PUBLIC

SAP BusinessObjects Business Intelligence platform

Document Version: 4.3 Support Package 4 – 2023-12-07

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

© 2024 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

# Content


**1 Document Version History. . . . 3**

**2 About This Guide. . . . 4**

**3 Audience. . . . 5**

**4 Conventions in This Guide. . . . 6**

**5 Opening a Web Intelligence document using OpenDocument URL. . . . 7**

**6 Customizing Web Intelligence User Interfaces. . . . 8**

6.1 Customizing Web Intelligence interface elements by user groups and folders. . . . 8

Customization interface. . . . 10

Customization rules. . . . 10

To customize the Web Intelligence interface appearance. . . . 11

6.2 Web Intelligence content alignment. . . . 12

**7 Creating Web Intelligence Visualizations with a Custom Element Service . . . . 13**

7.1 About the Custom Element Service APIs. . . . 15

7.2 Getting the Supported Formats. . . . 16

7.3 Getting the Visualization Types. . . . 17

7.4 Getting the Visualization Icon. . . . 18

7.5 Getting the Visualization Feed Definitions. . . . 19

7.6 Rendering the Visualization. . . . 20

7.7 Getting the Rendering Settings. . . . 24

**8 Adding Custom Formula in Web Intelligence Formula Language. . . . 31**

**9 Exposing Web Intelligence Features with REST Web Services. . . . 32**

**10 Consuming BI Semantic Layer Universes with REST Web Services. . . . 33**

**11 Developing Applications to Design and Administrate Universes. . . . 34**

**12 Creating a Data Access Driver with JavaBean. . . . 35**

# 1 Document Version History

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

| Version | Date | Change |
| --- | --- | --- |
| SAP BusinessObjects BI Developer's Guide for Web Intelligence and the BI Semantic Layer 4.3 SP4 | December 2023 | New customization items. |
| SAP BusinessObjects BI Developer's Guide for Web Intelligence and the BI Semantic Layer 4.3 SP1 | December 2020 | Added chapter about UI customization. See Customizing Web Intelligence interface elements by user groups and folders [page 8] |
| SAP BusinessObjects BI Developer's Guide for Web Intelligence and the BI Semantic Layer 4.3 | April 2020 | Initial release |

# 2 About This Guide

The SAP BusinessObjects BI platform 4.3 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:

- How to create your visualizations with 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

# 3 Audience

As it serves as en 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 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
- 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

# 4 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 Business Objects\SAP BusinessObjects Enterprise XI 4.0` directory.

The placeholder `<tomcat-dir>` stands for the `C:\Program Files (x86)\SAP BusinessObjects\tomcat` directory.

# 5 Opening a Web Intelligence document using OpenDocument URL

You can use the OpenDocument URL to create a hyperlink for content in the SAP BI Platform repository, including a folder, a Web Intelligence document, report, or block. Share this hyperlink with other users who can access the content without navigating in the repository after authentication.

If you are developing an application, you can also utilize Web Intelligence content by generating an OpenDocument URL to open a meaningful document.

The OpenDocument URL syntax offers numerous options for defining both the content to open and how to open it. For example, you can refresh a document or answer to prompts.

For more details, refer to *Viewing Documents Using OpenDocument* guide.

# 6 Customizing Web Intelligence User Interfaces

You can simplify the appearance of the web client by hiding some functionalities through the CMC.

## 6.1 Customizing Web Intelligence interface elements by user groups and folders

Customization allows you to hide multiple interface elements to simplify the way end-users interact with the application, depending on user groups and folders containing Web Intelligence documents. You can hide data sources types, edit mode toggle, turn-off the auto refresh feature, and many more.

By default, every interface element is enabled. If you want to hide them, you can do so in the Central Management Console. The table below lists the user interface elements you can hide.

| Features List | Description |
|---|---|
| *Mode* | Hides the available modes the user can access through the drop-down button.<br>• Reading<br>To hide the Reading mode from the drop-down button.<br>• Design<br>To hide both the Design and Structure modes from the drop-down button.<br>• Data<br>To hide the Data mode from the drop-down button.<br>If all modes are disabled, then documents can only be opened in Reading mode. |
| *Location* | Hides a whole category of data sources. Categories you can disable are:<br>• BI Platform Repository<br>• Local (only available in Rich Client)<br>• Web Services<br>• Google Drive<br>• Microsoft OneDrive |

| Features List | Description |
| --- | --- |
| *Data Source* | In Design mode, you can restrict the data sources available in the *Select a Data Source* and the *Change Source* dialog boxes.<br>The data sources you can disable are:<br>• Universes<br>• Web Intelligence documents<br>• Excel files<br>• Text files<br>• SAP BW<br>• SAP HANA views<br>• Free Hand SQL queries<br>• OData<br>• Google Spreadsheet |
| *Query* | • Refresh<br>In Reading mode, hides the *Data* section in the toolbar. In Design mode, hides the *Refresh* dropdown menu, the *Refresh All* command, the *Run* button and its dropdown menu in the Query Panel.<br>• Advanced Refresh<br>In Design mode, hides the *Advanced Refresh* command in the *Refresh* dropdown menu.<br>• Auto-Refresh<br>Hides the *Auto-Refresh* option in the Presentation Mode.<br>• Change Source<br>In Design mode, hides the ability to change the document's data sources. |
| *Data* | In Data mode, hides the Combine Cubes features. |
| *Analysis* | • Drill<br>In both Reading and Design mode, hides the *Drill* checkbox in the Analyze section of the toolbar, drill filters in the *Filter Bar*. Also, in the report, values that can be drilled are not displayed as hyperlinks and the drill actions and icons available for these values are hidden.<br>In Design mode, hides the drill filters in the *Build* panel, under *Data Filters*.<br>• Track Data Changes<br>In both Reading and Design mode, hide the *Track Data Changes* and *Show Changes* from the toolbar. |
| *Documents* | • *New*, *Open*, *Save*, *Favorites*, *Presentation Mode*.<br>Hides the corresponding buttons from the toolbar.<br>• Comments<br>In both Reading and Design modes, hide the *Comments* tab in the side panel, and the *Comments* command in the contextual menu.<br>• Shared Elements<br>In Design mode, hide the *Shared Elements* tab in the side panel, and the *Shared Elements* command in the *Insert* section of the toolbar. |

| Features List | Description |
| --- | --- |
| *Export To* | In any modes, hides the possibility to export documents report and cubes to:<br>• Excel<br>• PDF<br>• HTML<br>• TXT<br>• CSV |
| *Generate Link* | In Design mode, hides the ability to create OpenDocument link and generate OData links for queries and individual report elements from contextual menus. |
| *Schedule & Publishing* | Hides the possibility to schedule and publish documents to TXT, XLS, PDF, HTML, MHTML, and CSV. |

## 6.1.1 Customization interface

You can select individual folders so that the documents they contain automatically benefit from the customization. Simply select one or more folders in the *Customized folders* area, and move on to the *Features* tab to start customizing. By default, the customization applies to every document in the folder you have selected.

The *Features* tab lists all the features you can enable or disable. Use the dedicated checkboxes to toggle them on or off.

## 6.1.2 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 defines customizations for the document for user interface elements, features, and extensions.
- The customization defined for Default Folders applies 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 feature is only a shortcut to enable all user interface elements.
- Scenario: When the customization elements are displayed as a tree list and you disable a node on a system. Here, if you upgrade this system with a newer version of the product having new items in the nodes, then by default these items are activated even if the upper node is disabled.

## 6.1.3 To customize the Web Intelligence interface appearance

You can customize the appearance of the Web Intelligence user interface by hiding menu items, subitems, 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 *Customized folders* section, do one of the following:

| Option | Description |
|---|---|
| **To define a default customization** | 1. Select *Default Folders* in the *Customized folders* area. |
| **To add the document folders for which you want to apply customization for the selected user group** | 1. Click *Add Folder*.<br>2. Select the folders.<br><br>The folders displays in the *Customized folders* area. |
| **To avoid redefining the same customization for other folders** | 1. In the *Customized folders* area, select the folder from which you want to copy the customization.<br>2. In the dropdown list, click *Duplicate Customization*.<br>3. Select the folder to which you want define the customization.<br>4. Click *Paste Customization*.<br>5. Go to step 7. |
| **To remove the customization for a specific folder** | 1. In the *Customized folders* area, select the folder.<br>2. In the dropdown list, click *Remove Folder*.<br>3. Go to step 7.<br><br>**Note**<br>You can't remove *Default Folders*. |

6. Select or deselect items in the *Features* tab to display or hide them in Web Intelligence.

   If you deselect all children of a parent item, the parent item is also deselected and hidden in Web Intelligence. For more information, check out Customizing Web Intelligence interface elements by user groups and folders [page 8].
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.

> **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 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 doesn't affect existing documents.

# 7 Creating Web Intelligence 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 type of visualizations whose rendering is delegated to a third-party service. You develop this service (or ask a service provider to develop it for you) by implementing REST API calls, in order to allow the third-party server to communicate with Web Intelligence servers.

A custom element service must provide useful information for Web Intelligence, such as the size of the custom element, how to display the data in the chart, and the rendering settings.

You can develop as many custom element services as you need. A service can also host several custom elements, hence several visualizations.

## What You Need to Do

1. Develop or ask a service provider to develop the custom element service. The service must implement the REST APIs calls.
2. Deploy your service on your third-party server.
3. As an administrator, add the custom element service URL to the list of trusted URLs in the CMC. See .
4. As an administrator, connect to the CMC and add the name and URL of the custom element service you have deployed. Click *Test* to make sure the service respects the API contract.

> **Note**
>
> This is the responsibility of the administrator to ensure that data exchange between the service and the BI platform does not affect the security of the production system.

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. The end-user can select one of them to display their data as in any other charts supported by Web Intelligence.

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

- Size, position, and borders
- Formulas on feeds

- Rendering settings

For more information, see *Working with charts in reports* in the *SAP BusinessObjects Web Intelligence User's Guide*.

> **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 displays as blank.

> **Remember**
>
> Make sure your configuration allows access to the third-party server from the BI platform server. Otherwise, the custom element won't display. In the case of a Web Intelligence Rich Client not connected to the CMS repository, the custom element does not display either.

## How the Service Works

### When an administrator configures a custom element service

To confirm that the service is up and running and to select an output format for the custom element visualizations, the administrator tests the service in the Central Management Console. See Getting the Supported Formats [page 16] for more information.

### When an end-user inserts a custom element visualization in a Web Intelligence document:

The name of the custom element service is the name chosen by the administrator when configuring this service. In the interface, the end-user can select a visualization in the list offered by the service. Since the 4.2 SP4 release, the service can also display a thumbnail of each visualization to better illustrate its content. See Getting the Visualization Types [page 17] and Getting the Visualization Icon [page 18] for more information.

The end-user can assign data to the visualization he has selected according to the feeding defined by the custom element service. See Getting the Getting the Visualization Feed Definitions [page 19] for more information.

### When an end-user customizes the parameters of a custom element visualization (since 4.2 SP3):

The end-user can modify the settings values supported by the custom element service for the selected visualization. See Getting the Rendering Settings [page 24] for more information.

### When a report that contains a custom element is displayed:

Based on end-user choices, the Web Intelligence server performs the calculation and sends the computed data and their metadata to the service. The service then generates an image or an HTML document as output of the request. Finally, the HTML output is added to an <iframe> tag and the image output to a standard  tag to the report, which is displayed in the Web Intelligence interface.

### When publishing or printing a Web Intelligence document displaying custom elements:

The Web Intelligence server requests an image output from the custom element service, independently of the output format selected by the administrator in the Central Management Console.

## 7.1 About the Custom Element Service APIs

You implement the custom element service APIs to display your visualization through a custom element in Web Intelligence.

Using these REST APIs, you allow Web Intelligence to:

- Get the formats supported by the service
- Get the types of visualization provided by the service
- Get the feeding definitions of a specific custom visualization
- Display the visualization properly
- Get the rendering settings of the 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

HTTP Codes and Descriptions

| Code | Description |
|---|---|
| 200 | Successful request |
| 500 | Failed request |

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

```
{
 "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 must use the English locale by default for responses, it should also support 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
```

## 7.2 Getting the Supported Formats

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

The Web Intelligence server needs this information to know what media types the service supports.

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`.

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:**

```
{
```

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

## 7.3 Getting the Visualization Types

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

The Web Intelligence server needs this information to know what visualizations the service supports.

### Request

**URI:** `api/visualizations`

**HTTP Method:** *GET*

**Request Parameter:**

| Parameter | Required | Data Type | Description | Parameter Type |
|---|---|---|---|---|
| `locale` | No | String | The user locale as a language and/or country string. For example:<br>• `en`<br>• `fr_FR` | Query parameter |

### Response

**Format:** *JSON*

**Response Example:**

*GET* `/api/visualizations?locale=en`

```
{
 "visualizations": [
   {
    "id": "bullet-chart",
    "name": "Bullet chart",
    "description": "This a bullet chart"
   },
   {
    "id": "funnel",
    "name": "Funnel chart",
    "description": "This is a funnel chart"
   }
  ]
```

```
}
```

| Output | Description |
|---|---|
| `id` | The visualization identifier. Must be unique in the scope of the service. |
| `name` | The visualization name as it appears in Web Intelligence. Its translation can be returned, depending on the `locale` query parameter. |
| `description` | The visualization description as it appears in Web Intelligence. Its translation can be returned, depending on the `locale` query parameter. |

## 7.4 Getting the Visualization Icon

Returns a picture that illustrates the visualization type identified by identifier.

You get the visualization identifier with the `api/visualizations` call.

Web Intelligence calls this API to display the visualization thumbnail in the dialog box listing all available charts. The picture's requested dimensions (width and height) are passed in the request. In this case, the requested dimensions are 50x50 pixels and the requested format is picture/png.

### Request

**URI:** `api/visualizations/<vizID>/sample`

### HTTP Method

**URI:** *POST*

**Request Headers:**

| Header | Required | Value |
|---|---|---|
| Accept | Yes | The output format. For example, image/png |
| Content-type | Yes | application/json |

**Request Parameters:**

| Parameter | Required | Data Type | Description | Parameter Type |
|---|---|---|---|---|
| height | Yes | Numeric | The sample height | Request body |
| width | Yes | Numeric | The sample width | Request body |

**Request Example:**

*POST* `/api/visualizations/funnel/sample?height=50&width=50`

### Response

**Format:** a 50x50 pixels .png image representing the visualization sample.

## 7.5 Getting the Visualization Feed Definitions

Returns the list of data feeds of a visualization type specified by its identifier.

The Web Intelligence server needs this information to know the feed definition that a specific visualization type supports. You get the visualization identifier with the `api/visualizations` call.

### Request

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

**HTTP Method:** *GET*

**Request Parameters:**

| Parameter | Required | Data Type | Description | Parameter Type |
|---|---|---|---|---|
| `<vizID>` | Yes | String | The visualization type identifier | Path |
| `locale` | No | String | The user locale as a language and/or country string. For example:<br>• `en`<br>• `fr_FR` | Query |

### Response

**Format:** *JSON*

**Response Example:**

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

```
{
 "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"
     }
    ]
}
```

| Output | Description |
|---|---|
| `id` | The unique identifier in the scope of the current visualization type |
| `name` | The human-readable name of the feed. Can be localized using the `locale` query parameter |
| `description` | The human-readable description of the feed. Can be localized using the `locale` query parameter. |
| `type` | The type of data provided by the feed: `measure` or `dimension`, which includes the BI Semantic Layer types dimension, detail, hierarchy, and level |
| `axis` | The axis type: primary (`0`) or secondary (`1`). Mandatory for feeds of type `dimension` |
| `min` | The minimum number of formulas required by the feed |
| `max` | The maximum number of formulas that the feed can manage. `max=-1` means there is no limit to the number of formulas. |

## 7.6 Rendering the Visualization

Renders the visualization in the specified format.

The request contains the following inputs:

- The output format details (height, width, and dpi)
- The default font and palette settings to use in the visualization
- The end-user settings
- The Web Intelligence expressions applied by the end-user to each feed

- The metadata of the visualization and the dataset

You get the end-user settings from the call Getting the Rendering Settings [page 24]. Rendering setting values are saved in the Web Intelligence document.

## Request

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

**HTTP Method:** *POST*

**Request Headers:**

| Header | Required | Value |
|---|---|---|
| `Accept` | Yes | The output format. For example: `image/png` |
| `Content-Type` | Yes | `application/json` |

**Request Parameters:**

| Parameter | Required | Data Type | Description | Parameter Type |
|---|---|---|---|---|
| `<vizID>` | Yes | String | The visualization type identifier | Path |
| `locale` | No | String | The user locale as a language and/or country string. For example:<br>• `en`<br>• `fr_FR` | Query |
| `width` | Yes | Numeric | The visualization width | Request body |
| `height` | Yes | Numeric | The visualization height | Request body |
| `dpi` | Yes | Numeric | The visualization dpi | Request body |
| `font` | Yes | JSON object | The default font to use in the visualization. Must be installed on the third-party server. | Request body |
| `feed` | Yes | JSON array | 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. | Request body |

| Parameter | Required | Data Type | Description | Parameter Type |
|---|---|---|---|---|
| `data` | Yes | JSON array | The metadata of the visualization. Each `data` entry contains the following:<br>• A unique identifier<br>• An optional title<br>• `dataType` that can be `string`, `double` or `date`<br>• `dataStructure` that can be `tree` if it is based on hierarchical data or `simple` if it is flat data<br>• The data itself<br>• `cardinality` that specifies if the data is contained in an array (`1`) or a matrix (`2`). | Request body |
| `instances` | Yes | JSON object | The custom element source, specified by the following:<br>• The Web Intelligence document CUID<br>• The report identifier<br>• The block identifier<br>• The instance identifier, which can be used to differentiate two documents with the same CUID<br>• The temporary reference to the report element instance (not valid after a refresh)<br>• The unique reference identifier that can be used even after a refresh | Request body |
| `settings` | Yes | JSON array | The values defined by the end-user in the *Format Custom Element* dialog box in Web Intelligence. It contains only non-default values. You get these values from the call Getting the Rendering Settings [page 24]. The property identifier may contain a slash (`/`) if the property has subproperties. | Request body |
| `customProperties` | Yes | JSON array | The custom properties defined on the custom element. | Request body |
| `palette` | Yes | JSON array | The list of default hexadecimal colors to use in the visualization. | Request body |

**Request Example:**

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

```
{
 "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",
            "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"
    }
 ],
```

```
  "instance":{
     "documentId":"AVyTZhNVc9dHlaVGjuyOoqw",
     "instanceId":"7130cd02788a5cb8b1a44f409bfed81d58eb2c0c",
     "reportId":"4",
     "blockId":"5",
     "iref":"5.c",
     "uiref":"UIREF:RID=5:BID=38"
    },
  "settings":[
     {
      "property":"location",
      "region":"legend",
      "value":"top"
     }
  ],
  "customProperties":[
     {
      "property":"Custom_Prop2",
      "value":"{\"value1\":\"myvalue\",
\"value2\":\"myCustomJSONDefinedSetting\"}"
     },
     {
      "property":"Custom_Prop1",
      "value":"myvalue"
     }
  ],
  "palette":[

"#008fd3ff","#99d101ff","#f39b02ff","#9fcfecff","#4ba707ff","#f6d133ff","#cb4d2cf
f","#cac7baff
    ]
}
```


### 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.

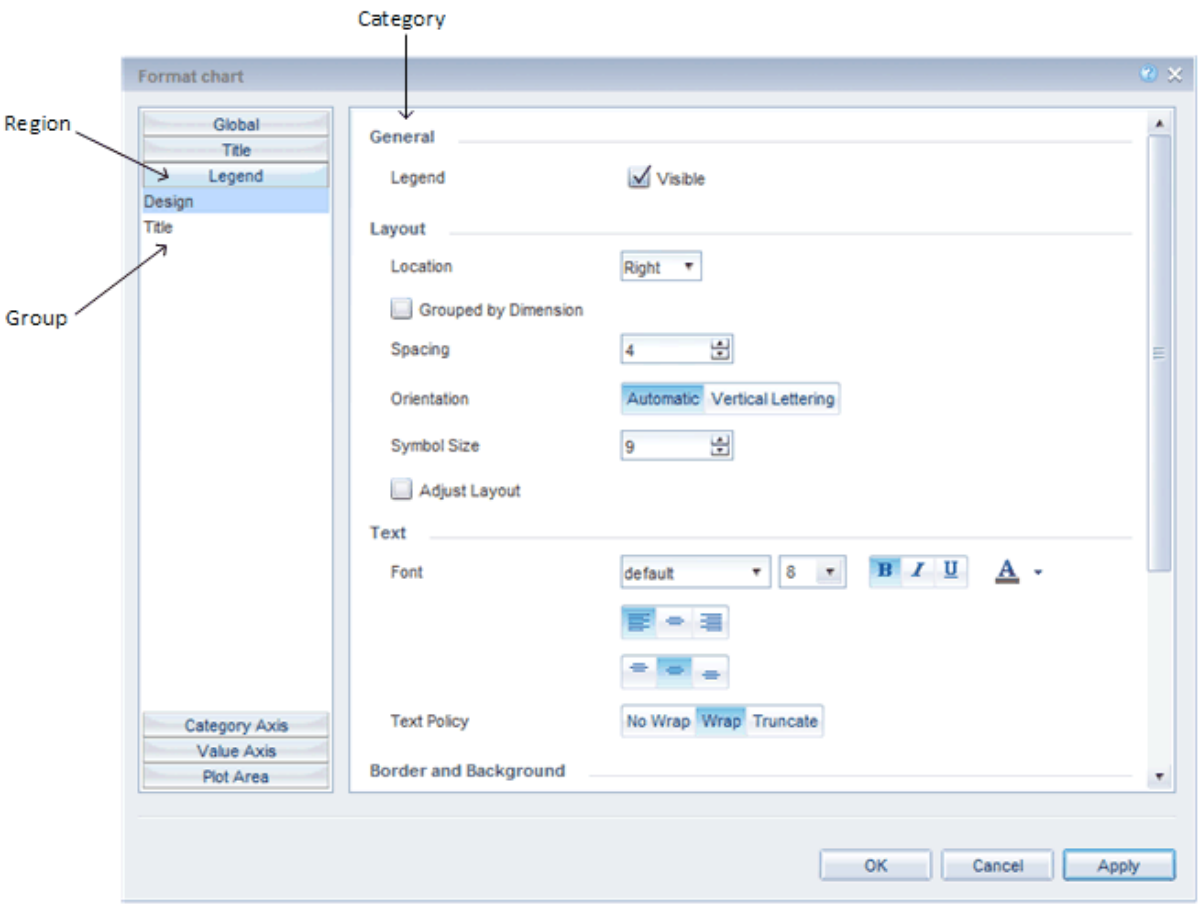
## 7.7 Getting the Rendering Settings

Returns the list of end-user settings supported by a type of visualization.

Web Intelligence needs this information to display the rendering settings returned by the service. You send the values actually set by the end-user by calling the service for rendering. Use these settings in the call Rendering the Visualization [page 20].

Settings are organized by regions such as title, legend, and axis. Each region contains several groups, and each group may contain several categories. Finally, a category contains a list of properties.

Settings are displayed in the *Format Custom Element* dialog box of Web Intelligence. Each region, group, and category of a setting is represented respectively as a left-pane button, section, and area of the dialog box. Web Intelligence assigns the appropriate widget to the setting according to its type (boolean, string, integer, and so on). A setting is displayed if at least one category is defined.




## Request

**URI:** `api/visualizations/<vizType>/settings`

**HTTP Method:** *GET*

**Request Headers:**

| Header | Required | Values |
|---|---|---|
| `Content-Type` | Yes | `application/json` |

**Request Parameters:**

| Parameter | Required | Data Type | Description | Parameter Type |
|---|---|---|---|---|
| `<vizType>` | Yes | String | The visualization type | Path |

| Parameter | Required | Data Type | Description | Parameter Type |
|---|---|---|---|---|
| `locale` | No | String | The user locale as a language and/or country string. For example:<br>• `en`<br>• `fr_FR` | Query |

## Response

**Format:** *JSON*

**Response Example:**

*GET* `api/visualizations/funnel/settings?locale=en`

Rendering settings are the following:

- Region is Legend.
- Group is Design.
- Design categories are General and Layout.
- General properties are isVisible, font and background color. The font property has 5 subproperties (name, size, bold, italics, and underline).
- Layout properties are location and spacing. Location has only 4 possible values. Spacing has a set of parameters that restrict its possible values.

```
{
 "regions":[
    {
     "id":"legend",
     "name":"Legend",
     "groups":[
     {
      "name":"Design",
      "categories":[
      {
       "name":"General",
       "description":"General design properties",
       "properties":[
          {
           "id":"isVisible",
           "name":"Visible ?",
           "description":"Indicates if title of the visualization is displayed
or not",
           "type":"boolean",
           "default":"true"
          },
          {
           "id":"font",
           "name":"Font",
           "description":"Font of the legend.",
           "type":"font",
           "default":"",
           "properties":[
              {
               "id":"name",
               "name":"Font name",
               "description":"Font name",
```

```
                "type":"string",
                "default":"arial"
               },
               {
                "id":"size",
                "name":"Font size",
                "description":"Font size",
                "type":"integer",
                "default":"8"
               },
               {
                "id":"bold",
                "name":"Is bold?",
                "description":"Is bold?",
                "type":"boolean",
                "default":"false"
               },
               {
                "id":"italic",
                "name":"Is italic?",
                "description":"Is italic?",
                "type":"boolean",
                "default":"false"
               },
               {
                "id":"underline",
                "name":"Is underline?",
                "description":"Is underline?",
                "type":"boolean",
                "default":"false"
               },
               {
                "id":"color",
                "name":"Font color",
                "description":"Font color",
                "type":"color",
                "default":"#ffffffff"
               }
             ]
           },
           {
            "id":"backgroundColor",
            "name":"Background color",
            "description":"Background color",
            "type":"color",
            "default":"#d0d0d0cc"
           }
         ]
        },
        {
         "name":"Layout",
         "description":"Layout properties",
         "properties":[
           {
            "id":"location",
            "name":"Location",
            "description":"Indicates where the title should be displayed",
            "type":"state",
            "default":"right",
            "choices":[
               {
                "id":"top",
                "name":"Top",
                "description":"Display title at the top."
               },
               {
                "id":"left",
                "name":"Left",
```

```
                "description":"Display title on the left."
               },
               {
                "id":"bottom",
                "name":"Bottom",
                "description":"Display title at the bottom."
               },
               {
                "id":"right",
                "name":"Right",
                "description":"Display title on the right."
               }
            ]
           },
           {
            "id":"spacing",
            "name":"Spacing",
            "description":"Indicates space between items",
            "type":"integer",
            "default":"4",
            "parameters":{
               "min":"2",
               "max":"6",
               "step":"1"
            }
           }
        ]
      }
     ]
    }
   ]
  }
 ]
}
```

Regions

| Output | Output Type | Description |
|---|---|---|
| `id` | String | The region identifier. Must be unique in the scope of the service. |
| `name` | String | The region name. Is localized. |
| `groups` | JSON array | The groups of a region |

Groups

| Output | Output Type | Description |
|---|---|---|
| `name` | String | The group name. Is localized. |
| `categories` | JSON array | The categories of a group |

Categories

| Output | Output Type | Description |
|---|---|---|
| `name` | String | The category name. Is localized. |
| `description` | String | The category description. Is localized. |
| `properties` | JSON array | The properties of a category |

Properties

| Output | Output Type | Description |
|---|---|---|
| `id` | String | The property identifier. Is unique in the region and must contain only characters from [a-z], [A-Z], [0-9], and [. \| - \| _]. |
| `name` | String | The property name. Is localized. |
| `description` | String | The property description. Is localized. |
| `type` | Enum | The property type. Is one of the following:<br>• `boolean`<br>• `color`<br>• `double`<br>• `font`<br>• `integer`<br>• `state`<br>• `string` |
| `default` | String | The property default value.<br>In the case of a color, default is a hexadecimal string.<br>In the case of a font, default is an empty string. |
| `parameters` | JSON object | The property parameters if any, for `int` and `double` type properties only. Specifies constraints to the property value.<br>If a property describes a size, the parameter `isUnit` is added and set to `true`. Hence, all parameters are expressed in metrics so that Web Intelligence can convert them into the end-user unit defined by its preferred viewing locale. |
| `properties` | JSON array | The subproperties of a property if any. The property itself has no value in this case. For example, the property of type `font` has a set of subproperties. |

### Properties: state

The values of a property of type `state` are restricted to a set of specific values. The default value must be one of the identifiers of these values. A property of type `state` cannot have parameters.

| Output | Output Type | Description |
| --- | --- | --- |
| `choices` | JSON array | The list of the possible values of the property |
| `id` | String | The value identifier. Is unique in the property |
| `name` | String | The value itself. Is localized. |
| `description` | String | The value description. Is localized. |

# 8 Adding Custom Formula in Web Intelligence Formula Language

The Web Intelligence formula language offers a broad range of functions and operators to calculate datasets retrieved from your data sources, including string manipulation, mathematical operations, Document and Data Sources properties. You can find these functions listed in the Functions section.

If the functions you require are not available or if you wish to optimize the processing of a formula by rewriting it, you can extend the formula language using external functions.

To achieve this, you must create a C++ external library and deploy it on the machine where Web Intelligence is running. For additional details, see the Overview of external functions document.

# 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.

| Related Documentation | Description |
|---|---|
| See the *SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer* on the SAP Help Portal. | The official guide for developing with the Web Intelligence RESTful Web Service SDK |
| See the *Business Intelligence platform RESTful Web Service Developer Guide* on the SAP Help Portal. | The official guide for developing with the BI platform RESTful Web Service SDK |

> **Note**
>
> Access to these guides is restricted to customers with a logon to the SAP Support Portal.

# 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.

| Related Documentation | Description |
|---|---|
| See the *SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer* on the SAP Help Portal. | The official guide for developing with the BI Semantic Layer RESTful Web Service SDK |
| See the *Business Intelligence platform RESTful Web Service Developer Guide* on the SAP Help Portal. | The official guide for developing with the BI platform RESTful Web Service SDK |

> **Note**
>
> Access to these guides is restricted to customers with a logon to the SAP Support Portal.

# 11 Developing Applications to Design and Administrate Universes

The BI Semantic Layer Java SDK 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`. For instructions on how to use the samples, see the BI Semantic Layer Java SDK Developer Guide.

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.

# 12 Creating a Data Access Driver with JavaBean

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 create your own JavaBean drivers and use them as data access drivers for your data sources. Refer to the Data Access Guide for more information.

## Writing a JavaBean

The JavaBean driver requires to write a JavaBean (ie a Java class) where some methods defined by the developer are exposed as stored procedures. To the data driver's consumer, it looks like a database that exposes only stored procedures.

For more details about how to use the JavaBean, refer to Section 5.5 of the Data Access Guide.

In functional terms the output of the driver is a set of independent tables (the results of the stored procedures), which are not manipulated using SQL. For the developer, SQL support does not need to be implemented, so such drivers are much easier and faster to write.

This JavaBean class must:

- Have a constructor that takes no parameters
- Have an initialize method, that takes some parameters to configure and initiate the connection to the data source
- Implement methods that are exposed as stored procedure if:
  - They are public methods
  - They return a java.sql.ResultSet
  - They have only scalar datatypes parameters: String, integer, float, double, ...

You can create several JavaBean drivers, but you need to write one class for each connection.

## Procedure

1. Include ConnectionServer.jar in your project in order to use the class Context. This JAR can be found in the folder:
   `<INSTALL DIR>\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\java\lib\`
2. Write a Java class constructor that does not need any argument. This constructor will give the name of your JavaBean. For example:

Sample Code

```
public class MyBean {
        public MyBean ()
   }
```

3. Implement the initialize method that connects to the data source.

Sample Code

```
public void initialize(Context context, String initString, Properties
properties) {
   // Add your own code here
}
```

This method must take the following arguments:

- Context: That contains the Connection Server configuration
- String: The string that the user can pass to the wizard
- Properties: Contains parameters like the authentication mode or the user's credentials

4. Implement the methods that retrieve and return dataset and are exposed as stored procedures. These methods must be public, returns a ResultSet and have non-scalar arguments.

Sample Code

```
public ResultSet myStoredProcMethod ( String stringArgument ) {
   // Add your own code here
}
```

5. Build the JAR file(s) for your JavaBean. Make sure to include potential dependencies.

To use this driver, in all server and client machines that need to access this data source:

- Create a dedicated folder in the `<INSTALL DIR>\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\dataAccess\connectionServer\javabean` folder.
- Copy the JAR file(s) in this folder.

## JavaBean Sample

A sample JavaBean is delivered with your SAP BI installation and can be used to query Google Cloud database. You need to generate a unique key from your Google account and use this key as the URL for authentication.

This JavaBean is installed when you install the BI platform. It is located in the

`<INSTALL DIR>\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\dataAccess\connectionServer\javabean\BigQuery` folder.

Its code is located in the folder:

`<INSTALL DIR>\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\dataAccess\connectionServer\DDK\examples\BigQueryJavabean\BigQuerySample`

You can use this code to create your own JavaBean, by following the indentation and comments provided in this sample.

# Important Disclaimers and Legal Information

## Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

## Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

## Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

## Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

## Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

© 2024 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.

THE BEST RUN SAP