-Left languages (the default option)
- Right-to-Left or Left-to-Right depending on the user’s Preferred viewing locale
- Always Right-to-Left
- Always Left-to-Right

i Note
The content alignment setting applies to all users.
For the Web Intelligence Rich Client interface, the content alignment is determined by the locales set in the BI launch pad preferences:

- The system uses right-to-left alignment only when both the Preferred Viewing Locale and Product Locale are set to right-to-left languages.
- In all other cases, the content alignment is left-to-right.

**Note**
For information about how to set locales, see the *Business Intelligence Launch Pad User Guide*.

**Note**
Content alignment applies only at document creation time, and does not affect existing documents.

### 6.2.1 To set content alignment for the Web Intelligence Applet interface

Set content alignment for the Web Intelligence Applet interface.

1. Log into the CMC as an Administrator.
2. From the Manage list, select *Applications*.
3. Select *Web Intelligence*.
4. Click **Manage > Properties**.
5. Scroll down to the *Content Alignment for New Documents* section and select the appropriate option.
7 Customizing Web Intelligence with UI Extension Points

You can customize the DHTML and Java interfaces of SAP BusinessObjects Web Intelligence and the Web Intelligence Rich Client interface using extensions on Microsoft Windows.

An extension contributes to the user interface by adding one or more UIElements, for example a button in the left-side pane, an icon button, a drop-down list, or a text field. Extensions provide end-users with advanced functionality to interact with Web Intelligence documents and reports via this UIElement.

7.1 About the JavaScript APIs

You make the extension interact with the application by using the Web Intelligence Application and Service JavaScript APIs.

The Web Intelligence Application JavaScript functions allow you to set up the Web Intelligence DHTML or Java interface for interaction with documents and reports. You can for example listen to and dispatch events, update the DHTML client context, display wait cursor and dialog boxes. The Web Intelligence Service JavaScript functions allow you to develop the functionality provided by the extension.

To do so, you create a target page such as an HTML or a JSP page that includes the appropriate JavaScript API files.

**Note**

You can also use these functions with Web Intelligence Rich Client in Connected mode only, because the extension is installed on the BI platform server.

<table>
<thead>
<tr>
<th>Documentation Deliverable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP BusinessObjects Web Intelligence UI Extension Points JavaScript API reference</td>
<td>The official JavaScript functions reference</td>
</tr>
</tbody>
</table>

**Related Information**

- To Append Contribution Files [page 35]
- To Develop with the JavaScript APIs [page 35]
7.2 UI Extension Points Task Sequence

Here is the series of tasks to perform for customizing SAP BusinessObjects Web Intelligence with an extension:

1. Build your development environment.
2. Create the extension point.
3. Declare the extension to the extension point.
4. Implement the extension using the IExtension interface.
5. Create a function for your extension with the help of the Javascript APIs.
6. Test the extension on your development environment.
7. Build the extension JAR file.
8. Deploy the JAR file on the BI platform server and Apache Tomcat server of your production environment.
9. In the customization panel of the CMC, select the extension that you want to make available to specific users, groups of users, or folders.

7.3 About the Extension Bundle

The extension bundle is a fragment linked to the webpath.AnalyticalReporting bundle host. In the BI platform OSGI framework, the bundle host and its fragment bundles such as language packs and extensions are merged. To avoid overriding files, respect the following organization of the extension bundle folders:

```
web
  webiApplet
  webIDHTML
  viewer
  ...
  extension
    <Provider>
      <ExtensionName>
    WEB-INF
    lib
```

Table 10:

<table>
<thead>
<tr>
<th>Folder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>web</td>
<td>Top folder of all webpath bundles</td>
</tr>
<tr>
<td>extension</td>
<td>Subfolders that belong to an extension bundle</td>
</tr>
<tr>
<td>&lt;Provider&gt;</td>
<td>The name of bundle provider. The provider can be a vendor or a company name.</td>
</tr>
<tr>
<td>&lt;ExtensionName&gt;</td>
<td>The extension. It should reflect the functionality provided by the extension.</td>
</tr>
<tr>
<td>WEB-INF/lib</td>
<td>The folder where you can deploy libraries. Mainly JAR files.</td>
</tr>
</tbody>
</table>

The web/extension/<Provider>/<ExtensionName> root path of the extension bundle is the folder from where the resource files are deployed. The bundle resources can be for example HTML, JavaScript, JSP, or images files. All URLs to resources must be related to the root path.
7.4 Where to Find the Bundle Host

The `webpath.AnalyticalReporting` bundle host is installed with SAP BusinessObjects BI platform server at `C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\warfiles\webapps\BOE\WEB-INF\eclipse\plugins`.

7.5 Prerequisites

Before creating an extension:

- You must have the SAP BusinessObjects BI platform servers installed on your Microsoft Windows development machine.
- SAP recommends that you build your development environment on Eclipse 3.6 or higher and using Apache Tomcat version 7.0.41 or higher.
- SAP also recommends to set `sapjvm` as the only installed JRE. It is located in `<bip-install-dir>\win64_x64\sapjvm`.
- The compilation level must be 1.6.

Note

To learn how to build your development environment, see the Creating Extensions to SAP BusinessObjects Web Intelligence document on the SAP Community Network.

7.6 To Import the Bundle Host

The `webpath.AnalyticalReporting` bundle host is the master bundle, which the extension bundles will refer to.

1. Open Eclipse, select `Window` > `Open Perspective` > `Other` and select `Plug-in Development` as your work perspective.
2. Select `Window` > `Preferences` to set the Java/Installed runtimes to JDK 1.8.0.
3. Select `File` > `Import`.
4. In the `Import` dialog box, select `Plug-in Development` > `Plug-ins and Fragments` and click `Next`. 

Remember

Add folders and JAR files to the CLASSPATH so that they can be found and loaded by the class loader.
5. Select the **Directory** option in **Import From**. The directory must be `C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\warfiles\webapps\BOE\WEB-INF\eclipse\plugins`.

6. Select `com.businessobjects.webpath.AnalyticalReporting (1.0.0)` bundle from the list of plug-ins and fragments, click **Add** and **Finish**.

### 7.7 To Create an Extension Bundle

1. Select **File > New > Project**

2. In the **New Project** dialog box, select **Plug-in Development > Fragment Project** and click **Next**.

3. Fill in the **New Fragment Project** dialog box as below and click **Next**.
   a. Enter the project name: `com.businessobjects.webpath.AnalyticalReporting_<Provider>_ <ExtensionName>`. `<Provider>` and `<ExtensionName>` must be the same as those provided by the extension attributes. They must contain only alphanumeric characters. SAP does not recommend that you use space characters, but underscores instead.
   b. Select **Use default location**.
   c. Select **Create a Java project** with (Source folder: src, and Output folder: bin).
   d. Select the **Equinox OSGi framework** and click **Next**. The Equinox Registry contains a list of all extensions and extension points belonging to the loaded and resolved bundles during the Apache Tomcat startup.

4. Fill in the **Fragment Content** as below:
   a. ID: `com.businessobjects.webpath.AnalyticalReporting_<Provider>_ <ExtensionName>`
   b. Version: 1.0.0
   c. Name: `<ExtensionName>`
   d. Provider: `<Provider>`
   e. Execution Environment: **JavaSE-1.6**
   f. Plug-in ID: `com.businessobjects.webpath.AnalyticalReporting`

5. Click **Finish**.

### 7.8 To Declare the Extension Bundle Contribution

The extension bundle is created to contribute to the `webpath.AnalyticalReporting` bundle host. You declare the extension in the **MANIFEST.MF** file.

1. Open the **MANIFEST.MF** file and select the **Overview** tab.
2. Click **Extensions** in **Extension/Extension Point Content** area.
3. In **Extensions** tab, click **Add** to display the list of available extension points.

5. Select the created extension `com.businessobjects.webpath.AnalyticalReporting_<Provider>_<ExtensionName>` to define attributes.
   a. id: `com.<Provider>.<ExtensionName>`
      The id attribute can take the bundle or package name.
   b. class: `com.<Provider>.<ExtensionName>.Extension`
      The class attribute refers to the Java class that implements the IExtension interface.
   c. name: `<ExtensionName>`
   d. provider: `<Provider>`
   e. version: `1.0.0`

6. In the Runtime tab, click Add to add the extension class path and select the bin folder.
7. In the Overview tab, check that the minimum execution environment is JavaSE-1.6.
8. Save the changes.

### 7.9 To Implement the IExtension Interface

The extension bundle must implement the IExtension interface to contribute to the bundle host. A Java class is created in the `com.sap.webi.toolkit.extension` package, which must be exported to make the interface available for the extensions.

   The package name must contain alphanumeric characters and a dot character as separator. It should be only in lower case.
2. Create a folder tree for extension files under the project.

```
web
  extension
    <Provider>
      <ExtensionName>
        assets
        css
        img
        js
    WEB-INF
    lib
```

3. Select the created package and right-click it to create the Extension Java class.
   The Java class must implement the IExtension interface.
4. Implement the `getExtensionProperties` and `getContribution` methods.

#### 7.9.1 getExtensionProperties

The `getExtensionProperties(String lang)` method returns an instance of the `ExtensionProperties` class in the required language. The `lang` parameter takes the current user interface language as value.
The instance is provided by the extension and must contain the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>The extension title displayed in Extension Management UI in the CMC.</td>
</tr>
<tr>
<td>Description</td>
<td>A short description of the feature provided by the extension.</td>
</tr>
</tbody>
</table>

### 7.9.2 getContribution

The `getContribution(String lang)` method returns a list of `UIElement` objects. The `lang` parameter takes the current user interface language as value.

In SAP BusinessObjects Web Intelligence, a `UIElement` is a graphical element such as a button, drop-down list, or text field.

> Remember

In the present release, the left-side pane and the status bar areas can accept contributions.

### Contribution to the Left-Side Pane

A contribution to the left-side pane can only be a Button widget. There can be more than one contribution.

Each contribution is an instance of the `SidepaneButton` class. The class is a `UIElement` that contains a set of properties of a button on the left-side pane:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Button identifier. It must be unique within the extension.</td>
</tr>
<tr>
<td>Title</td>
<td>Text displayed on the drop-down list of available buttons in the side pane. It also displays as tooltip of the button.</td>
</tr>
<tr>
<td>Description</td>
<td>Button description. It is only displayed on the Java Web Intelligence application.</td>
</tr>
<tr>
<td>IconURL</td>
<td>URL related to the icon button. SAP recommends to use the image type as &quot;png&quot; and the size of 24 * 24.</td>
</tr>
<tr>
<td>TargetPage</td>
<td>URL related to the main page. The content of this page is displayed on the Side panel frame. The page type could be any type that can be displayed by your Java application server.</td>
</tr>
<tr>
<td>Perspectives</td>
<td>List of perspectives where the added UIElement is visible. The list of available perspectives is listed in the <code>Perspective</code> class.</td>
</tr>
</tbody>
</table>
Contribution to the Status Bar

A contribution to the status bar can be a Button or a Toggle Button widget. There can be more than one contribution.

Each contribution is an instance of the `StatusBarButton` or `StatusBarToggleButton` class. The class is a UIElement that contains a set of properties of a button on the status bar:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Button identifier. It must be unique within the extension.</td>
</tr>
<tr>
<td>Title</td>
<td>Text displayed on the drop-down list of available buttons in the status bar. It also displays as tooltip of the button.</td>
</tr>
<tr>
<td>Description</td>
<td>Button description. It is only displayed on Java Web Intelligence application.</td>
</tr>
<tr>
<td>IconURL</td>
<td>URL related to the icon button. SAP recommends you use the image type as &quot;png&quot; and the size of 16 * 16.</td>
</tr>
<tr>
<td>TargetPage</td>
<td>URL related to the main page. The content of this page is displayed on the Side panel frame. The page type could be any type that can be displayed by your Java application server.</td>
</tr>
<tr>
<td>Text</td>
<td>Text to display in the button.</td>
</tr>
</tbody>
</table>

### 7.10 To Append Contribution Files

Once you have created the Java class that implements the `IExtension` interface, you need to add an image that represents the extension on the user interface, and a target file of the functionality provided by the extension.

A target page can be any page type supported by Java application servers (HTML, JSP, Servlet, and so on.).

When adding a button to Web Intelligence, the icon size must be 24x24 pixels.

1. Add your contribution icon to the `web\extension\<Provider>\<ExtensionName>\assets\img` folder.
2. To create an HTML page, right-click the `<ExtensionName>` root folder of the extension and select `New`.
3. Select `Web `HTML File` and click `Next`.
4. Select the parent folder, enter the HTML file name and click `Finish`.

### 7.11 To Develop with the JavaScript APIs

Once you have created the target page, you need to develop the functionality provided by the extension with the Web Intelligence Application and Service JavaScript APIs.

1. To access the Web Intelligence Application APIs, include the following files into your target pages:
   - `webi.application.js`
○ webi.application.sidepane.js
○ webi.application.dialogbox.js
○ webi.application.bar.js

○ For a left-side pane:

```html
<script type="text/javascript" src="../../../js/extension/webi.application.js"></script>
<script type="text/javascript" src="../../../js/extension/webi.application.sidepane.js"></script>
```

○ For a dialog box:

```html
<script type="text/javascript" src="../../../js/extension/webi.application.js"></script>
<script type="text/javascript" src="../../../js/extension/webi.application.dialogbox.js"></script>
```

○ For a status bar button:

```html
<script type="text/javascript" src="第六届js/extension/webi.application.js"></script>
<script type="text/javascript" src="第六届js/extension/webi.application.bar.js"></script>
```

2. To access the Web Intelligence Service APIs, include the `webi.services.js` file into your target pages as follows:

```html
<script type="text/javascript" src="第六届js/extension/webi.services.js"></script>
```

3. Develop the functionality of the extension.

### 7.12 To Make the Extension Visible on the Interface

The BOE Equinox must load the extension to make it visible in the BI launch pad.

➤ **Remember**

You must perform this task in your development environment. This is done automatically in a production environment.

1. Under the BOE project, open the `WebContent\WEBI-INF\eclipse\configuration\config.ini` file for editing.
2. Add a reference to your extension as follows:

```ini
#Eclipse Runtime Configuration File
osgi.bundles= \
org.eclipse.equinox.common@2:start, \
org.eclipse.update.configurator@start, \
org.eclipse.equinox.ds@start, \
com.businessobjects.servletbridge.core@start, \
reference\file\C:\\workshop/com.businessobjects.webpath.AnalyticalReporting_<Provider>_<ExtensionName>
osgi.bundles.defaultStartLevel=4
```
3. Save the file.
4. In the Server view, start the Apache Tomcat server.
5. In the Console view, type `ss` and press Enter to check that the extension bundle has been loaded properly.
   You should see the following information on the fifth line:

   ```
   5 RESOLVED
   com.businessobjects.webpath.AnalyticalReporting_<Provider>_<ExtensionName>_1.0.0
   ```

### 7.13 To Test the Extension Bundle

To test the extension in your development environment, you need to select it as a Web Intelligence customization in the CMC.

1. Log on to the CMC (http://<server-name>:8090/BOE/CMC) and click Users and Groups on the home page.

   ➡️ **Remember**

   Since the extension has not been deployed at this point, you must use the port configured for the Tomcat server of Eclipse (8090) to see the extension in the CMC.

2. Click Group List on the left pane to display all available groups.
3. Right-click on a group name you want to customize and click Customization. A customization dialog page is displayed.
4. Click Add Folders to select the document folders for which you want to enable the extension point. The folder appears in the list of customized folders.
5. Select the Extensions tab to display all installed extensions.
6. Check the extension you want to validate and make available for the customized folder of the users of the selected group.
   If you check the extension "<ExtensionName> 1.0.0", it will be added as an icon to the application user interface.
7. Click Save to save your selection.
8. Log on to the BI launch pad (http://<server-name>:8090/BOE/BI) and open a Web Intelligence document.
   You must use the port configured for the Tomcat server of Eclipse.

   ➡️ **Remember**

   To be able to see the extension icon:
   - The test User must be a member of the customized User Group.
   - The document must belong to the customized folder of the test User.

   You should see the extension icon on the application interface.
### 7.14 To Build the Extension Bundle

You build the extension bundle to create a deployable JAR file. In our use case we only create a build binary.

In the out-of-the-box installation, Tomcat 6 is used as an application server. The bundles are not deployed as JAR files but as subfolders under the `<tomcat-dir>\webapps\BOE\WEB-INF\eclipse\plugins` folder.

1. In Project Explorer or Package explorer, double-click the `build.properties` file to open it.
2. Select the following folders in Binary Build:
   - `META-INF`
   - `bin`
   - `fragment.xml`
   - `web`
3. Click *File* > *Export*.
4. In the Export dialog box, select *Deployable plug-ins and fragments* and click Next.
5. Select the extension bundle project in Available Plug-ins and Fragments and specify in the Destination tab the folder where the JAR file is generated.
   - If you select the project folder, then a new folder with name “plugins” is created and the JAR file is copied to this folder.

### 7.15 To Deploy the Extension Bundle in Production

You deploy the extension in your production environment.

1. Stop Apache Tomcat.
2. Copy the extension JAR file to the following folders:
   - `<bip-install-dir>\warfiles\webapps\BOE\WEB-INF\eclipse\plugins`
   - `<tomcat-dir>\webapps\BOE\WEB-INF\eclipse\plugins`
   - `<tomcat-dir>\work\Catalina\localhost\BOE\eclipse\plugins`

**Remember**

Perform the additional step below on a client machine to use the extension bundle with the Java interface of Web Intelligence. You must have Java 7 installed. You do not have to perform this step when deploying the extension points for Web Intelligence Rich Client.

4. Copy the `jfxrt.jar` file to the `ext` folder. How you do this depends on whether you are using a 32-bit or 64-bit internet browser on your client machine:
   - For a 32-bit internet browser, copy the file `C:\Program Files\Java\jre7\lib\jfxrt.jar` to the folder `C:\Program Files\Java\jre7\lib\ext`.
   - For a 64-bit internet browser, copy the file `C:\Program Files (x86)\Java\jre7\lib\jfxrt.jar` to the folder `C:\Program Files (x86)\Java\jre7\lib\ext`. 
To use the extension in your production environment, make sure you selected the extension in the CMC. Use the port configured for the Tomcat server of the BI platform (8080). See To Test the Extension Bundle [page 37].

### 7.16 About the Web Intelligence UI Extension Point Sample

The sample is a ready-to-use extension that demonstrates the usage of the JavaScript APIs. It allows you to test the following features:

- Refresh reports periodically
- List document reports in JSON format

The sample is installed with the SAP BusinessObjects BI platform servers and deployed automatically at installation in the following directories:

- `<bip-install-dir>\warfiles\webapps\BOE\WEB-INF\eclipse\plugins\com.businessobjects.webpath.AnalyticalReporting_SAP_ExtensionSample_1.0.0.jar`
- `<tomcat-dir>\webapps\BOE\WEB-INF\eclipse\plugins`
- `<tomcat-dir>\work\Catalina\localhost\BOE\eclipse\plugins`

The JAR file content is extracted automatically on the Tomcat server.

#### 7.16.1 To Use the Extension Sample

The `com.businessobjects.webpath.AnalyticalReporting_SAP_ExtensionSample_1.0.0.jar` sample has been deployed automatically on the BI platform and Tomcat servers at platform installation.

1. Logon to the CMC to make the extension visible on the Web Intelligence interface. See instructions To Test the Extension Bundle [page 37].
2. Logoff from the CMC and logon to the BI launch pad.
3. Open any Web Intelligence document.
   - The Extension Sample pane displays in the Web Intelligence left-side panel.
4. Play with the extension:
   - Enter a refresh schedule time (in seconds) and click Start. Click Stop to stop the refresh.
   - Click Display. A dialog box opens that contains the list of reports as a JSON object.
8 Creating Visualizations with a Custom Element Service

You can extend the list of visualizations supported by SAP BusinessObjects Web Intelligence through custom elements.

Custom elements are a new type of visualizations, whose implementation relies on a REST web service that you or a service provider must develop. This implementation must use the APIs described in the following sections. The implementation of the service defines the format of the custom elements and how to handle the data that the visualizations must render.

What You Need to Do

1. Develop or ask a service provider to develop the custom element service with the REST APIs. You can develop as many custom element services as you need.
2. Add the name and URL of a custom element service in the CMC.

Users can now use custom elements in their reports. For more information, see Managing Web Intelligence settings in the Business Intelligence Platform Administrator Guide.

How Custom Elements Display in Web Intelligence

Once a custom element service has been added to the Web Intelligence settings in the CMC, the visualizations provided by the service display as available charts in the Web Intelligence client interfaces. You can select one of them to display your data as in any other charts supported by Web Intelligence.

A custom element is defined as any other block by the following properties:

- Size, position, and borders
- Formulas on feeds

**Note**

To export a custom element into a PDF document or a Microsoft Excel file, the service must return the custom element as an image. Otherwise, the custom element is filled with blank.

For more information, see Working with charts in reports in the SAP BusinessObjects Web Intelligence User’s Guide.
How the Service Works

When a report that contains a custom element is displayed:
1. The Web Intelligence Server performs the calculation and sends the computed data and their metadata to the service.
2. The service generates an image or an HTML document as output of the request.
3. The HTML output is added to an Iframe and the image output to a standard div to the report.

8.1 About the Custom Element Service APIs

The Custom Element Service APIs are REST APIs that provide the following features:
- Getting the supported formats
- Getting the visualization types
- Getting the feeding definitions of a visualization
- Rendering a visualization

Base URL

The base URL to use within the call requests must refer to the server where the service is deployed. For example:

http://myvisualizationserver.domain.com:8095

Response Status and Error Messages

Table 14: HTTP Codes and Descriptions

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Successful request</td>
</tr>
<tr>
<td>500</td>
<td>Failed request</td>
</tr>
</tbody>
</table>

If the request has failed, the call returns a response in JSON format as follows:

```json
{
   "code": "XXX 00001",
   "message": "Invalid feeding."
}
```

Where:
- code specifies an error code for a type of issue. The preferred format consists of 3 letters indicating your service name and a 5-digit number separated by a space character.
• message is a descriptive error message.

Localization

Although the service uses the English locale by default for responses, it also supports other languages according to user preferences. To this end, you can use the locale query parameter that contains the country and optionally the language code in requests. For example:

api/visualizations?locale=en
api/vizualisations/vizID/feeds?locale=fr_FR

8.2 Getting the Supported Formats

Returns the list of media types supported by the custom element service.

It can be a selection of the following types:

- text/html
- image/png
- image/jpg
- image/gif
- image/bmp

The preferred media type is text/html, which allows interactivity in SAP BusinessObject Web Intelligence interfaces and a better user experience. The preferred image output is image/png.

Note

The custom element service can return several media types. For example, it can return text/html to display the custom elements in a Web Intelligence report and image/png to export that report into a PDF document or a Microsoft Excel file.

Request

URI: api/formats

HTTP Method: GET

Request Parameters: None
Response

Format: JSON

Response Example:

```json
{
    "formats": [
        "text/html",
        "image/png"
    ]
}
```

8.3 Getting the Visualization Types

Returns the list of visualizations supported by the Custom Element service.

Request

URI: api/visualizations

HTTP Method: GET

Request Parameter:

Table 15:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
<th>Send As</th>
</tr>
</thead>
</table>
| locale    | No       | String| The user locale as a language and/or country string. For example:  
|           |          |       |             | en, fr_FR          |

Response

Format: JSON

Response Example:

```json
{
    "visualizations": [
        {
            "id": "bullet-chart",
            "name": "Bullet chart",
            "description": "This a bullet chart"
        },
    ]
}
```
Table 16: Description of a Visualization

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The visualization identifier. Must be unique in the scope of the service.</td>
</tr>
<tr>
<td>name</td>
<td>The visualization name can be localized by using the locale query parameter.</td>
</tr>
<tr>
<td>description</td>
<td>The visualization description can be localized by using the locale query parameter.</td>
</tr>
</tbody>
</table>

8.4 Getting the Visualization Feed Definitions

Returns the list of data feeds of a visualization type specified by its identifier.
This is the description of each feed supported by the visualization type.

Request

**URI:** api/visualizations/<vizID>/feeds

**HTTP Method:** GET

**Request Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
<th>Send As</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;vizID&gt;</td>
<td>Yes</td>
<td>String</td>
<td>The visualization type identifier</td>
<td>Path parameter</td>
</tr>
</tbody>
</table>
| locale    | No       | String | The user locale as a language and/or country string. For example:  
  - en  
  - fr_FR | Query parameter |

Response

**Format:** JSON
Response Example:

**GET** /api/visualizations/funnel/feeds?locale=en

```json
{
  "feeds": [
    {
      "id": "category-axis",
      "name": "Category Axis",
      "description": "The primary chart axis",
      "axis": "0",
      "type": "dimension",
      "min": "1",
      "max": "-1"
    },
    {
      "id": "region-color",
      "name": "Color",
      "description": "This feeds is used to customize bar colors",
      "type": "dimension",
      "axis": "1",
      "min": "0",
      "max": "-1"
    },
    {
      "id": "primary-values",
      "name": "Values",
      "description": "Feeds for measures",
      "type": "measure",
      "min": "0",
      "max": "-1"
    }
  ]
}
```

Table 18: Description of a Feed

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The unique identifier in the scope of the current visualization type</td>
</tr>
<tr>
<td>name</td>
<td>The human-readable name of the feed. Can be localized using the locale query parameter</td>
</tr>
<tr>
<td>description</td>
<td>The human-readable description of the feed. Can be localized using the locale query parameter</td>
</tr>
<tr>
<td>type</td>
<td>The type of data provided by the feed: measure or dimension, which includes the BI Semantic Layer types dimension, detail, hierarchy, and level</td>
</tr>
<tr>
<td>axis</td>
<td>The axis type: primary (0) or secondary (1). Mandatory for feeds of type dimension</td>
</tr>
<tr>
<td>min</td>
<td>The minimum number of formulas required by the feed</td>
</tr>
<tr>
<td>max</td>
<td>The maximum number of formulas that the feed can manage</td>
</tr>
</tbody>
</table>
8.5 Rendering the Visualization

Renders the visualization in the specified format.

The request must contain the following inputs:

- The output format details
- The Web Intelligence expressions
- The dataset

Request

**URI**: api/visualizations/render<vizID>/render

**HTTP Method**: POST

**Request Header Parameters**:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Yes</td>
<td>The output format. For example: image/png</td>
</tr>
<tr>
<td>Content-Type</td>
<td>Yes</td>
<td>application/json</td>
</tr>
</tbody>
</table>

**Request Parameters**:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
<th>Send As</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;vizID&gt;</td>
<td>Yes</td>
<td>String</td>
<td>The visualization type identifier</td>
<td>Path parameter</td>
</tr>
<tr>
<td>width</td>
<td>Yes</td>
<td>Numeric</td>
<td>The visualization width</td>
<td>Request body parameter</td>
</tr>
<tr>
<td>height</td>
<td>Yes</td>
<td>Numeric</td>
<td>The visualization height</td>
<td>Request body parameter</td>
</tr>
<tr>
<td>dpi</td>
<td>Yes</td>
<td>Numeric</td>
<td>The visualization dpi</td>
<td>Request body parameter</td>
</tr>
<tr>
<td>font</td>
<td>Yes</td>
<td>N/A</td>
<td>The default font to use in the visualization. Must be installed on the third-party server.</td>
<td>Request body parameter</td>
</tr>
<tr>
<td>feed</td>
<td>Yes</td>
<td>N/A</td>
<td>The list of Web Intelligence formulas applied by the end-user to each feed. Each feed contains an array of expressions. Each expression contains a reference to an object of the data parameter based on its identifier.</td>
<td>Request body parameter</td>
</tr>
<tr>
<td>Parameter</td>
<td>Required</td>
<td>Type</td>
<td>Description</td>
<td>Send As</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>------</td>
<td>-------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| data      | Yes      | N/A  | The dataset. Each data entry contains the following:  
- A unique identifier  
- An optional title  
- dataType that can be string, double or date.  
- dataStructure that can be tree if it is based on hierarchical data or simple if it is flat data.  
- The data itself  
- cardinality that specifies if the data is contained in an array (1) or a matrix (2). | Request body parameter |

Request Example:

**POST /api/visualizations/funnel/feeds/render?locale=en_US**

```json
{
  "width":400.0,
  "height":300.0,
  "dpi":96,
  "font":{
    "name":"Arial",
    "size":9,
    "color":"#333333",
    "isBold":false,
    "isItalic":false
  },
  "feeding":[
    {
      "id":"category-axis",
      "expressions":[
        {
          "dataId":3
        }
      ],
    },
    {
      "id":"region-color",
    },
    {
      "id":"primary-values",
      "expressions":[
        {
          "dataId":1
        }
      ]
    }
  ],
  "data":[
    {
      "id":3,
      "title":"Year",
      "values":{
        "dataType":"string",
        "rawvalues":[
          "2004",
          "2005",
          ...SMARTTABLE...
        ]
      }
    }
  ]
}
```
"2006",
  "dataStructure":"simple",
  "cardinality":1,
},
  "type":"dimension"
},
  {
    "id":"1",
    "title":"Sales revenue",
    "values":{
      "dataType":"double",
      "rawvalues":[
        8095814,
        1.3232246E7,
        1.50591428E7
      ],
      "dataStructure":"simple",
      "cardinality":1
    },
    "type":"measure"
  }
}

**Response**

**Format:** A stream containing the response.

- If the output format is `text/html`, it is an HTML document that will be loaded to the dedicated iframe.
- If the output format is an image, it is the resulting image.
9 Exposing Web Intelligence Features with REST Web Services

The Web Intelligence RESTful Web Service SDK provides a series of REST APIs that allows you to expose the Web Intelligence functionalities into your analytics applications.

Since the SDK is provided with the BI platform, you have nothing to install on the developer machine or where your application is deployed. The major benefit of the SDK is that you can use the REST APIs with any programming language that support the HTTP protocol, so that end-users can access the broad range of Web Intelligence features in many ways. A web service performs CRUD (Create, Read, Update, Delete) operations on data over HTTP, sending requests and receiving responses either in XML or JSON format. The way you implement these services is at your convenience. For example, you can automate batch operations on Web Intelligence documents. You can also make documents and reports available into non-SAP web applications.

The REST APIs expose features that relate to all Web Intelligence functional domains:

- Creating documents and building queries
- Creating reports with tables, sections, and charts
- Refreshing documents to get data
- Formatting reports
- Saving and exporting documents and reports
- Scheduling documents

Some Java samples are also provided to help you understand the REST APIs. They are supplied in the archive `<bip-install-dir>\Samples\webi\RaylightRESTWS_Samples.zip`

### Note

Before using the APIs, you need to logon to the BI platform and access the document or universe folder via the BI platform RESTful Web Service SDK.

<table>
<thead>
<tr>
<th>Documentation Deliverable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>See the SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer on the help portal.</td>
<td>The official guide for developing with the Web Intelligence RESTful Web Service SDK</td>
</tr>
<tr>
<td>See the Business Intelligence platform RESTful Web Service Developer Guide on the help portal.</td>
<td>The official guide for developing with the BI platform RESTful Web Service SDK</td>
</tr>
</tbody>
</table>
10 Consuming BI Semantic Layer Universes with REST Web Services

The BI Semantic Layer RESTful Web Service SDK provides a series of REST APIs that allow you to access relational universes, browse universe metadata, create and execute queries. It supports UNV universes created with the universe design tool as well as UNX universes created with the information design tool.

Since the SDK is provided with the BI platform, you have nothing to install on the developer machine or where your application is deployed. The major benefit of the SDK is that you can use the REST APIs with any programming language that support the HTTP protocol. For example, you can rewrite a query panel using JavaScript. A web service performs CRUD (Create, Read, Update, Delete) operations on data over HTTP, sending requests and receiving responses either in XML or JSON format. The way you implement these services is at your convenience. Result sets are returned using the OData protocol.

The REST APIs expose features that relate to the main functional domains of the SAP BusinessObjects reporting tools:

- Browsing universes and retrieving metadata
- Building queries on universes and retrieving data

Some Java samples are provided to help you understand the REST APIs. They are supplied in the archive `<bip-install-dir>\SL SDK\SDK Samples\SLRESTWebService.zip`.

**Note**

Before using the APIs, you need to logon to the BI platform and access the universe folder via the BI platform RESTful Web Service SDK.

<table>
<thead>
<tr>
<th>Documentation Deliverable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>See the SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer RESTful Web Service SDK</td>
<td>The official guide for developing with the BI Semantic Layer RESTful Web Service SDK</td>
</tr>
<tr>
<td>See the Business Intelligence platform RESTful Web Service Developer Guide on the help portal</td>
<td>The official guide for developing with the BI platform RESTful Web Service SDK</td>
</tr>
</tbody>
</table>
11 Developing Applications to Design and Administrate Universes

The BI Semantic Layer Java SDK gives allows you to access the features of the information design tool within a program of your own. You can develop Java applications to design the UNX universe resources (data foundations, business layers, and connections), to publish them in a CMS repository, and to configure security settings on published universes.

Some samples are also provided to help you understand the Java SDK APIs. They are supplied in the archive \bip-install-dir\SL SDK\SDK Samples\com.sap.sl.sdk.authoring.samples.source.jar.

Similarly, the Universe Design Tool COM SDK gives you access to the features of the universe design tool. You can develop applications to design and manage UNV universes using the provided COM objects.

Table 23:

<table>
<thead>
<tr>
<th>Documentation Deliverable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP BusinessObjects BI Semantic Layer Java SDK Developer Guide</td>
<td>The official user guide for developing with the BI Semantic Layer Java SDK</td>
</tr>
<tr>
<td>SAP BusinessObjects BI Semantic Layer Java API Reference</td>
<td>The reference for interfaces and methods of the Java APIs</td>
</tr>
<tr>
<td>SAP BusinessObjects BI Semantic Layer Java Object Model Diagrams</td>
<td>The object model diagrams of the BI Semantic Layer Java SDK</td>
</tr>
<tr>
<td>SAP BusinessObjects Universe Design Tool COM API Reference</td>
<td>The reference for the COM objects and methods</td>
</tr>
<tr>
<td>SAP BusinessObjects Universe Design Tool Object Model Diagrams</td>
<td>The object model diagrams of the Universe Design Tool COM SDK</td>
</tr>
</tbody>
</table>
Creating a Data Access Driver

A data access driver is a software component that runs with the data access service of the BI platform called Connection Server to perform requests to data sources and retrieve data for UNV and UNX universes. SAP BusinessObjects applications use a wide range of data access drivers to communicate with database middleware. In addition to the supplied data access drivers, you can use the Driver Development Kit to develop, using Java, data access drivers for data sources for which there are no drivers available.

**Note**

The JavaBean driver installed on the BI platform manages data that only results from stored procedure execution. Only UNV universes support stored procedures.

Some JavaBean and Open driver samples are provided at `<bip-install-dir>/data access\connectionServer/DDK\examples` to help you develop your own Java driver:

- The Open driver sample illustrates how to develop a driver that accesses data stored in a comma-separated value (CSV) file.
- The JavaBean driver sample illustrates how to develop a JavaBean driver that accesses data stored in an Excel spreadsheet.

The driver sample codes are also available on the SCN.

Table 24:

<table>
<thead>
<tr>
<th>Documentation Deliverable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP BusinessObjects Data Access Driver Java SDK Developer Guide</td>
<td>The official guide for developing with the Driver Development Kit. This document version is related to the 4.0 platform, but its content is also valid for the 4.1 and 4.2 releases.</td>
</tr>
<tr>
<td>SAP BusinessObjects Data Access Driver Java API Reference</td>
<td>The reference for interfaces and methods of the Java APIs</td>
</tr>
</tbody>
</table>

Related Information

Data Access Driver Samples
Important Disclaimers and Legal Information

Coding Samples

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

Accessibility

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

Gender-Neutral Language

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

Internet Hyperlinks

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