Content

1  REST API Quick Reference ................................................................. 8
   1.1  Universe APIs ................................................................. 8
   1.2  Query APIs ................................................................. 9
   1.3  Document APIs ............................................................. 11
   1.4  Document Lifecycle APIs .................................................. 15
   1.5  Document Refresh and Scheduling APIs ............................... 16
   1.6  Report APIs ................................................................. 17
   1.7  Report Drill APIs ........................................................... 19
   1.8  Report Element APIs ....................................................... 20
   1.9  Data Provider APIs ........................................................ 23
   1.10 SAP BW Connection and BEx Query APIs ............................. 25
   1.11 Search Resource API ....................................................... 25

2  Document Version History .............................................................. 27

3  Introduction to the RESTful Web Service SDK User Guide ............... 32
   3.1  Audience ................................................................. 33
   3.2  Conventions in This Guide ............................................... 33
   3.3  Key Tasks ................................................................. 33
   3.4  Use Cases ................................................................. 34

4  Using the RESTful Web Service SDKs ............................................. 35
   4.1  Default Base URLs ........................................................ 36
   4.2  Getting Web Intelligence RESTful Web Service SDK Information ........................................................................... 37
   4.3  Supporting Multiple Languages ......................................... 38
   4.4  Dispensable Information in a Request Body ........................... 39
   4.5  cURL Examples ................................................................ 40
   4.6  HTTP Request Status ...................................................... 42
   4.7  Successful and Error Messages .......................................... 43

5  Using the RESTful Web Service Samples ........................................ 46
   5.1  About the BI Semantic Layer REST API Samples .................. 46
       To Deploy the Samples with Eclipse ..................................... 47
   5.2  About the Web Intelligence REST API Samples .................... 48
       To Deploy the LCMBIAR File ............................................. 49
       To Deploy the Samples with Eclipse ..................................... 50

6  Concepts and Schemas of the RESTful Web Service SDKs ............... 51
6.1 Change Source. .......................................................... 51
   Mappings Request and Response Body Schemas. ........................................ 53
   Strategies. .................................................................................. 55
   Compatibility Rules. ................................................................. 56

6.2 Charts. ........................................................................ 58
   Chart Types. ........................................................................... 58
   Chart Response Body Schema. ...................................................... 59

6.3 Date and Time Formats. .................................................. 92

6.4 Object Full Paths. .......................................................... 93

6.5 Parameters. ................................................................. 95
   Parameter Response Body Schemas. ..................................................... 95
   Lists of Values. ......................................................................... 98
   Default Values. ........................................................................ 101
   Previous Values. ...................................................................... 103
   Incompatible Contexts. .............................................................. 103
   Answer Request Body Schemas. .................................................... 104

6.6 Queries. ........................................................................ 107
   Query Specification Body Schema. ..................................................... 108
   Query Options. ........................................................................ 109
   Result Objects. .......................................................................... 110
   Sort Objects. ........................................................................... 111
   Query Filters. ........................................................................... 112
   Combined Queries. .................................................................... 125
   Query Workflows. .................................................................... 126
   The OData Query Service. .......................................................... 128

6.7 Report Structure. .......................................................... 131

6.8 Schedules. ...................................................................... 132

6.9 Universes. ...................................................................... 136

6.10 User Rights. ................................................................. 140

7 BI Semantic Layer REST API Reference. ................................. 141

7.1 Managing Universes with the BI Semantic Layer RESTful Web Service SDK. 141
   Getting the List of the Universes (Semantic Layer). .................................. 141
   Getting the Details of a Universe (Semantic Layer). ................................... 143
   Getting the Business Views of a Universe. ............................................. 146
   Getting the Query Capabilities of a Universe (Semantic Layer). ............... 149
   Getting the Link Groups of a Universe. ............................................... 151
   Getting the Object Parameters of a Universe. ....................................... 153

7.2 Managing Queries. .......................................................... 154
   Creating a Query. ...................................................................... 155
   Getting the List of Queries. .......................................................... 156
   Getting the Details of a Query. ....................................................... 157
### 7.3 Managing Parameters

- Getting the List of Parameters .................................................. 159
- Responding to Parameters ......................................................... 161
- Getting the Details of a Parameter ............................................... 173

### 7.4 Managing Query Results

- Accessing the OData Service ..................................................... 179
- Getting the OData Flow Metadata ............................................... 181
- Getting the OData Flow Content ............................................... 182
- OData Content - Getting All Data .............................................. 184
- OData Content - Getting the Row Count ..................................... 186
- OData Content - Getting the First Row ....................................... 186
- OData Content - Getting Property Content ................................. 187
- OData Content - Getting Property Raw Content ........................... 187
- OData Content - Getting Content After Offset ............................. 187
- OData Content - Getting the First N Rows ................................. 189
- OData Content - Getting the First N Rows After Offset ................. 190

### 8 Web Intelligence REST API Reference

#### 8.1 Managing Documents

- Creating a Document ............................................................... 193
- Getting the List of Documents ................................................. 195
- Getting the Details of a Document ........................................... 197
- Copying a Document .................................................................. 200
- Saving a Document .................................................................... 201
- Updating the State of a Document ............................................. 202
- Deleting a Document .................................................................. 204
- Getting the List of Values of a Data Object ................................. 205
- Getting the List of Values Under a Specific Data Object Value ....... 207
- Working with Document Input Controls ..................................... 209
- Working with Properties ............................................................ 219
- Working with ZIP files .............................................................. 222
- Exporting Documents ............................................................... 225
- Working with Snapshots ............................................................ 230
- Managing Styles, Formats, Fonts, Skins, andCharsets .................... 233
- Working with Color Palettes ...................................................... 249
- Managing Functions, Operators, and Variables .......................... 255
- Managing Links ......................................................................... 266
- Managing Alerters and Tracker Settings ..................................... 272
- Managing Attachments .............................................................. 284
- Adding a Cache Entry to a Document .......................................... 287

#### 8.2 Managing Reports

.......................................................... 288
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a Report.</td>
<td>289</td>
</tr>
<tr>
<td>Getting the List of Reports.</td>
<td>290</td>
</tr>
<tr>
<td>Getting the Details of a Report.</td>
<td>291</td>
</tr>
<tr>
<td>Moving a Report.</td>
<td>293</td>
</tr>
<tr>
<td>Copying a Report.</td>
<td>294</td>
</tr>
<tr>
<td>Updating the Properties of a Report.</td>
<td>295</td>
</tr>
<tr>
<td>Deleting a Report.</td>
<td>296</td>
</tr>
<tr>
<td>Getting the Map of a Report.</td>
<td>297</td>
</tr>
<tr>
<td>Exporting Reports.</td>
<td>298</td>
</tr>
<tr>
<td>Working with Input Controls of a Report.</td>
<td>306</td>
</tr>
<tr>
<td>Getting the Structure of a Report.</td>
<td>319</td>
</tr>
<tr>
<td>Updating the Structure of a Report.</td>
<td>320</td>
</tr>
<tr>
<td>Managing Data Filters in Reports.</td>
<td>325</td>
</tr>
<tr>
<td>8.3 Drilling on Report Data.</td>
<td>331</td>
</tr>
<tr>
<td>Getting the Drill Mode.</td>
<td>332</td>
</tr>
<tr>
<td>Updating the Drill Mode.</td>
<td>332</td>
</tr>
<tr>
<td>Enabling the Query Drill.</td>
<td>333</td>
</tr>
<tr>
<td>Disabling the Query Drill.</td>
<td>334</td>
</tr>
<tr>
<td>Getting Information on the Drill Hierarchies.</td>
<td>335</td>
</tr>
<tr>
<td>Getting the Free Drill Elements.</td>
<td>337</td>
</tr>
<tr>
<td>Creating a Drill Filter.</td>
<td>338</td>
</tr>
<tr>
<td>Getting the Drill Filters of a Report.</td>
<td>339</td>
</tr>
<tr>
<td>Getting the Details of a Drill Filter.</td>
<td>340</td>
</tr>
<tr>
<td>Updating a Drill Filter.</td>
<td>341</td>
</tr>
<tr>
<td>Removing a Drill Filter.</td>
<td>342</td>
</tr>
<tr>
<td>Performing a Drill.</td>
<td>343</td>
</tr>
<tr>
<td>Making a Snapshot of a Report in Drill Mode.</td>
<td>344</td>
</tr>
<tr>
<td>8.4 Managing Report Elements.</td>
<td>345</td>
</tr>
<tr>
<td>Creating a Report Element.</td>
<td>346</td>
</tr>
<tr>
<td>Getting the List of Report Elements.</td>
<td>349</td>
</tr>
<tr>
<td>Getting the Details of a Report Element.</td>
<td>352</td>
</tr>
<tr>
<td>Getting the Background of a Report Element.</td>
<td>360</td>
</tr>
<tr>
<td>Getting the Datapaths of a Report Element.</td>
<td>361</td>
</tr>
<tr>
<td>Getting the Dataset of a Report Element.</td>
<td>364</td>
</tr>
<tr>
<td>Updating a Report Element.</td>
<td>370</td>
</tr>
<tr>
<td>Inserting a Row or Column into a Table.</td>
<td>375</td>
</tr>
<tr>
<td>Merging or Splitting Cells of a Table.</td>
<td>376</td>
</tr>
<tr>
<td>Exporting a Report Element.</td>
<td>377</td>
</tr>
<tr>
<td>Deleting a Report Element.</td>
<td>381</td>
</tr>
<tr>
<td>Working with Axes.</td>
<td>383</td>
</tr>
<tr>
<td>Working with Rankings.</td>
<td>390</td>
</tr>
</tbody>
</table>
### Working with Custom Properties

- Managing Data Filters in Report Elements. ........................................ 394
- 8.5 Managing Data Providers. ...................................................... 404
  - Getting the List of Data Providers. ........................................ 405
  - Getting the Details of a Data Provider. .................................... 407
  - Adding a Data Provider. ...................................................... 410
  - Updating a Data Provider. .................................................... 413
  - Moving a Data Provider in a Document ................................. 421
  - Deleting a Data Provider. ..................................................... 422
  - Changing the Data Providers. ................................................. 423
  - Getting the Flow Count of a Data Provider ............................ 433
  - Getting the Details of a Flow. .............................................. 434
  - Getting the Samples of a Flow. ............................................ 436
  - Getting the Query Plan. ...................................................... 437
  - Updating the Query Plan. .................................................... 440
  - Getting the Query Specification. ......................................... 442
  - Updating the Query Specification. ....................................... 443

### Managing Connections for Free-Hand SQL Data Providers

- 8.6 Managing Connections for Free-Hand SQL Data Providers. ......... 445
  - Getting the List of Connections. .......................................... 445
  - Getting the Details of a Connection. ..................................... 447

### Managing Personal Data Providers

- 8.7 Managing Personal Data Providers. ....................................... 448
  - Uploading a Microsoft Excel File to the CMS Repository. ......... 448
  - Getting the List of Microsoft Excel Files. ............................ 450
  - Getting the Details of a Microsoft Excel File. ....................... 451
  - Updating a Microsoft Excel File to the CMS Repository. .......... 453
  - Deleting a Microsoft Excel File. ........................................... 454

### Managing SAP BW Connections and BEx Queries

- 8.8 Managing SAP BW Connections and BEx Queries. .................. 455
  - Getting the List of SAP BW Connections. ................................ 455
  - Getting the Details of an SAP BW Connection. ....................... 457
  - Browsing the Details of an SAP BW connection. ..................... 458
  - Getting the Outline of a BEx Query. ..................................... 460
  - Getting the Capabilities of a BEx Query. .............................. 462

### Refreshing Documents

- 8.9 Refreshing Documents. ...................................................... 463
  - Getting the Refresh Parameters of a Document. ....................... 464
  - Refreshing a Document. ..................................................... 468
  - Cancelling the Refresh of a Document. .................................. 485
  - Getting the Details of a Parameter. ..................................... 486

### Refreshing Data Providers

- 8.10 Refreshing Data Providers. ................................................. 489
  - Getting the Refresh Parameters of a Data Provider. ............... 489
  - Refreshing a Data Provider. ............................................... 490

### Scheduling Documents

- 8.11 Scheduling Documents. .................................................... 491
Getting the List of Schedules. .................................................. 491
Getting the Details of a Schedule. ................................................ 493
Adding a Schedule. .............................................................. 495
Deleting a Schedule. ............................................................. 499

8.12 Searching for Resources. ..................................................... 499
Getting Resources. .............................................................. 500

8.13 Managing Universes with the Web Intelligence RESTful Web Service SDK. ................................................. 503
Getting the List of Universes (Web Intelligence). ........................... 503
Getting the Details of a Universe (Web Intelligence). ....................... 505
Getting the Query Capabilities of a Universe (Web Intelligence). ........... 508
1  REST API Quick Reference

An overview of the REST APIs for Web Intelligence and the BI Semantic Layer.

REST APIs are grouped by functional area with descriptions and links to more information.

- **Universe APIs [page 8]**
  The REST APIs to work with universes.

- **Query APIs [page 9]**
  The REST APIs to work with queries and get query results.

- **Document APIs [page 11]**
  The REST APIs to manage Web Intelligence documents.

- **Document Lifecycle APIs [page 15]**
  The REST APIs to manage the lifecycle of a Web Intelligence document.

- **Document Refresh and Scheduling APIs [page 16]**
  The REST APIs to refresh and schedule Web Intelligence documents.

- **Report APIs [page 17]**
  The REST APIs to work with reports in Web Intelligence documents.

- **Report Drill APIs [page 19]**
  The REST APIs to drill through reports in a Web Intelligence document.

- **Report Element APIs [page 20]**
  The REST APIs to work with the elements of a report.

- **Data Provider APIs [page 23]**
  The REST APIs to work with queries in reports.

- **SAP BW Connection and BEx Query APIs [page 25]**
  The REST APIs to manage SAP BW connections and BEx queries.

- **Search Resource API [page 25]**
  The REST API to search for resources on the CMS repository.

## 1.1  Universe APIs

The REST APIs to work with universes.

The tables below indicate the 4.0 and/or 4.1 release in which support for the API was introduced.

**Base URL:** http://<server_name>:6405/biprws/sl/v1

**Path parameters:** `<universeID>` is the universe identifier retrieved from the list of universes.
Table 1:

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the List of the Universes (Semantic Layer) [page 141]</td>
<td>GET</td>
<td>/universes</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Getting the Details of a Universe (Semantic Layer) [page 143]</td>
<td>GET</td>
<td>/universes/&lt;universeID&gt;</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Getting the Business Views of a Universe [page 146]</td>
<td>GET</td>
<td>/universes/&lt;universeID&gt;/businessviews</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Getting the Query Capabilities of a Universe (Semantic Layer) [page 149]</td>
<td>GET</td>
<td>/universes/&lt;universeID&gt;/capabilities</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Getting the Link Groups of a Universe [page 151]</td>
<td>GET</td>
<td>/universes/&lt;universeID&gt;/linkgroups</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Getting the Object Parameters of a Universe [page 153]</td>
<td>GET</td>
<td>/universes/&lt;universeID&gt;/prompts</td>
<td>4.1 SP6</td>
</tr>
</tbody>
</table>

**Base URL:** http://<server_name>:6405/biprws/raylight/v1

**Path parameter:** `<universeID>` is the universe identifier retrieved from the list of universes.

Table 2:

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the List of Universes (Web Intelligence) [page 503]</td>
<td>GET</td>
<td>/universes</td>
<td>4.0 SP5</td>
</tr>
<tr>
<td>Getting the Details of a Universe (Web Intelligence) [page 505]</td>
<td>GET</td>
<td>/universes/&lt;universeID&gt;</td>
<td>4.0 SP5</td>
</tr>
<tr>
<td>Getting the Query Capabilities of a Universe (Web Intelligence) [page 508]</td>
<td>GET</td>
<td>/universes/&lt;universeID&gt;/capabilities</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>

1.2 Query APIs

The REST APIs to work with queries and get query results.

The tables below indicate the 4.0 and/or 4.1 release in which support for the API was introduced.

**Base URL:** http://<server_name>:6405/biprws/sl/v1

**Path parameters:**

- `<queryID>`: query statement identifier retrieved from the list of queries
- `<parameterID>`: parameter identifier
- `<flowName>`: query flow name
- `<rowIndex>`: row index of the query flow
• `<fieldName>`: name of an object of the flow

**Query parameters**: see the detailed description of the related API.

Table 3:

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the List of Parameters [page 159]</td>
<td>GET</td>
<td>/queries/&lt;queryID&gt;/parameters</td>
<td>4.1 SP3</td>
</tr>
<tr>
<td>Responding to Parameters [page 161]</td>
<td>PUT</td>
<td>/queries/&lt;queryID&gt;/parameters</td>
<td>4.1 SP3</td>
</tr>
<tr>
<td>Getting the Details of a Parameter [page 173]</td>
<td>GET</td>
<td>PUT /queries/&lt;queryID&gt;/parameters/&lt;parameterID&gt;</td>
<td>4.2</td>
</tr>
</tbody>
</table>

Table 4:

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a Query [page 155]</td>
<td>POST</td>
<td>/queries</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Deleting a Query [page 158]</td>
<td>DELETE</td>
<td>/queries/&lt;queryID&gt;</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Getting the Details of a Query [page 157]</td>
<td>GET</td>
<td>/queries/&lt;queryID&gt;</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Getting the List of Queries [page 156]</td>
<td>GET</td>
<td>/queries</td>
<td>4.1 SP2</td>
</tr>
</tbody>
</table>

Table 5:

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessing the OData Service [page 179]</td>
<td>GET</td>
<td>/queries/&lt;queryID&gt;/data.svc/</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Getting the OData Flow Metadata [page 181]</td>
<td>GET</td>
<td>/queries/&lt;queryID&gt;/data.svc/$metadata</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Getting the OData Flow Content [page 182]</td>
<td>GET</td>
<td>/queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>OData Content - Getting the Row Count [page 186]</td>
<td>GET</td>
<td>/queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;/$count</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>OData Content - Getting the First Row [page 186]</td>
<td>GET</td>
<td>/queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;/&lt;rowIndex&gt;</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>OData Content - Getting Property Content [page 187]</td>
<td>GET</td>
<td>/queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;/&lt;rowIndex&gt;/&lt;fieldName&gt;</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>OData Content - Getting Property Raw Content [page 187]</td>
<td>GET</td>
<td>/queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;/&lt;rowIndex&gt;/&lt;fieldName&gt;/$value</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>OData Content - Getting Content After Offset [page 187]</td>
<td>GET</td>
<td>/queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;? $skip=&lt;offset&gt;</td>
<td>4.1 SP2</td>
</tr>
</tbody>
</table>
1.3 Document APIs

The REST APIs to manage Web Intelligence documents.

The tables below indicate the 4.0 and/or 4.1 release in which support for the API was introduced.

**Base URL:** http://<server_name>:6405/biprws/raylight/v1

**Path parameters:**
- `<documentID>`: document identifier retrieved from the list of documents
- `<dataObjectID>`: data object identifier retrieved from the definition of the input control to which the data object is assigned
- `<alerterID>`: alerter identifier retrieved from the list of alerters
- `<inputControlID>`: input control identifier retrieved from the list of input controls
- `<linkID>`: link identifier retrieved from the list of links
- `<paletteID>`: custom palette identifier retrieved from the list of palettes
- `<styleID>`: style identifier from the list of styles of the document
- `<variableID>`: variable identifier retrieved from the list of variables of the document
- `<zipID>`: ZIP file identifier obtained by creating the zip file

**Query parameters:** see the detailed description of the related API.

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a Document</td>
<td>POST</td>
<td>/documents</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Copying a Document</td>
<td>POST</td>
<td>/documents</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the List of Documents</td>
<td>GET</td>
<td>/documents</td>
<td>4.0 SP5</td>
</tr>
<tr>
<td>Getting the Details of a Document</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;</td>
<td>4.0 SP5</td>
</tr>
<tr>
<td>Saving a Document</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;</td>
<td>4.1 SP4</td>
</tr>
<tr>
<td>Deleting a Document</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>
### Exporting a Document in Listing Mode [page 226]

**GET** /documents/<documentID>  
**Since** 4.0 SP5 | 4.1

### Exporting a Document as a Series of Pages [page 228]

**GET** /documents/<documentID>/pages  
**Since** 4.0 SP5 | 4.1

### Adding an Attachment [page 285]

**POST** /documents/<documentID>/attachments  
**Since** 4.0 SP6 | 4.1

### Getting the List of Attachments [page 284]

**GET** /documents/<documentID>/attachments  
**Since** 4.0 SP6 | 4.1

### Adding a Cache Entry to a Document [page 287]

**POST** /documents/<documentID>/cache  
**Since** 4.0 SP6 | 4.1

### Getting the CSS of a Document [page 244]

**GET** /documents/<documentID>/css  
**Since** 4.0 SP6 | 4.1

### Updating the CSS of a Document [page 245]

**PUT** /documents/<documentID>/css  
**Since** 4.0 SP6 | 4.1

### Getting the List of Values of a Data Object [page 205]

**GET** /documents/<documentID>/dataobjects/<dataObjectID>/lov  
**Since** 4.1 SP3

### Getting the List of Values Under a Specific Data Object Value [page 207]

**PUT** /documents/<documentID>/dataobjects/<dataObjectID>/lov  
**Since** 4.1 SP3

### Getting Custom Formats for Numbers [page 235]

**GET** /documents/<documentID>/formats  
**Since** 4.0 SP6 | 4.1

### Getting the Properties of a Document [page 219]

**GET** /documents/<documentID>/properties  
**Since** 4.0 SP6 | 4.1

### Updating the Properties of a Document [page 221]

**PUT** /documents/<documentID>/properties  
**Since** 4.0 SP6 | 4.1

### Adding an Alerter [page 275]

**POST** /documents/<documentID>/alerters  
**Since** 4.0 SP6 | 4.1

### Getting the List of Alerters [page 273]

**GET** /documents/<documentID>/alerters  
**Since** 4.0 SP6 | 4.1

### Getting the Details of an Alerter [page 274]

**GET** /documents/<documentID>/alerters/<alerterID>  
**Since** 4.0 SP6 | 4.1

### Editing an Alerter [page 277]

**PUT** /documents/<documentID>/alerters/<alerterID>  
**Since** 4.0 SP6 | 4.1

### Deleting an Alerter [page 278]

**DELETE** /documents/<documentID>/alerters/<alerterID>  
**Since** 4.0 SP6 | 4.1

### Getting the Charsets [page 246]

**GET** /configuration/charsets  
**Since** 4.0 SP6 | 4.1

---

**Table 7: Alerters**

**Table 8: Document Information**
### Table 9: Document Input Controls

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the Chart Types [page 248]</td>
<td>GET</td>
<td>/configuration/visualizations</td>
<td>4.1SP1</td>
</tr>
<tr>
<td>Getting the Default Color Palettes [page 249]</td>
<td>GET</td>
<td>/configuration/palettes</td>
<td>4.1SP1</td>
</tr>
<tr>
<td>Getting the Configuration Formats [page 234]</td>
<td>GET</td>
<td>/configuration/formats</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Font Mappings [page 236]</td>
<td>GET</td>
<td>/configuration/fontmappings</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Formula Engine Functions [page 256]</td>
<td>GET</td>
<td>/configuration/functions</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Formula Engine Operators [page 257]</td>
<td>GET</td>
<td>/configuration/operators</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Report Skins [page 237]</td>
<td>GET</td>
<td>/configuration/skins</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>

### Table 10: Links

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding a Link [page 269]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/links</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Action</td>
<td>Method</td>
<td>URL</td>
<td>Since</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>--------</td>
<td>------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Getting the Links of a Document [page 267]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/links</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Details of a Link [page 268]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/links/&lt;linkID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Editing a Link [page 270]</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/links/&lt;linkID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Deleting a Link [page 271]</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;/links/&lt;linkID&gt;</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>

**Table 11: Custom Palettes**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a Custom Palette [page 253]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/palettes</td>
<td>4.1 SP5</td>
</tr>
<tr>
<td>Getting the List of Custom Palettes [page 251]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/palettes</td>
<td>4.1 SP5</td>
</tr>
<tr>
<td>Getting the Details of a Custom Palette [page 252]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/palettes/&lt;paletteID&gt;</td>
<td>4.1 SP5</td>
</tr>
<tr>
<td>Updating the Definition of a Custom Palette [page 254]</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/palettes/&lt;paletteID&gt;</td>
<td>4.1 SP5</td>
</tr>
<tr>
<td>Deleting a Custom Palette [page 255]</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;/palettes/&lt;paletteID&gt;</td>
<td>4.1 SP5</td>
</tr>
</tbody>
</table>

**Table 12: Styles**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the Styles of a Document [page 239]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/styles</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Adding a Style to a Document [page 241]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/styles</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Details of a Style [page 240]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/styles/&lt;styleID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Updating a Style [page 242]</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/styles/&lt;styleID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Deleting a Style [page 244]</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;/styles/&lt;styleID&gt;</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>

**Table 13: Trackers**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the Tracker Settings [page 279]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/tracker</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Creating the Tracker Settings [page 280]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/tracker</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>
1.4  Document Lifecycle APIs

The REST APIs to manage the lifecycle of a Web Intelligence document.

The table below indicates the 4.0 and/or 4.1 release in which support for the API was introduced.

**Base URL:** http://<server_name>:6405/biprws/roylight/v1

**Path parameter:** `<documentID>` is the document identifier retrieved from the list of documents.

**Query parameter:** `<snapshotID>` is the snapshot identifier retrieved from the list of snapshots.

### Table 14: Variables

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editing the Tracker Settings [page 282]</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/tracker</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Deleting the Tracker Settings [page 283]</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;/tracker</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>

### Table 15: ZIP Files

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deleting a ZIP File [page 225]</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;/zips/&lt;zipID&gt;</td>
<td>4.1 SP3</td>
</tr>
</tbody>
</table>
The REST APIs to refresh and schedule Web Intelligence documents.

The tables below indicate the 4.0 and/or 4.1 release in which support for the API was introduced.

**Base URL:** http://<server_name>:6405/biprws/raflight/v1

### Path parameters:
- `<documentID>`: document identifier retrieved from the list of documents
- `<parameterID>`: parameter identifier retrieved from the list of refresh parameters
- `<scheduleID>`: schedule identifier retrieved from the list of schedules

**Query parameters:** see the detailed description of the related API.

### Table 17: Document Refresh

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Request</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the Refresh Parameters of a Document</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/parameters</td>
<td>4.0 SP5</td>
</tr>
<tr>
<td>Refreshing a Document</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/parameters</td>
<td>4.0 SP5</td>
</tr>
<tr>
<td>Getting the Details of a Parameter</td>
<td>GET</td>
<td>PUT</td>
<td>4.2</td>
</tr>
<tr>
<td>Cancelling the Refresh of a Document</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/parameters/execution?cancel=&lt;mode&gt;</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>

### Table 18: Document Scheduling

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding a Schedule</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/schedules</td>
<td>4.0 SP5</td>
</tr>
</tbody>
</table>
### 1.6 Report APIs

The REST APIs to work with reports in Web Intelligence documents.

The tables below indicate the 4.0 and/or 4.1 release in which support for the API was introduced.

**Base URL:** `http://<server_name>:6405/biprws/raylight/v1`

**Path parameters:**
- `<documentID>`: document identifier retrieved from the list of documents
- `<reportID>`: report identifier retrieved from the list of reports
- `<pageIndex>`: number of the page to export
- `<inputControlID>`: input control identifier retrieved from the list of input controls

**Query parameters:** see the detailed description of the related API.

#### Table 19: Reports

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a Report [page 289]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/reports</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the List of Reports [page 290]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports</td>
<td>4.0 SP5</td>
</tr>
<tr>
<td>Getting the Details of a Report [page 291]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;</td>
<td>4.0 SP5</td>
</tr>
<tr>
<td>Copying a Report [page 294]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/reports</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Moving a Report [page 293]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/reports?fromId=&lt;fromID&gt;&amp;toId=&lt;toID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Updating the Properties of a Report [page 295]</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Deleting a Report [page 296]</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Exporting a Report in Listing Mode [page 299]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;</td>
<td>4.0 SP5</td>
</tr>
<tr>
<td>Action</td>
<td>Method</td>
<td>URL</td>
<td>Since</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>--------</td>
<td>------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Exporting a Report as a Series of Pages</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/pages</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Exporting a Page of a Report</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/pages/&lt;pageIndex&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Map of a Report</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/map</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Structure of a Report</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/specification</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Updating the Structure of a Report</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/specification</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>

**Table 20: Data Filters of a Report**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a Data Filter</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/datafilter</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Getting the Details of a Data Filter</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/datafilter</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Updating a Data Filter</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/datafilter</td>
<td>4.1 SP2</td>
</tr>
<tr>
<td>Deleting a Data Filter</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/datafilter</td>
<td>4.1 SP2</td>
</tr>
</tbody>
</table>

**Table 21: Input Controls**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the Input Controls of a Report</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/inputcontrols</td>
<td>4.1 SP3</td>
</tr>
<tr>
<td>Moving an Input Control from a Document</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/inputcontrols?fromId=&lt;fromID&gt;</td>
<td>4.1 SP6</td>
</tr>
<tr>
<td>Getting the Definition of an Input Control</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/inputcontrols/&lt;inputControlID&gt;</td>
<td>4.1 SP3</td>
</tr>
<tr>
<td>Updating the Definition of an Input Control</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/inputcontrols/&lt;inputControlID&gt;</td>
<td>4.1 SP3</td>
</tr>
<tr>
<td>Adding an Input Control</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/inputcontrols</td>
<td>4.1 SP3</td>
</tr>
<tr>
<td>Deleting an Input Control</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/inputcontrols/&lt;inputControlID&gt;</td>
<td>4.1 SP3</td>
</tr>
<tr>
<td>Getting the Selection of an Input Control</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/inputcontrols/&lt;inputControlID&gt;/selection</td>
<td>4.1 SP3</td>
</tr>
</tbody>
</table>
### 1.7 Report Drill APIs

The REST APIs to drill through reports in a Web Intelligence document.

The tables below indicate the 4.0 and/or 4.1 release in which support for the API was introduced.

**Base URL:** http://<server_name>:6405/biprws/raylight/v1

**Path parameters:**
- `<documentID>`: document identifier retrieved from the list of documents
- `<reportID>`: report identifier retrieved from the list of reports
- `<filterID>`: drill filter identifier retrieved from the list of the drill filters

#### Table 22: Report Drill

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting the Selection of an Input Control [page 315]</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/inputcontrols/&lt;inputControlID&gt;/selection</td>
<td>4.1 SP3</td>
</tr>
<tr>
<td>Deleting the Selection of an Input Control [page 316]</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/inputcontrols/&lt;inputControlID&gt;/selection</td>
<td>4.1 SP3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the Drill Mode [page 332]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Enabling the Query Drill [page 333]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Updating the Drill Mode [page 332]</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Disabling the Query Drill [page 334]</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Free Drill Elements [page 337]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller/drillelements</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting Information on the Drill Hierarchies [page 335]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller/hierarchies</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Performing a Drill [page 343]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller/instructions</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Making a Snapshot of a Report in Drill Mode [page 344]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller/snapshot</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>
Table 23: Drill Filters

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a Drill Filter [page 338]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller/filters</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Drill Filters of a Report [page 339]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller/filters</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Details of a Drill Filter [page 340]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller/filters/&lt;filterID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Updating a Drill Filter [page 341]</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller/filters/&lt;filterID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Removing a Drill Filter [page 342]</td>
<td>DELETE</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/driller/filters/&lt;filterID&gt;</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>

1.8 Report Element APIs

The REST APIs to work with the elements of a report.

The tables below indicate the 4.0 and/or 4.1 release in which support for the API was introduced.

**Base URL:** http://<server_name>:6405/bipws/raylight/v1

**Path parameters:**
- `<documentID>`: document identifier retrieved from the list of documents
- `<reportID>`: report identifier retrieved from the list of reports
- `<elementID>`: table identifier retrieved from the list of elements of a report
- `<axisID>`: axis identifier when the report element is a section, table, or chart
- `<propertyKey>`: key of the custom property of the report element
- `<breakID>`: break identifier retrieved from the details of the report element

**Query parameters:** see the detailed description of the related API.

Table 24: Report Elements

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the List of Report Elements [page 349]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/elements</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Action</td>
<td>Method</td>
<td>URL</td>
<td>Since</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>Inserting a Row or Column into a Table [page 375]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/reports/&lt;reportID&gt;/elements/&lt;elementID&gt;?strip=&lt;strip&gt;&amp;position=&lt;position&gt;</td>
<td>4.1SP6</td>
</tr>
</tbody>
</table>

Table 25: Custom Properties of a Report Element

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
</table>

Table 26: Data Filters of a Report Element

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Method</td>
<td>URL</td>
<td>Since</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
<td>--------</td>
</tr>
</tbody>
</table>

Table 27: Rankings of a Report Element

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
</table>

Table 28: Expressions of a Report Element Axis

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
</table>

Table 29: Table Axes and Breaks

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
</table>
# 1.9 Data Provider APIs

The REST APIs to work with queries in reports.

The tables below indicate the 4.0 and/or 4.1 release in which support for the API was introduced.

**Base URL:** http://<server_name>:6405/biprws/raylight/v1

**Path parameters:**
- `<documentID>`: document identifier retrieved from the list of documents
- `<dataProviderID>`: identifier of the data provider available for a document
- `<flowID>`: flow identifier that you can know from the flow count
- `<spreadsheetID>`: Microsoft Excel file identifier retrieved from the list of files
- `<connectionID>`: connection identifier retrieved from the list of connections

**Query parameters:** see the detailed description of the related API.

<table>
<thead>
<tr>
<th>Table 30: Data Providers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action</strong></td>
</tr>
<tr>
<td>Getting the List of Data Providers [page 405]</td>
</tr>
<tr>
<td>Getting the Details of a Data Provider [page 407]</td>
</tr>
<tr>
<td>Adding a Data Provider [page 410]</td>
</tr>
<tr>
<td>Moving a Data Provider in a Document [page 421]</td>
</tr>
<tr>
<td>Updating a Data Provider [page 413]</td>
</tr>
<tr>
<td>Deleting a Data Provider [page 422]</td>
</tr>
<tr>
<td>Getting the Flow Count of a Data Provider [page 433]</td>
</tr>
<tr>
<td>Getting the Details of a Flow [page 434]</td>
</tr>
<tr>
<td>Getting the Samples of a Flow [page 436]</td>
</tr>
<tr>
<td>Getting the Query Plan [page 437]</td>
</tr>
<tr>
<td>Updating the Query Plan [page 440]</td>
</tr>
<tr>
<td>Getting the Query Specification [page 442]</td>
</tr>
<tr>
<td>Action</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Updating the Query Specification [page 443]</td>
</tr>
</tbody>
</table>

Table 31: Personal Data Providers

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uploading a Microsoft Excel File to the CMS Repository [page 448]</td>
<td>POST</td>
<td>/spreadsheets</td>
<td>4.1 SP5</td>
</tr>
<tr>
<td>Getting the List of Microsoft Excel Files [page 450]</td>
<td>GET</td>
<td>/spreadsheets</td>
<td>4.1 SP5</td>
</tr>
<tr>
<td>Getting the Details of a Microsoft Excel File [page 451]</td>
<td>GET</td>
<td>/spreadsheets/&lt;spreadsheetID&gt;</td>
<td>4.1 SP5</td>
</tr>
<tr>
<td>Updating a Microsoft Excel File to the CMS Repository [page 453]</td>
<td>PUT</td>
<td>/spreadsheets/&lt;spreadsheetID&gt;</td>
<td>4.2</td>
</tr>
<tr>
<td>Deleting a Microsoft Excel File [page 454]</td>
<td>DELETE</td>
<td>/spreadsheets/&lt;spreadsheetID&gt;</td>
<td>4.1 SP6</td>
</tr>
</tbody>
</table>

Table 32: Free-Hand SQL Data Providers

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the List of Connections [page 445]</td>
<td>GET</td>
<td>/connections</td>
<td>4.1 SP5</td>
</tr>
<tr>
<td>Getting the Details of a Connection [page 447]</td>
<td>GET</td>
<td>/connections/&lt;connectionID&gt;</td>
<td>4.1 SP5</td>
</tr>
</tbody>
</table>

Table 33: Change Data Sources

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the Possible Object Mappings Using the Default Strategies [page 423]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/dataproviders/&lt;dataProviderID&gt;/mappings?originDataproviderIds=&lt;DP1ID&gt;,&lt;DP2ID&gt;&amp;targetDatasourceId=&lt;DatasourceID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Changing the Data Objects of a Data Provider [page 428]</td>
<td>POST</td>
<td>/documents/&lt;documentID&gt;/dataproviders/&lt;dataProviderID&gt;/mappings?originDataproviderIds=&lt;DP1ID&gt;,&lt;DP2ID&gt;&amp;targetDatasourceId=&lt;DatasourceID&gt;</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>
Table 34: Refresh Data Providers

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the Refresh Parameters of a Data Provider [page 489]</td>
<td>GET</td>
<td>/documents/&lt;documentID&gt;/dataproviders/&lt;dataProviderID&gt;/parameters</td>
<td>4.1 SP5</td>
</tr>
<tr>
<td>Refreshing a Data Provider [page 490]</td>
<td>PUT</td>
<td>/documents/&lt;documentID&gt;/dataproviders/&lt;dataProviderID&gt;/parameters</td>
<td>4.1 SP5</td>
</tr>
</tbody>
</table>

### 1.10 SAP BW Connection and BEx Query APIs

The REST APIs to manage SAP BW connections and BEx queries.

The table below indicates the 4.0 and/or 4.1 release in which support for the API was introduced.

**Base URL:** http://<server_name>:6405/biprws/raylight/v1

**Path parameter:** <bwConnectionID> is the connection identifier retrieved from the list of SAP BW connections.

Table 35:

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the List of SAP BW Connections [page 455]</td>
<td>GET</td>
<td>/bwconnections</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Details of an SAP BW Connection [page 457]</td>
<td>GET</td>
<td>/bwconnections/&lt;bwConnectionID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Browsing the Details of an SAP BW connection [page 458]</td>
<td>PUT</td>
<td>/bwconnections/&lt;bwConnectionID&gt;</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Outline of a BEx Query [page 460]</td>
<td>PUT</td>
<td>/bwconnections/&lt;bwConnectionID&gt;/outline</td>
<td>4.0 SP6</td>
</tr>
<tr>
<td>Getting the Capabilities of a BEx Query [page 462]</td>
<td>PUT</td>
<td>/bwconnections/&lt;bwConnectionID&gt;/capabilities</td>
<td>4.0 SP6</td>
</tr>
</tbody>
</table>

### 1.11 Search Resource API

The REST API to search for resources on the CMS repository.

The table below indicates the 4.0 and/or 4.1 release in which support for the API was introduced.

**Base URL:** http://<server_name>:6405/biprws/raylight/v1
<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>URL</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting Resources [page 500]</td>
<td>POST</td>
<td>/searches</td>
<td>4.1 SP6</td>
</tr>
</tbody>
</table>
2 Document Version History

The following tables provide an overview of the document changes.

Table 37:

SAP BusinessObjects RESTful Web Service SDK for Web Intelligence and the BI Semantic Layer 4.2 SP1 (December 2015)

The following updates have been made to this document:

- Links refer to merged dimensions, see Managing Links [page 266]
- Offset added to Example - Mail Destination for a Hourly Schedule [page 496]
- Carriage returns added to Uploading a Microsoft Excel File to the CMS Repository [page 448]

Table 38:

SAP BusinessObjects RESTful Web Service SDK for Web Intelligence and the BI Semantic Layer 4.2 (November 2015)

New supported features in SAP BusinessObjects BI Semantic Layer RESTful Web Service SDK:

- Retrieving the details of a parameter through its identifier

New supported features in SAP BusinessObjects Web Intelligence RESTful Web Service SDK:

- Retrieving the details of a parameter through its identifier
- Replacing a Microsoft Excel file used as data provider with another one
- Updating a Microsoft Excel file to the CMS repository
- Getting and setting if a measure expression of a data provider must return values with a higher precision
- Searching for resources of type documents, Microsoft Excel files, and CSV files in the CMS repository
- Support of dotted and dashed lines in Line charts
- Timestamp added to the schedule details

The /about call and JSON required syntax are now documented in the chapter Using the RESTful Web Service SDKs.
Table 39:

SAP BusinessObjects RESTful Web Service SDK for Web Intelligence and the BI Semantic Layer 4.1 SP6 (June 2015)

New supported features in SAP BusinessObjects BI Semantic Layer RESTful Web Service SDK:

- UNV universes
- BOTH and EXCEPT operators in constant comparison filters
- Ranking filters
- Object parameters of UNIX universes (GET)
- New business object attribute (hasLov) in universe outlines
- New parameter attributes (keepLastValues, mandatorySearch) and new intervalSize values
- In the case of a parameter of type context, incompatible context values are retrieved with the context details

New supported features in SAP BusinessObjects Web Intelligence RESTful Web Service SDK:

- New business object attribute (hasLov) in universe outlines
- New parameter attributes (keepLastValues, mandatorySearch) and new intervalSize values
- In the case of a parameter of type context, incompatible context values are retrieved with the context details
- Use of custom strategies to change data sources
- Managing the input controls of a document
- Deleting a Microsoft Excel file from the CMS repository
- Search for resources of the CMS repository (/searches)
- Updates of connections and SQL scripts in free-hand SQL providers
- Exporting the last page of a report
- Inserting or deleting a row or column of a table for a given cell
- Merging or splitting cells of a table
- Changing the type of a report element of type table or visualization (Turn Into function)
- Updating or deleting the sorts of an axis of a section or visualization
- Managing rankings in a visualization

New samples that demonstrate how to use the Web Intelligence REST APIs are also provided.
### Table 40:
SAP BusinessObjects RESTful Web Service SDK for Web Intelligence and the BI Semantic Layer 4.1 SP5 (November 2014)

New feature for SAP BusinessObjects BI Semantic Layer RESTful Web Service SDK:
- Support of denied master view of a UNX universe

New features for SAP BusinessObjects Web Intelligence RESTful Web Service SDK:
- Support charts (create, get details, update, delete)
- Export charts as images
- Refresh data providers
- Upload Microsoft Excel files to the CMS repository
- Use Microsoft Excel files as data providers (get list, details, and add)
- Support free-hand SQL data providers (get, create, update, delete)
- Support connections for free-hand SQL data providers (get connections, get connection details)
- Get schedules with full details (destination, recurrence, parameters, and group servers)
- Provide new types of schedule recurrences (Weekly, Nth of each month, First Monday of each month, Last day of each month, X days of Nth Week of each month, and Calendar)
- Export raw values of documents, reports, and report elements
- Support of denied master view of a UNX universe
- Get the background image of a report element
- Use of custom palettes
- Change in the default palette definition

### Table 41:
SAP BusinessObjects RESTful Web Service SDK for Web Intelligence and the BI Semantic Layer 4.1 SP4 (June 2014)

New features for SAP BusinessObjects Web Intelligence RESTful Web Service SDK:
- Save a Web Intelligence document on the user machine
- Add, edit or delete a grouping variable of a Web Intelligence document
- Get the query plan of a data provider and update it with your own SQL
### Table 42:
**SAP BusinessObjects RESTful Web Service SDK for Web Intelligence and the BI Semantic Layer 4.1 SP3 (March 2014)**

New features for SAP BusinessObjects Web Intelligence RESTful Web Service SDK:

- Get the datapaths and the dataset of a report element
- Create report elements as tables
- Manage the ranking of tables or sections
- Manage expressions, sortings, and breaks of a table axis
- Manage custom properties of report elements
- Manage input controls of a report (get, create, update, delete)
- Export documents as ZIP files
- Get the list of values of a data object
- Add a BMP image as an attachment of a document
- Specify the file name when exporting a report in listing mode or a page of a report as a ZIP file
- Specify the reference when getting the details of a report element
- Specify the reference or datapath when exporting a report element

Second release of the document for SAP BusinessObjects BI Semantic Layer RESTful Web Service SDK. Support of the following new features:

- Get contexts and parameters of unx universes
- Respond to contexts and parameters

### Table 43:
**SAP BusinessObjects RESTful Web Service SDK for Web Intelligence and the BI Semantic Layer 4.1 SP2 (November 2013)**

The name of the guide has been modified. The guide has been reorganized.

New features for SAP BusinessObjects Web Intelligence RESTful Web Service SDK:

- Use the new `unit` parameter to describe sizes when exporting a document, report, or report element
- Create the Section and Cell report elements
- Update the Section, Cell, and PageZone report elements
- Delete a report element
- Create, update, and delete a report filter
- Get report filter details
- Create, update, and delete report element data filters
- Get the filter details on a report element

First release of the document for SAP BusinessObjects BI Semantic Layer RESTful Web Service SDK. Support of the following new features:

- Get the list of universes, the universe metadata, the business views, the query capabilities, and the link groups of a universe
- Create a query, get the details of a query, get the list of queries, delete a query
- Get the query results as OData flow metadata and content
### Table 44:
SAP BusinessObjects Web Intelligence RESTful Web Service SDK 4.1 SP1 (August 2013)

**New features:**
- Export a report in paginated mode as a zipped HTML file
- Export a page of a report as a zipped HTML file
- Get the list of supported chart types (configuration/visualization)
- Get the list of default color palettes (configuration/palettes)

### Table 45:
SAP BusinessObjects Web Intelligence RESTful Web Service SDK 4.1 (June 2013)

**SDK functionality:**
- Browse for documents and manage document life cycle
  - List all documents in the CMS, get document details, open/save/close a document, access and export a document, a report and a report element, get a report page in listing or paginated mode
  - Refresh a document
  - Support of text, numeric, optional/mandatory prompts, prompting and context resolution, support of date and date-time, support of List of Values in prompts, support of prompts on SAP variables, support of nested prompts
- Manage data providers
  - Get the list of data providers of a document, get data provider details for a document, rename/sort/delete a data provider, export data as CSV or XML, purge a document, get the query statement generated by query on universes, change data provider source workflow
- Manage data sources
  - Get the list of universes stored in the CMS, get universe basic information, get universe detailed information, browse the list of BEx connections and BW BEx queries, get details of a BEx query, get data source query capabilities
- Author queries
  - Get/update the query specification associated with a data provider
- Author documents
  - Create a blank document, add a report to a document and set its report specification, delete a report from a document, get/create/delete/update track data changes, alerters, styles, skins, variable expressions, manage drill and create drill snapshots, manage document state through storage token, get custom format number from the document, get document default configuration from server, get font mappings, CSS styles, format numbers, default skins, and supported operators
- Support scheduling properties
  - Support of main recurrences (Once, Hourly, Daily, Weekly, Monthly), support of server group, get details of a schedule, add a schedule
3 Introduction to the RESTful Web Service SDK User Guide

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

What this Guide is About

The SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer provides conceptual and reference information on the BI Semantic Layer RESTful Web Service SDK and the Web Intelligence RESTful Web Service SDK:

- Purpose and concepts
- How to use the samples provided with the SDKs
- HTTP methods and URLs of the REST APIs
- Request and response examples of the REST APIs

Administration Tasks

See the BI Platform RESTful Web Service Developer Guide to learn about:

- Setting up the development environment
- Installing the RESTful web services for a custom installation
- Configuring the RESTful web services

See the Business Intelligence Platform Administrator Guide to learn about:

- Starting the WACS server
- Activating trace log
- Displaying the trace

See the guides at http://help.sap.com/Analytics SAP BusinessObject Business Intelligence SAP BusinessObjects Business Intelligence platform 4.2

Audience [page 33]
Conventions in This Guide [page 33]
Key Tasks [page 33]
Use Cases [page 34]
3.1 Audience

The SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer is intended for developers responsible for writing programs that access and consume the BI platform RESTful web services. These programs can be embedded within Business Intelligence solutions on the BI platform.

3.2 Conventions in This Guide

For consistency and readability across this guide:

- Request and response body schemas are described in XML format without closing tags. On the contrary, in the REST API call examples, requests and responses are described thoroughly in either XML or JSON format.
- The REST API references do not specify the default base URL, which is described in the Default Base URLs section.

3.3 Key Tasks

This guide provides you with key information for using the RESTful Web Service SDKs. For each of the following tasks, see the appropriate section:

Table 46:

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To retrieve the universe description</td>
<td>Getting the Details of a Universe (Semantic Layer) [page 143]</td>
</tr>
<tr>
<td>To design a query specification</td>
<td>Queries [page 107]</td>
</tr>
<tr>
<td>To query a universe</td>
<td>Creating a Query [page 155]</td>
</tr>
<tr>
<td>To get the query results</td>
<td>Managing Query Results [page 179]</td>
</tr>
<tr>
<td>To create an empty Web Intelligence document</td>
<td>Creating a Document [page 193]</td>
</tr>
<tr>
<td>To add a data source to a document</td>
<td>Adding a Data Provider [page 410]</td>
</tr>
<tr>
<td>To add a query specification to a document</td>
<td>Updating the Query Specification [page 443]</td>
</tr>
<tr>
<td>To get document data</td>
<td>Refreshing a Document [page 468]</td>
</tr>
<tr>
<td>To create an empty report</td>
<td>Creating a Report [page 289]</td>
</tr>
<tr>
<td>To create elements in a report</td>
<td>Creating a Report Element [page 346]</td>
</tr>
<tr>
<td>To format a report</td>
<td>Updating a Report Element [page 370]</td>
</tr>
<tr>
<td>To refresh a document</td>
<td>Refreshing Documents [page 463]</td>
</tr>
<tr>
<td>To schedule a document</td>
<td>Adding a Schedule [page 495]</td>
</tr>
</tbody>
</table>
3.4 Use Cases

The SAP BusinessObjects RESTful Web Service SDKs allow you to access the resources of the SAP BusinessObjects BI platform (universes, Web Intelligence documents and reports) stored in the CMS repository using the HTTP protocol. You implement CRUD (Create, Read, Update, Delete) operations so that the provided web services send requests over HTTP and receive responses in the XML or JSON format.

For example, you can use these SDKs to retrieve data from a data provider and expose it through reports built with report elements. You can perform report scheduling and drill through a report to explore and analyze business data. You can also browse universe metadata, submit and execute queries.

The SDKs allow you to address some business requirements that can include:

- Exposing universe metadata within a custom application
- Building a custom query panel and embedding it anywhere
- Creating queries on the fly during analysis
- Running queries directly into custom scripts, Java applications, and so on
- Running queries to feed a client tool with business data
- Embedding documents and reports in non-SAP client tools or web applications
- Working with documents and reports within custom applications
4 Using the RESTful Web Service SDKs

To develop applications that use the RESTful Web Service SDKs, you must be able to log on to a BI platform deployment that has RESTful web services installed.

See the BI platform RESTful Web Service Developer Guide to learn about:

- Retrieving the base URL, either from the CMC or with a program
- Making RESTful web service requests
- Logging on to the BI platform RESTful Web Service SDK
- Logging off from the BI platform RESTful Web Service SDK

The RESTful Web Service SDKs also rely on the BI platform RESTful Web Service API for session management and repository access.

REST Methodology

The SDKs comply with the REST methodology. You can access them using any programming language that supports making HTTP requests. You can send HTTP requests from any operating system, either Microsoft Windows or UNIX, or even from a mobile platform. You can also make HTTP requests without writing code by using tools that make HTTP requests.

Format

Both XML and JavaScript Object Notation (JSON) request and response formats are supported. For more information on the JSON format, see [www.json.org](http://www.json.org) and [RFC4627](http://www.json.org). For more information on the XML format, see [www.w3.org/XML](http://www.w3.org/XML).

The SDKs require the following JSON syntax in request bodies:

- The at sign (@) for attributes
- The dollar symbol ($) for element values
For example:

Table 47:

<table>
<thead>
<tr>
<th>XML</th>
<th>JSON</th>
</tr>
</thead>
</table>
| <schedule>
  <id>9439</id>
  <name>now-schedule</name>
  <format type="webi"/>
  <status id="1">Completed</status>
  <destination>
    <ftp>
      <host>vs0202</host>
      <port>21</port>
      <username>admin</username>
      <account/>
      <directory>./</directory>
    </ftp>
  </destination>
</schedule> | 
| {  
  "schedule": {
    "id": "9439",
    "name": "now-schedule",
    "format": {
      "@type": "webi"
    },
    "status": {
      "@id": "1",
      "$": "Completed"
    },
    "destination": {
      "ftp": {
        "host": "vs0202",
        "port": "21",
        "username": "admin",
        "directory": "/"
      }
    }
  }
} |

Before You Begin

Before using the RESTful Web Service SDKs, make sure the WebApplicationContainerServer (WACS) server is running. This server hosts the RESTful web services. You can also decide to trace the server activity.

- Default Base URLs [page 36]
- Getting Web Intelligence RESTful Web Service SDK Information [page 37]
- Supporting Multiple Languages [page 38]
- Dispensable Information in a Request Body [page 39]
- cURL Examples [page 40]
- HTTP Request Status [page 42]
- Successful and Error Messages [page 43]

4.1  Default Base URLs

To use the RESTful web services for Web Intelligence and the BI Semantic Layer, you must know the protocol, server name, port number and path of the service that listens to the HTTP requests. You configure the default base URL in the CMC from Applications ➔ REST Web Service ➔ Properties ➔ Access URL. See chapter 12 of the Business Intelligence Platform Administrator Guide for more information.

Basic installations of the BI platform that are installed on a single server use the default base URLs:
Table 48:

<table>
<thead>
<tr>
<th>Web Service SDK</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI Semantic Layer</td>
<td>http://&lt;server_name&gt;:6405/biprws/sl/v1</td>
</tr>
<tr>
<td>Web Intelligence</td>
<td>http://&lt;server_name&gt;:6405/biprws/raylight/v1</td>
</tr>
</tbody>
</table>

**Note**

6405 is the default HTTP port number used by the RESTful web services.

**Conventions in this Guide**

The following conventions apply to the REST APIs described in this guide:

- `<base_webi_REST_URL>` stands for http://<server_name>:6405/biprws/raylight/v1
- `<base_sl_REST_URL>` stands for http://<server_name>:6405/biprws/sl/v1

**Note**

The API references do not show the default base URLs to make the reading easier.

### 4.2 Getting Web Intelligence RESTful Web Service SDK Information

**Usage**

Returns the following information about your running version of SAP BusinessObjects Web Intelligence RESTful Web Service SDK:

- Product name
- Version number
- Vendor name
- Major release, minor release, Support Package, patch, and build numbers
- Timestamp
- Copyright

**Note**

- The result is computed at build time, not running time.
- You can access this URL without authentication in any Web browser.
4.3 Supporting Multiple Languages

The RESTful Web Service SDKs allow you to work in multilingual environments. You can request Web Intelligence documents and system messages to be returned in your preferred language. Include the following declarations in the request header:
Table 49:

<table>
<thead>
<tr>
<th>Header</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept-Language</td>
<td>The preferred language used to retrieve system and error messages. This corresponds to the Product Locale (PL) of the BI platform. To get system messages returned in French, set Accept-Language to fr-FR.</td>
</tr>
<tr>
<td>X-SAP-PVL</td>
<td>The preferred language used to retrieve BI resource content such as Web Intelligence documents. This corresponds to the Preferred Viewing Language (PVL). To request all documents available in German, set X-SAP-PVL to de-DE.</td>
</tr>
</tbody>
</table>

The RESTful web service opens one instance of the Web Intelligence document in the client machine memory for each Preferred Viewing Locale requested by the client.

**Related Information**

cURL Examples [page 40]

### 4.4 Dispensable Information in a Request Body

You can send a POST or PUT HTTP request with a body to create or update a Web Intelligence resource. If you provide some piece of information, which is not needed to perform the requested action, the method simply ignores it and returns a successful message.

**Example**

You want to update some `<dataLabels>` property of a Waterfall chart by sending the following request:

```
PUT <base_webi_REST_URL>/documents/8084/reports/12/elements/27
```

In SAP BusinessObjects Web Intelligence, a Waterfall chart does not contain any legend. However, the request body you are sending contains a legend as follows:

```xml
<element type="Visualization">
  <parentId>2</parentId>
  <content>
    <chart type="Waterfall">
      <layout showDimensionsWithEmptyMeasureValues="true"
               showDimensionsWithMeasuresEqualToZero="true"
               showDimensionsWithSumOfMeasuresEqualToZero="true"
               showMeasuresWithEmptyDimensionValues="false"
               showTotal="true" parentAsTotal="false"
               duplicateRowAggregation="true" horizontal="false"/>
      <title visible="true">...
      </title>
      <legend visible="true">
        <style>
          <border thickness="None">
            <color rgb="#000000" alpha="0"/>
          </color>
        </border>
      </legend>
    </chart>
  </content>
</element>
```
The REST Web Service SDK ignores the `<legend>` part and updates the chart as requested.

The response is the following message:

```
<success>
  <message>The resource of type 'Report element' with identifier '27' has been successfully updated.</message>
  <id>27</id>
</success>
```

### 4.5 cURL Examples

The `cURL` tool has excellent support for HTTP requests. The following examples illustrate the `cURL` commands to run either on Microsoft Windows or on UNIX to send HTTP requests. See the `cURL` documentation for more information.

#### Example

**GET Universe Details (Microsoft Windows)**

```
curl -G -i -H "accept:application/xml" -H X-SAP-LogonToken:""%tokenValue%"
"<base_webi_REST_URL>/universes/5808"
```
Example

GET Universe Details (UNIX)

Note

You use the logtok variable set at login.

curl -H "accept:application/xml" -H 'X-SAP-LogonToken:"$logtok"'"<base_sl_REST_URL>/universes/5808"

Example

GET Document Details in French Locale

This example uses the X-SAP-PVL variable.


Example

POST Document

You create an empty document whose name and folder are described in the newdocument.xml file passed as an argument.


Example

PUT Document Alerter

You update the definition of an alerter for a specified document. The new definition is contained in the alerter4.xml file passed as an argument.


Example

DELETE

You delete a document schedule.

Related Information

http://curl.haxx.se/

4.6 HTTP Request Status

The response of an HTTP request contains a `Status Code` attribute that gives information on the success or error of the request call.

See the table below to learn about the codes that provide status for HTTP requests.

<table>
<thead>
<tr>
<th>Code</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
<td>Successful request.</td>
</tr>
<tr>
<td>400</td>
<td>Bad request</td>
<td>The requested resource exists, but the request contains errors.</td>
</tr>
<tr>
<td>401</td>
<td>Failed to logon or invalid session</td>
<td>Logon failed. Check that the user name, password, and server name are correct. The current session may have expired. Log on to obtain a new session.</td>
</tr>
<tr>
<td>403</td>
<td>Access denied</td>
<td>You do not have permission to operate on the requested resource.</td>
</tr>
<tr>
<td>404</td>
<td>Service is not available</td>
<td>The requested service is not provided by the RESTful Web Service SDK.</td>
</tr>
<tr>
<td>405</td>
<td>Invalid request method</td>
<td>A request was made using a method that was not supported by the resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, using a <code>PUT</code> request on a read-only resource.</td>
</tr>
<tr>
<td>406</td>
<td>Not acceptable</td>
<td>The requested resource cannot generate the content type specified by the <code>Accept</code> attribute of the request header.</td>
</tr>
<tr>
<td>408</td>
<td>BI platform server timeout</td>
<td>The server timed out waiting for the request.</td>
</tr>
<tr>
<td>415</td>
<td>Unsupported media type</td>
<td>The request contains a media type that the server or resource does not support.</td>
</tr>
<tr>
<td>500</td>
<td>RESTful web service internal error</td>
<td>An unclassified error occurred. See the response body for more information.</td>
</tr>
</tbody>
</table>
### Example

**Content of a Response Header**

This example shows a successful response code.

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Date: Fri, 15 Jun 2012 10:14:15 GMT
Content-Type: application/xml
Content-Length: 204
```

### 4.7 Successful and Error Messages

#### Successful Messages

If the request is successful, HTTP calls to REST APIs return a message in the XML or JSON format, as follows:

```
{"success":
   {"message": "a successful message",
    "id": "an identifier"
    }
}
```

Where `id` refers to the identifier of the object concerned, such as a query, document, variable, or style. In some rare cases, `id` is not present.

---

**Example**

**JSON Successful Message**

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Date: Tue, 13 Jan 2015 10:14:15 GMT
Content-Type: application/json
Content-Length: 204
{"success":
   {"message": "The resource of type "Document" with identifier "16706" has been successfully updated.",
    "id": "16706"
    }
}
```
Error Messages

If the request was unsuccessful, the call returns a message in the XML or JSON format, as follows:

```xml
<error>
    <error_code>An error code</error_code>
    <message>An error message</message>
</error>
```

The following table shows the possible Web Intelligence RESTful Web Service SDK error codes. Each of these codes addresses a category of error messages.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSR 00001</td>
<td>The user does not provide session token. The session is not found.</td>
</tr>
<tr>
<td>WSR 00002</td>
<td>The session token is invalid.</td>
</tr>
<tr>
<td>WSR 00100</td>
<td>A rule is not respected.</td>
</tr>
<tr>
<td>WSR 00101</td>
<td>An argument is not correct.</td>
</tr>
<tr>
<td>WSR 00102</td>
<td>The request body is malformed.</td>
</tr>
<tr>
<td>WSR 00400</td>
<td>The user tries to get a resource that does not exist.</td>
</tr>
<tr>
<td>WSR 00401</td>
<td>The user tries to create a resource that already exist.</td>
</tr>
<tr>
<td>WSR 00402</td>
<td>The user fails to access a resource.</td>
</tr>
<tr>
<td>WSR 00501</td>
<td>The user tries to perform an action that is not supported.</td>
</tr>
<tr>
<td>WSR 00999</td>
<td>This is an internal error.</td>
</tr>
</tbody>
</table>

The BI Semantic Layer RESTful Web Service SDK error codes are more specific. Each of these codes corresponds to a particular error message and to a specific error. See the Error Messages Explained guide for a detailed description of the error messages.

You can display the full error stack traces for debugging purpose by checking the Show Error Stack property for your WACS server in the CMC. See the Business Intelligence Platform Administrator Guide for the complete procedure.

Note

The API reference chapters do not describe the call responses if an error occurs.

Example

400 Error Code

```xml
Content-Type: application/xml
HTTP Response Code: 404
<?xml version="1.0" encoding="UTF-8"?>
<error>
    <error_code>WSR 00400</error_code>
    <message>The expression "[DUMMY]" cannot be found in the document dictionary.</message>
</error>
```
Example

401 Error Code

Content-Type: application/xml
HTTP Response Code: 400
<?xml version="1.0" encoding="UTF-8"?>
<error>
    <error_code>WSR 00401</error_code>
    <message>The resource of type "Report" with identifier "154769" already has an inner resource of type "Data filter".</message>
</error>
5 Using the RESTful Web Service Samples

Code samples are provided to demonstrate the usage of the Web Intelligence and BI Semantic Layer RESTful Web Service SDKs.

The following samples are Java code snippets that allow you to understand how a Java application can be implemented to run HTTP calls to the REST APIs. You can use these samples as a basis to facilitate and accelerate your own Java programming.

Samples for the BI Semantic Layer REST APIs are supplied in the archive C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\SL SDK\SDK Samples\SLRESTWebService.zip.

Samples for the Web Intelligence REST APIs are supplied in the archive C:\Program Files\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\Samples\webi\RaylightRESTWS_Samples.zip.

5.1 About the BI Semantic Layer REST API Samples

The SDK samples described in the following table work with the Warehouse.unx sample universe. You can find this universe on the SAP Community Network at http://scn.sap.com/docs/DOC-22145. You must download the universe from this page and store it in a CMS repository to be able to run the samples.

Table 5.1:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetUniverseMetadataTest</td>
<td>Returns the details of the Warehouse.unx universe using GET /universes/ &lt;universeID&gt;</td>
</tr>
<tr>
<td>GetUniversesTest</td>
<td>Returns the list of the universes in the CMS repository using GET /universes? offset&amp;limit</td>
</tr>
<tr>
<td>GetUniverseViews</td>
<td>Returns the views of the Warehouse.unx universe using GET /universes/ &lt;universeID&gt;/businessviews</td>
</tr>
<tr>
<td>QueryExecutionTest</td>
<td>Creates a query on top of Warehouse.unx:</td>
</tr>
<tr>
<td></td>
<td>● Creates a query using POST /queries</td>
</tr>
<tr>
<td></td>
<td>● Runs the query using GET /queries/&lt;queryID&gt;/data.svc</td>
</tr>
<tr>
<td></td>
<td>● Returns the query results using GET /queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;</td>
</tr>
<tr>
<td></td>
<td>● Deletes the query using DELETE /queries/&lt;queryID&gt;</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>QueryParametersTest</td>
<td>Creates a query on top of Warehouse.unx and answers parameters this query requires:</td>
</tr>
<tr>
<td></td>
<td>● Creates a query with parameters using POST /queries/&lt;queryID&gt;/parameters</td>
</tr>
<tr>
<td></td>
<td>● Returns the query parameters using GET /queries/&lt;queryID&gt;/parameters</td>
</tr>
<tr>
<td></td>
<td>● Runs the query using GET /queries/&lt;queryID&gt;/data.svc</td>
</tr>
<tr>
<td></td>
<td>● Returns the query results using GET /queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;</td>
</tr>
<tr>
<td></td>
<td>● Deletes the query using DELETE /queries/&lt;queryID&gt;</td>
</tr>
</tbody>
</table>

resources and util Packages

The com.sap.sl.sdk.consumption.samples.resources package contains XML descriptions used in sample requests, such as the query specification. The com.sap.sl.sdk.consumption.samples.util package provides classes that help to perform utility actions in the samples (for example, login, fill in CMS parameters, send a request, and so on).

To Deploy the Samples with Eclipse [page 47]

5.1.1 To Deploy the Samples with Eclipse

You can run Java samples in your Eclipse project.

1. Launch Eclipse by double-clicking the eclipse.exe file and select your workspace. Use Eclipse 3.6 or higher.
2. Create a project.
3. Select File ➤ Import. The Import dialog box appears.
4. Expand General, select Existing Projects into Workspace, and click Next.
5. Select the Select archive file option and click Browse.
6. Browse to the sample directory and select SLRESTWebService.zip.
8. Under com.sap.sl.sdk.consumption.samples.util, open the BaseTest.java file for editing.
9. Fill in the following parameters with the values corresponding to your installation:
   ○ CMS_SERVER_URL: the URL exposed by the server hosting the RESTful web services
   ○ CMS_USER: user name used in your samples to connect to the CMS repository
   ○ CMS_PASSWORD: password used in your samples to connect to the CMS repository
10. Under com.sap.sl.sdk.consumption.samples, right-click a sample class and select Run As ➤ 2 JUnit Test. The sample runs, allowing you to find out the features of the BI Semantic Layer RESTful Web Service SDK.
5.2 About the Web Intelligence REST API Samples

The SDK samples described in the following table work with the following set of universes and documents:

- Warehouse.unx, eFashion.unx, and eFashion.unv universes
- [Raylight Sample Template] Change Source, [Raylight Sample Template] Empty, [Raylight Sample Template] Refresh, and [Raylight Sample Template] Schedule documents

You must deploy the LCMBIAR file included in the ZIP file in your CMS repository to be able to run the samples. The archive file is lcmbiar/RaylightSamples.lcmbiar. Once run, Web Intelligence documents are generated in the CMS repository and can be found under Documents/Folders/Web Intelligence Samples.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateAndSaveSample</td>
<td>Creates and saves an empty Web Intelligence document:</td>
</tr>
<tr>
<td></td>
<td>• Creates a new empty document</td>
</tr>
<tr>
<td></td>
<td>• Adds a data provider</td>
</tr>
<tr>
<td></td>
<td>• Adds a query specification</td>
</tr>
<tr>
<td></td>
<td>• Adds a report</td>
</tr>
<tr>
<td></td>
<td>• Saves the document</td>
</tr>
<tr>
<td></td>
<td>The data provider and query specification are attached to the document and</td>
</tr>
<tr>
<td></td>
<td>available for future use.</td>
</tr>
<tr>
<td></td>
<td>To display traces in the console, you can uncomment the line containing</td>
</tr>
<tr>
<td></td>
<td>deleteDocument(newDocID).</td>
</tr>
<tr>
<td>ChartSample</td>
<td>Adds four charts to an existing document:</td>
</tr>
<tr>
<td></td>
<td>• Refreshes the document (data provider/query specification)</td>
</tr>
<tr>
<td></td>
<td>• Adds a cell used to display the title</td>
</tr>
<tr>
<td></td>
<td>• Adds a column chart to the report</td>
</tr>
<tr>
<td></td>
<td>• Adds a donut chart to the report</td>
</tr>
<tr>
<td></td>
<td>• Adds a bar chart with its own data filter</td>
</tr>
<tr>
<td></td>
<td>• Adds a surface chart with its own data filter</td>
</tr>
<tr>
<td>TableSample</td>
<td>Adds a table with seven columns to a document:</td>
</tr>
<tr>
<td></td>
<td>• Refreshes the document with prompt values</td>
</tr>
<tr>
<td></td>
<td>• Adds a cell used to display the title</td>
</tr>
<tr>
<td></td>
<td>• Adds a vertical table to the report</td>
</tr>
<tr>
<td></td>
<td>• Resizes the product column width as automatic</td>
</tr>
<tr>
<td></td>
<td>• Adds alerters and a variable</td>
</tr>
<tr>
<td></td>
<td>• Adds an empty column to the table and fills it with the variable</td>
</tr>
<tr>
<td>RefreshSample</td>
<td>Refreshes a document through different ways:</td>
</tr>
<tr>
<td></td>
<td>• Refreshes without changes</td>
</tr>
<tr>
<td></td>
<td>• Refreshes with changes: custom context &amp; prompt</td>
</tr>
<tr>
<td></td>
<td>• Cancels a refresh</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ScheduleSample</td>
<td>Schedules a document through different ways:</td>
</tr>
<tr>
<td></td>
<td>● Purges the document from its data and saves it</td>
</tr>
<tr>
<td></td>
<td>● Schedules the document to the BI launch pad inbox</td>
</tr>
<tr>
<td></td>
<td>● Schedules the document to an email address</td>
</tr>
<tr>
<td></td>
<td>● Schedules the document to an FTP server</td>
</tr>
<tr>
<td></td>
<td>● Schedules the document to a file system</td>
</tr>
<tr>
<td></td>
<td>Only the schedule to the BI launch pad does not require CMS server configuration.</td>
</tr>
<tr>
<td>FreeHandSQLSample</td>
<td>Uses the free-hand SQL data provider:</td>
</tr>
<tr>
<td></td>
<td>● Creates an empty document</td>
</tr>
<tr>
<td></td>
<td>● Renames the report</td>
</tr>
<tr>
<td></td>
<td>● Checks if the connection is relational</td>
</tr>
<tr>
<td></td>
<td>● Adds a free-hand SQL data provider using a custom query</td>
</tr>
<tr>
<td></td>
<td>● Adds a cell title</td>
</tr>
<tr>
<td></td>
<td>● Refreshes the data without any changes</td>
</tr>
<tr>
<td></td>
<td>● Adds a chart using data from the free-hand SQL data provider</td>
</tr>
<tr>
<td></td>
<td>● Adds an input control</td>
</tr>
<tr>
<td></td>
<td>You must know the database schema before using a custom query.</td>
</tr>
<tr>
<td>ChangeSourceSample</td>
<td>Changes the data source of an existing document:</td>
</tr>
<tr>
<td></td>
<td>● Gets the suggested mapping to change source</td>
</tr>
<tr>
<td></td>
<td>● Applies the suggested mapping without any changes</td>
</tr>
<tr>
<td></td>
<td>To edit the mapping and set up your own data objects, you can uncomment the line containing editMapping().</td>
</tr>
</tbody>
</table>

resources and util Packages

The `com.sap.webi.raylight.samples.resources` package contains XML descriptions used in sample requests, such as the query specification. The `com.sap.webi.raylight.utils` package provides classes that help to perform utility actions in the samples (for example, login, fill in CMS parameters, send a request, and so on). The `com.sap.webi.raylight.param` contains the parameters required for sample execution.

To Deploy the LCMBIAR File [page 49]
To Deploy the Samples with Eclipse [page 50]

5.2.1 To Deploy the LCMBIAR File

You must deploy the LCMBIAR archive file before you deploy and run the samples.

1. Login to the CMC.
2. Select **Promotion Management** under the **Manage** menu.
3. Select [Import] > [Import file].
4. Check **File System** and click **Choose File** to upload the **RaylightSamples.lcmbrar** file from the ZIP.
5. Click **OK**.
   The **New Job** dialog box appears. The new job name **RSamples** is prefilled.
6. Select **From File** for the **Source** and **Login to a New CMS** for Destination.
   The **Login To System** dialog box appears.
7. Type in the credentials of your CMS repository and click **OK**.
8. Click **Create** in the **New Job** dialog box.
   The LCMBIAR file content appears.
9. Click **Manage Dependencies** to check the dependencies between universes and documents. If OK, click **Apply & Close**.
10. Click **Promote** to add the job to the Promotion Job folder.
    The **RSamples** job appears in the Promotion Job folder.
11. Select this job and click **Promote**.
    All the objects to be promoted are listed.
12. Click **Schedule** to finish the deployment.

### 5.2.2 To Deploy the Samples with Eclipse

You can run Java samples in your Eclipse project.

1. Launch Eclipse by double-clicking the **eclipse.exe** file and select your workspace.
   Use Eclipse 3.6 or higher.
2. Create a project.
3. Select [File] > [Import].
   The Import dialog box appears.
4. Expand **General**, select **Existing Projects into Workspace**, and click **Next**.
5. Select the **Select archive file** option and click **Browse**.
6. Browse to the sample directory and select **RaylightRESTWS_Samples.zip**.
7. In the **Projects** area, select **com.sap.webi.raylight.samples** and click **Finish**.
   The **com.sap.webi.raylight.samples** project folder appears in the **Package Explorer** view.
8. Under **com.sap.sl.webi.raylight.param**, open the **Config.java** file for editing.
9. Fill in the following parameters with the values corresponding to your installation:
   - **CMS_SERVER_URL**: the URL exposed by the server hosting the RESTful web services
   - **CMS_USER**: user name used in your samples to connect to the CMS repository
   - **CMS_PASSWORD**: password used in your samples to connect to the CMS repository
10. Under **com.sap.webi.raylight.samples**, right-click a sample class and select [Run As] > [JUnit Test].
   The sample runs, allowing you to find out the generated Web Intelligence documents in your CMS repository under **Documents/Folders/Web Intelligence Samples**.
6 Concepts and Schemas of the RESTful Web Service SDKs

The following sections present thorough descriptions of XML request and response schemas for the universes, queries, parameters, charts, schedules, and data source mappings.

The following BI Semantic Layer concepts are supported in this release:

- Dimensions, attributes, and measures
- Predefined filters
- Simple custom query filters and subquery filters
- Combined queries
- List of values
- Parameters

Change Source [page 51]
Charts [page 58]
Date and Time Formats [page 92]
Object Full Paths [page 93]
Parameters [page 95]
Queries [page 107]
Report Structure [page 131]
Schedules [page 132]
Universes [page 136]
User Rights [page 140]

6.1 Change Source

Changing data sources of a query in a Web Intelligence document means replacing data objects from a data source with data objects from another data source. This replacement is computed according to a set of rules called strategies, which define how target data objects should match the data objects to be replaced.

Use Cases

The "change source" functionality addresses two main use cases:

- Linking a document to a universe that has been converted from UNV to UNX
• Linking a document uploaded to the CMS repository to a data source actually stored in the CMS repository

**Supported Data Sources**

The following table describes the source and target data sources supported by the "change source" functionality.

Table 53:

<table>
<thead>
<tr>
<th>Source UNV</th>
<th>Target UNV</th>
<th>Target UNX</th>
<th>Target BEx Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>Source UNX</td>
<td>Not supported</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>Source BEx Query</td>
<td>Not supported</td>
<td>Not supported</td>
<td>OK</td>
</tr>
</tbody>
</table>

**Note**

The "change source" functionality does not support text files and Microsoft Excel spreadsheets as data sources.

"Change Source" Workflow

**Before 4.1 SP4**

1. **GET:** getting the suggested data object mappings for the selected data providers based on a given target data source. The default strategy applies.
2. **POST:** updating the document with the selected data objects of the target data source mapped to the data objects to be replaced. The object mapping used can be the one suggested by the first call or can have been edited.

**Since 4.1 SP4 Patch 4 and 4.1 SP5 Patch 1**

It is possible to apply the suggested mapping (POST call) without getting it (GET call). This means the mapping is considered to be valid.

In the POST call:

- If the request body is empty, then the object mapping used is the one found by the default strategies.
- If the request body contains an incomplete object mapping, then the default mapping is used to replace the missing objects.

**Since 4.1 SP6**

1. Getting the suggested data object mappings using either the default strategies (GET) or the given strategies (PUT).
   In the GET call, the default strategies are the ones used in previous releases. In the PUT call, you specify the order of the strategies, which apply one after the other for each data object only when the mapping status is "not found" and until a match is found.
2. **POST:** updating the document with the selected data objects of the target data source mapped to the data objects to be replaced.
The object mapping used in the POST call can be the one suggested either by the GET call (default mapping) or by the PUT call (mapping found by the given strategies). A custom mapping can also be specified in the request body.

In the POST call:
- If the request body is empty, then the object mapping used is the default mapping.
- If the request body contains an incomplete object mapping, then the default mapping is used to replace the missing objects.
- If the request body contains an empty or incomplete object mapping but with strategies, then the mapping used is overloaded by the one found with the strategies.

**Note**

For custom mappings, you can change only the target ID of a given mapping, but must keep all the source IDs. If the target ID is null, then the data source object will be removed.

Mappings Request and Response Body Schemas [page 53]
Strategies [page 55]
Compatibility Rules [page 56]

### 6.1.1 Mappings Request and Response Body Schemas

**Request Body (XML)**

(POST method)

```
<mappings>
  <policy qualificationTolerance="Low|Normal|High" dataTypeTolerance="Low|Normal|High">  
    <strategies mappingSourceIds="string">
      <strategy name="SameId|SameName|SameTechnicalName|SamePath|CloseName|Selection|Removal" targetId="string" />
    </content>
    <mapping>
      <source>
        <id>
      </target>
      <id>
      <parameters>
```

Table 54:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<policy>` | **Attributes:**  
| qualificationTolerance (Low|Normal|High)  
| dataTypeTolerance (Low|Normal|High)  
| They specify the tolerance to be considered when matching the source and target datasource objects. They define the compatibility rules. |
<strategies>
The list of strategies to apply. This element is optional.  
Attribute: `mappingSourceIds` to apply the strategies to the specified source data objects. The IDs are separated by a comma. If no attribute is specified, the strategy will apply to all source data objects (default strategy).

<strategy>
A strategy selected to compute the possible object mappings.  
Attributes:  
- `name`: strategy name  
- `targetId` required by the Selection strategy to specify the ID of the target data object to be selected

<content>
The custom object mapping. This element is optional.

<parameters>
The document prompt answers if necessary. This element is optional.

The following table describes the `qualificationTolerance` attribute values:

<table>
<thead>
<tr>
<th>Attribute value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>The source and target data objects must have the same qualification.</td>
</tr>
<tr>
<td>Normal</td>
<td>The source and target data objects can have close qualifications.</td>
</tr>
<tr>
<td>High</td>
<td>The source and target data objects can have different qualifications.</td>
</tr>
</tbody>
</table>

The following table describes the `dataTypeTolerance` attribute values:

<table>
<thead>
<tr>
<th>Attribute value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>The source and target data objects must have the same data type.</td>
</tr>
<tr>
<td>Normal</td>
<td>The source and target data objects can have close data types.</td>
</tr>
<tr>
<td>High</td>
<td>The source and target data objects can have different data types.</td>
</tr>
</tbody>
</table>

Example

`qualificationTolerance="Low" dataTypeTolerance="High"` means:  
- The qualifications of the source and target data objects must be exactly the same.  
- The data types of the source and target data objects can be different.
Response Body (XML)

(GET and POST methods)

```xml
<mappings>
  <content>
    <mapping status="Ok|Ambiguous|Not found">
      <source>
        <id></id>
      </source>
      <target>
        <id></id>
      </target>
    </mapping>
  </content>
</mappings>
```

Table 57:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| <mapping> | N/A           | Attribute: status  
- Ok if the mapping fully matches  
- Ambiguous in case of uncertain mapping  
- Not found if there is no possible match |
| source <id> | string       | The identifier of the source data object |
| target <id> | string       | The identifier of the target data object |

6.1.2 Strategies

A strategy is a rule that defines how an object of the current data source can match an object of the target data source. A strategy allows you to find the data object mapping.

The following table describes the strategies that can apply:

Table 58:

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“SameID”</td>
<td>Searches a valid object that has the same ID</td>
</tr>
<tr>
<td>“SameTechnicalName”</td>
<td>Searches a valid object that has the same technical name</td>
</tr>
<tr>
<td></td>
<td>Note: This strategy may also apply to UNV universes, but does not return any result, because UNV universes do not support technical names.</td>
</tr>
<tr>
<td>“SamePath”</td>
<td>Searches a valid object that has the same path. The path includes the object name and its parent folders.</td>
</tr>
</tbody>
</table>
### Strategy

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;CloseName&quot;</td>
<td>Searches a valid object that has the closest name based on the Levenshtein distance, with the distance lower than a fixed threshold. If several objects have the same distance, then the selected object is the one whose ID comes first in alphabetical order.</td>
</tr>
<tr>
<td>&quot;Removal&quot;</td>
<td>Removes the object from the data provider if the previous strategies did not find any target object</td>
</tr>
<tr>
<td>&quot;Selection&quot;</td>
<td>Selects a specific target object. Requires the attribute &quot;targetId&quot; attribute containing the ID of the target data object to be selected</td>
</tr>
<tr>
<td>&quot;SameName&quot;</td>
<td>Maps a valid object whose Levenshtein distance is null, that is, finds an object with the same name</td>
</tr>
</tbody>
</table>

The following default strategies apply in the following order to find what is called the default mapping:

1. "SameID"
2. "SameTechnicalName"
3. "SamePath"
4. "CloseName"

### Related Information

- Getting the Possible Object Mappings Using the Default Strategies [page 423]
- Getting the Possible Object Mappings Using Selected Strategies [page 425]

### 6.1.3 Compatibility Rules

When running the "change source", the following compatibility rules apply to avoid an inconsistent mapping:

- Compatibility of the source and target object types (dimension, measure, attribute, hierarchy, and so on)
- Compatibility of the source and target object data type (member, string, numeric, date, and so on)
- Compatibility of the source and target object usage, if the object can be used as a result object, a filter object, or a sort object

These compatibility rules are defined through the `qualificationTolerance` and `dataTypeTolerance` attributes. They specify the tolerance to be considered when matching the source and target objects.

The following table shows the possible object matches according to the `qualificationTolerance` attribute value applied to the object type.

| Table 59: |
|-----------|-------------------|----------------|-----------------|------------------|-----------------|-----------------|----------------|
| Source Dimension | Target Dimension | Target Hierarchy | Target Level | Target Dimension Attribute | Target Measure Attribute | Target Measure | Target Others |
| Low         | Medium            | Medium          | Medium        | Medium           | High            | High            |               |
The following table shows the possible object matches according to the `dataTypeTolerance` attribute value applied to the data type.

Table 60:

<table>
<thead>
<tr>
<th>Source N/A</th>
<th>Target N/A</th>
<th>Target Member</th>
<th>Target Numeric</th>
<th>Target String</th>
<th>Target Time/Date</th>
<th>Target Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source N/A</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Source Mem-</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>ber</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Nu-</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>meric</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source String</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Source Time/</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>Low if the same, else Medium</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Source Others</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Low if the same, else High</td>
</tr>
</tbody>
</table>
6.2 Charts

SAP BusinessObjects Web Intelligence offers end-users various possibilities to display data as graphical representations in their reports. These representations are called charts.

In the Web Intelligence semantics, charts can be viewed as report elements of which the type is `Visualization`. As for any report element, the REST Web Service SDK provides you with REST methods to:

- Create charts
- Retrieve chart details
- Update chart settings and expressions
- Delete charts

This section describes the XML grammar for any type of chart supported in this release.

Chart Types [page 58]
Chart Response Body Schema [page 59]

6.2.1 Chart Types

The following list describes the possible types of chart as returned by the call `GET <base_webi_REST_URL>/configuration/visualizations`. For a definition of each chart type, see the SAP BusinessObjects Web Intelligence User’s Guide.

- Bar Series (HorizontalBar, HorizontalStackedBar, HorizontalStackedPercentBar, VerticalBar, VerticalStackedBar, VerticalStackedPercentBar, Bar3D)
- Pie Series (Pie, PieWithDepth, Doughnut)
- Point Series (Scatter, Bubble, PolarScatter, PolarBubble)
- Line
- Surface
- Map Series (TreeMap, HeatMap)
- TagCloud
- BoxPlot
- Radar
- Waterfall
- Dual and combined charts (DualBar, DualLine, CombinedBarLine, and DualCombinedBarLine)
6.2.2 Chart Response Body Schema

Response Body Schema (XML)

(GET .../documents/<documentID>/reports/<reportID>/elements/<elementID>)

```
<element type="Visualization">
  ...
  <content>
    <chart type="string">
      <layout>
      <title>
      <legend>
      <dataLabels>
      <plotArea>
      <graphics>
      <axes>
    </chart>
  </content>
```

The chart type attribute value is a string that specifies the type of chart [page 58].

- Colors [page 59]
- Layout [page 60]
- Title [page 61]
- Legend [page 62]
- Data Labels [page 64]
- Plot Area [page 67]
- Graphics [page 75]
- Axes [page 87]

6.2.2.1 Colors

You can apply colors to different areas of a chart such as titles, legends, labels, or axes. You define a color either with the RGB value or by using a gradient.

Fixed Colors

A color is expressed using the RGB color model and an opacity.
Table 61:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>rgb</td>
<td>The RGB color</td>
<td>The # sign followed by an hexadecimal value</td>
</tr>
<tr>
<td>alpha</td>
<td>The color opacity</td>
<td>[0, 255] from opaque to invisible</td>
</tr>
</tbody>
</table>

Example

```xml
<color rgb="#9d9d9d" alpha="187"/>
```

Gradient

A gradient of colors is defined by a starting color and an ending color. Each color has an RGB value and an opacity. The gradient is oriented according to the **orientation** attribute value (Horizontal | HorizontalInverse | Vertical | VerticalInverse | DiagonalUp | DiagonalUpInverse | DiagonalDown | DiagonalDownInverse).

Example

```xml
<gradient orientation="DiagonalDown">
  <start rgb="#ffffff" alpha="95"/>
  <end rgb="#000000" alpha="95"/>
</gradient>
```

6.2.2.2 Layout

The following table describes the **<layout>** attributes. All attributes are Boolean (true | false).

Table 62:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showDimensionsWithEmptyMeasure Values</td>
<td>Specifies whether the chart displays dimensions when some of their corresponding measures have no value</td>
</tr>
<tr>
<td>showDimensionsWithMeasuresEqualToZero</td>
<td>Specifies whether the chart displays dimensions when some of their corresponding measures have values equal to zero</td>
</tr>
<tr>
<td>showDimensionsWithSumOfMeasuresEqualToZero</td>
<td>Specifies whether the chart displays dimensions whose the corresponding sum of measure values is equal to zero</td>
</tr>
<tr>
<td>showMeasuresWithEmptyDimension Values</td>
<td>Specifies whether the chart displays measures when some of their corresponding dimensions have no value</td>
</tr>
<tr>
<td>showParentNodes</td>
<td>Specifies whether the chart displays the parent of each dimension</td>
</tr>
<tr>
<td>showMissingParent</td>
<td>Specifies whether the chart displays a missing parent value as a node for the dimension</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>showTotal</td>
<td>Specifies whether the chart displays the sum of the measures</td>
</tr>
<tr>
<td>parentAsTotal</td>
<td>Specifies whether the chart displays the dimension parent for the sum of its measures</td>
</tr>
<tr>
<td>duplicateRowAggregation</td>
<td>Specifies whether the chart displays duplicate data. Measure values are not aggregated</td>
</tr>
<tr>
<td>horizontal</td>
<td>Specifies the chart orientation (horizontal or vertical)</td>
</tr>
</tbody>
</table>

### 6.2.2.3 Title

The **visible** Boolean attribute specifies whether the title should be displayed in the chart.

#### Table 63:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;style&gt;</td>
<td>The style of a title defines the following properties:</td>
</tr>
<tr>
<td></td>
<td>- <code>&lt;border&gt;</code>, which is the border thickness and color</td>
</tr>
<tr>
<td></td>
<td>- <code>&lt;background&gt;</code>, which is the background color</td>
</tr>
<tr>
<td></td>
<td>- <code>&lt;font&gt;</code>, which defines title font properties (size, face, italic, bold, strikethrough, underline, and RGB color)</td>
</tr>
<tr>
<td></td>
<td>- <code>&lt;alignment&gt;</code>, which defines the text alignment (horizontal, vertical, and text policy). The text policy can be Wrap, NoWrap or Truncate.</td>
</tr>
<tr>
<td>&lt;label&gt;</td>
<td>The title itself. Can be text or formula.</td>
</tr>
<tr>
<td>&lt;layout&gt;</td>
<td><strong>Attributes:</strong></td>
</tr>
<tr>
<td></td>
<td>- location (Top</td>
</tr>
<tr>
<td></td>
<td>- orientation (Auto</td>
</tr>
<tr>
<td></td>
<td>- spacing, whose value range is [0, 8]</td>
</tr>
<tr>
<td></td>
<td>- adjust (Boolean)</td>
</tr>
</tbody>
</table>

If `adjust` is true, then horizontal and vertical proportionalities can be defined. Their possible types are:

- Auto. No value has to be set.
- Fixed, which defines a value in unit [page 346]
- Proportional, which defines a ratio whose range is [0.0, 1.0] with a step of 0.01

#### Example

```xml
<title visible="true">
  <style>
    <border thickness="None">
      <color rgb="#000000" alpha="255"/>
    </border>
  </style>
</title>
```
Related Information

Colors [page 59]

6.2.2.4 Legend

i Note
The Waterfall chart has no legend.

<legend visible="Boolean" mode="string">
  <style>
    <title><layout><extraInfo>

Table 64:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>visible</td>
<td>Boolean</td>
<td>Specifies whether the legend is displayed in the chart</td>
</tr>
<tr>
<td>mode</td>
<td>Color</td>
<td>Size</td>
</tr>
</tbody>
</table>


Table 65:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;style&gt;</td>
<td>The style of a legend has the following properties:</td>
</tr>
<tr>
<td></td>
<td>• &lt;border&gt;, which defines the border thickness and color</td>
</tr>
<tr>
<td></td>
<td>• &lt;background&gt;, which defines the background color</td>
</tr>
<tr>
<td></td>
<td>• &lt;font&gt;, which defines legend font properties (size, face, italic, bold,</td>
</tr>
<tr>
<td></td>
<td>strikethrough, underline, and RGB color)</td>
</tr>
<tr>
<td></td>
<td>• &lt;alignment&gt;, which defines the text alignment (horizontal, vertical, and</td>
</tr>
<tr>
<td></td>
<td>text policy). The text policy can be Wrap, NoWrap or Truncate.</td>
</tr>
<tr>
<td>&lt;title&gt;</td>
<td>Defines the same properties as a chart title. See Title [page 61].</td>
</tr>
<tr>
<td>&lt;layout&gt;</td>
<td>Attributes:</td>
</tr>
<tr>
<td></td>
<td>• groupByDimension (Boolean)</td>
</tr>
<tr>
<td></td>
<td>• symbolSize, whose value range is [4, 32]</td>
</tr>
<tr>
<td></td>
<td>• location (Top</td>
</tr>
<tr>
<td></td>
<td>• orientation (Auto</td>
</tr>
<tr>
<td></td>
<td>• spacing, whose value range is [0, 8]</td>
</tr>
<tr>
<td></td>
<td>• adjust (Boolean)</td>
</tr>
<tr>
<td></td>
<td>If adjust is true, then horizontal and vertical proportionalities can be</td>
</tr>
<tr>
<td></td>
<td>defined. Their possible types are:</td>
</tr>
<tr>
<td></td>
<td>• Auto. No value has to be set.</td>
</tr>
<tr>
<td></td>
<td>• Fixed, which defines a value in unit [page 346]</td>
</tr>
<tr>
<td></td>
<td>• Proportional, which defines a ratio whose range is [0.0, 1.0] with a step</td>
</tr>
<tr>
<td></td>
<td>of 0.01</td>
</tr>
<tr>
<td>&lt;extraInfo&gt;</td>
<td>i Note</td>
</tr>
<tr>
<td></td>
<td>For BoxPlot, TreeMap, PieWithDepth, Bubble, PolarBubble, and TagCloud charts</td>
</tr>
<tr>
<td></td>
<td>only.</td>
</tr>
<tr>
<td></td>
<td>The visible Boolean attribute specifies whether additional information is</td>
</tr>
<tr>
<td></td>
<td>displayed in the legend.</td>
</tr>
<tr>
<td></td>
<td>The &lt;font&gt; child element defines font properties of additional information</td>
</tr>
<tr>
<td></td>
<td>(size, face, italic, bold, strikethrough, underline, and RGB color).</td>
</tr>
</tbody>
</table>

Example

<legend visible="true">
  <style>
    <border thickness="None"/>
    <color rgb="#000000" alpha="0"/>
  </border>
  <background>
    <color rgb="#000000" alpha="0"/>
  </background>
  <font size="12" face="Helvetica" italic="true" bold="true" strikethrough="false" underline="false" rgb="#00c7ff"/>
  <alignment horizontal="Left" vertical="Center" textPolicy="Wrap"/>
</style>
</legend>
Related Information

Colors [page 59]

6.2.2.5 Data Labels

Note

The BoxPlot and TagCloud charts have no data label.

Table 66:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string (see values below)</td>
<td>The type of data caption to display</td>
</tr>
<tr>
<td>visible</td>
<td>Boolean</td>
<td>Specifies whether the labels should be displayed in the chart</td>
</tr>
</tbody>
</table>

The following table describes the possible type attribute values.
Table 67:

<table>
<thead>
<tr>
<th>Attribute Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto</td>
<td>A label is added automatically.</td>
</tr>
<tr>
<td>Value</td>
<td>The data caption displays the value. This corresponds to a value axis.</td>
</tr>
<tr>
<td>Label</td>
<td>The data caption displays the category. This corresponds to a category axis.</td>
</tr>
<tr>
<td>Percent</td>
<td>The data caption displays the values in percent.</td>
</tr>
<tr>
<td>LabelAndValue</td>
<td>The data caption displays the category followed by the value.</td>
</tr>
<tr>
<td>LabelAndPercent</td>
<td>The data caption displays the category followed by the value in percent.</td>
</tr>
<tr>
<td>XValue</td>
<td>The data caption displays the second value corresponding to the second value axis.</td>
</tr>
<tr>
<td>Weight</td>
<td>The data caption displays the category followed by the value.</td>
</tr>
<tr>
<td>Scale</td>
<td>The data caption displays the value corresponding to the bubble width.</td>
</tr>
</tbody>
</table>

The charts do not display all data label types. The following table shows the possible types of data labels per type of chart.

Table 68:

<table>
<thead>
<tr>
<th>Attribute Value</th>
<th>Type of chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto</td>
<td>All except Waterfall</td>
</tr>
<tr>
<td>Value</td>
<td>All</td>
</tr>
<tr>
<td>Label</td>
<td>All except Line, DualLine, Surface, Radar, and HeatMap</td>
</tr>
<tr>
<td>Percent</td>
<td>All except Point series, Radar, HeatMap, and Waterfall</td>
</tr>
<tr>
<td>LabelAndValue</td>
<td>Pie series only</td>
</tr>
<tr>
<td>LabelAndPercent</td>
<td>Pie series only</td>
</tr>
<tr>
<td>XValue</td>
<td>Point series only</td>
</tr>
<tr>
<td>Weight</td>
<td>TreeMap only</td>
</tr>
<tr>
<td>Scale</td>
<td>Bubble and PolarBubble charts only</td>
</tr>
</tbody>
</table>

The following table shows the `<dataLabels>` children elements.

Table 69:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<style>` | The style of a label defines the following properties:  
  - `<border>`, which is the border thickness and color  
  - `<background>`, which is the background color  
  - `<font>`, which defines the label font properties (size, face, italic, bold, strikethrough, underline, and RGB color)  
  - `<alignment>`, which defines the text alignment (horizontal, vertical, and text policy). The text policy can be Wrap, NoWrap or Truncate. |
**Element**

<table>
<thead>
<tr>
<th><strong>Element</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;layout&gt;</td>
<td><strong>Attributes:</strong></td>
</tr>
<tr>
<td></td>
<td>● autoHiding, which defines whether labels are hidden automatically when there is not enough space</td>
</tr>
<tr>
<td></td>
<td>● resolveOverlapping, which defines whether the overlapped labels are resolved</td>
</tr>
<tr>
<td></td>
<td>● position(InsideFirstOutsideOtherwise</td>
</tr>
<tr>
<td></td>
<td>● orientation(Horizontal</td>
</tr>
<tr>
<td></td>
<td>● spacing, which defines the margin and whose value range is [0, 8]</td>
</tr>
<tr>
<td></td>
<td>● mode, which defines the display mode for Pie charts only(Side</td>
</tr>
<tr>
<td></td>
<td>● singleLine, which defines whether the data label is on one single line (for LabelAndValue or LabelAndPercent type only)</td>
</tr>
<tr>
<td></td>
<td>● percentMinValue, which defines the minimum percent value to display for Pie charts only. Value range is [0.0, 100.0] with a step of 0.01.</td>
</tr>
</tbody>
</table>

Layout also shows the <separator> child element to define the starting and ending symbols to use in data labels in the case of a Pie chart. The string maximum length for a value is 256.

| **<format>** | The expression to use to format and display tick values. |

### Example

```xml
<dataLabels type="LabelAndPercent" visible="false">
  <style>
    <border thickness="None">
      <color rgb="#000000" alpha="255"/>
    </border>
    <background>
      <color rgb="#000000" alpha="0"/>
    </background>
    <font size="8" face="Arial" italic="false" bold="true" strikethrough="false" underline="false" rgb="#707070"/>
    <alignment textPolicy="NoWrap"/>
  </style>
  <layout autoHiding="true" resolveOverlapping="false" mode="Side" position="Outside" spacing="0" singleLine="true" percentMinValue="10.0">
    <separator>
      <start>{</start>
      <end>}</end>
    </separator>
    <format default="true" sample="1234,57" type="Custom">
      <template positive="0.00"/>
    </format>
  </layout>
</dataLabels>
```

**Note**

For Percent or LabelAndPercent type only.
Related Information

Colors [page 59]

6.2.2.6  Plot Area

**Note**
The HeatMap chart has no plot area.

```xml
<plotArea>
  <xSeries>
  </xSeries>
  <title>
  </title>
  <background>
  </background>
  <grids>
</plotArea>
```

**Table 70:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;xSeries&gt;</code></td>
<td>Defines specific layout properties for each chart type except Pie charts.</td>
</tr>
<tr>
<td><code>&lt;title&gt;</code></td>
<td>Defines the same properties of a chart title. See Title [page 61].</td>
</tr>
<tr>
<td><code>&lt;background&gt;</code></td>
<td>The background area can have one color (plain) or two colors (striped).</td>
</tr>
<tr>
<td></td>
<td>• Plain</td>
</tr>
<tr>
<td></td>
<td>&lt;background mode=&quot;Plain&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;coloring&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;color rgb=&quot;#000000&quot; alpha=&quot;0&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/coloring&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/background&gt;</td>
</tr>
<tr>
<td></td>
<td>• Stripped</td>
</tr>
<tr>
<td></td>
<td>&lt;background mode=&quot;Striped&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;coloring lightingAdjustment=&quot;0.9&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;color rgb=&quot;#00ff00&quot; alpha=&quot;255&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/coloring&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;coloring lightingAdjustment=&quot;0.9&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;color rgb=&quot;#ff00ff&quot; alpha=&quot;255&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/coloring&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/background&gt;</td>
</tr>
</tbody>
</table>

lightingAdjustment defines the lighting color adjustment value. Value range is [0.0, 1.0] with a step of 0.01.
**6.2.2.6.1 Bar, Line, and Surface Series Layout**

The following table describes the attributes common to the all charts of the Bar, Line, and Surface series (<barSeries>, <lineSeries>, and <surfaceSeries> elements of the chart definition).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashedLines</td>
<td>Boolean</td>
<td>Specifies whether lines are displayed as dashed lines</td>
</tr>
<tr>
<td>invertSuperimpositionOrder</td>
<td>Boolean</td>
<td>Specifies whether the superimposition order of the layers plotted to each data series is inverted. In the case where lines or surfaces are stacked, this setting has no effect.</td>
</tr>
<tr>
<td>spacingBetweenGroups</td>
<td>double</td>
<td>The space between groups. Value range is [-1.0, 1.0] with a step of 0.01.</td>
</tr>
<tr>
<td>spacingWithinGroups</td>
<td>double</td>
<td>The space within groups. Value range is [-1.0, 1.0] with a step of 0.01.</td>
</tr>
</tbody>
</table>

The following attributes are specific to the Bar3D chart.
### Table 72:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showFloor</td>
<td>Boolean</td>
<td>Specifies whether a virtual 3D floor is displayed</td>
</tr>
<tr>
<td>showFirstEdge</td>
<td>Boolean</td>
<td>Specifies whether the 3D wall first edge is displayed</td>
</tr>
<tr>
<td>showSecondEdge</td>
<td>Boolean</td>
<td>Specifies whether the 3D wall second edge is displayed</td>
</tr>
</tbody>
</table>

#### Example

**Bar Series**

```xml
<plotArea>
  <barSeries dashedLines="false" spacingBetweenGroups="0.2" spacingWithinGroups="0.2"/>
</plotArea>
```

#### Box Series Layout

The following table describes the attributes specific to the BoxPlot chart definition (`<boxSeries>`).

### Table 73:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashedLines</td>
<td>Boolean</td>
<td>Specifies whether lines are displayed as dashed lines</td>
</tr>
</tbody>
</table>
### Example

```xml
<plotArea>
    <boxSeries dashedLines="true" hideOutliers="true" spacingBetweenItems="0.2"/>
    <background mode="Plain">
        <coloring>
            <color alpha="0" rgb="#000000"/>
        </coloring>
    </background>
    <grids>
        <grid type="Vertical">
            <color alpha="0" rgb="#000000"/>
        </grid>
        <grid type="Horizontal">
            <color alpha="255" rgb="#e7e7e7"/>
        </grid>
    </grids>
</plotArea>
```

### 6.2.2.6.3 Map Series Layout

The following table describes the attributes specific to the TreeMap chart definition.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showTreeMapRoot</td>
<td>Boolean</td>
<td>Specifies whether the root node of the hierarchy is displayed</td>
</tr>
<tr>
<td>depth</td>
<td>integer</td>
<td>The depth for 3D look in proportion to chart size. Value range is [-1, 16].</td>
</tr>
<tr>
<td>fixParentWeight</td>
<td>Fix</td>
<td>Strict</td>
</tr>
<tr>
<td>hierarchicalView</td>
<td>Boolean</td>
<td>Specifies whether zones are displayed in a hierarchical view.</td>
</tr>
</tbody>
</table>

The `<zoneTitle>` element has the `visible` Boolean attribute that defines whether the title of the zone should be displayed in the chart.

The following table shows the `<zoneTitle>` children elements.
Table 75:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;font&gt;</code></td>
<td>The font properties of the zone title (size, face, italic, bold, strikethrough, underline, and RGB color)</td>
</tr>
<tr>
<td><code>&lt;alignment&gt;</code></td>
<td>The text alignment (horizontal, vertical, and text policy). The text policy can be Wrap, NoWrap or Truncate.</td>
</tr>
<tr>
<td><code>&lt;layout&gt;</code></td>
<td>Attributes:</td>
</tr>
<tr>
<td></td>
<td>● orientation(Auto</td>
</tr>
<tr>
<td></td>
<td>● spacing, whose value is in unit [page 346] and the metric range is [0, 1500]</td>
</tr>
</tbody>
</table>

Example

```
<plotArea>
  <mapSeries showTreeMapRoot="true" depth="-1" fixParentWeight="Fix"
    hierarchicalView="true">
    <zoneTitle visible="true">
      <font size="8" face="Arial" italic="false" bold="true"
strikethrough="false" underline="false" rgb="#ffffff"/>
      <alignment horizontal="Center" vertical="Center"
textPolicy="Truncate"/>
      <layout orientation="Auto" spacing="0.417"/>
    </zoneTitle>
  </mapSeries>
</plotArea>
```

6.2.2.6.4 Point Series Layout

The following table describes the attributes common to all charts of the Point series (`<pointSeries>`).

Table 76:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashedLines</td>
<td>Boolean</td>
<td>Specifies whether lines are displayed as dashed lines</td>
</tr>
<tr>
<td>invertSuperimpositionOrder</td>
<td>Boolean</td>
<td>Specifies whether the superimposition order of the layers plotted to each data series is inverted. In the case where lines or surfaces are stacked, this setting has no effect.</td>
</tr>
</tbody>
</table>

The following attributes are specific to the Bubble and PolarBubble charts.

Table 77:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scale</td>
<td>integer</td>
<td>The ratio between the plot area size and the biggest bubble diameter. Value range is [2, 10].</td>
</tr>
</tbody>
</table>
### 6.2.2.6.5 Radar Series Layout

The following table shows the attributes specific to the Radar chart definition (`<radarSeries>`).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invertSuperimpositionOrder</td>
<td>Boolean</td>
<td>Specifies whether the superimposition order of the layers plotted to each data series is inverted. In the case where lines or surfaces are stacked, this setting has no effect.</td>
</tr>
<tr>
<td>bringToFront</td>
<td>Boolean</td>
<td>Specifies whether the grid is in front of the data</td>
</tr>
<tr>
<td>polygonal</td>
<td>Boolean</td>
<td>Specifies whether the grid is displayed as polygonal lines as opposed to circles</td>
</tr>
</tbody>
</table>

**Example**

```xml
<plotArea>
  <radarSeries invertSuperimpositionOrder="true" bringToFront="true" polygonal="true"/>
  <background mode="Plain">
    <coloring>
      <color alpha="89" rgb="#ffffff"/>
    </coloring>
  </background>
</plotArea>
```
### 6.2.2.6.6 Tag Cloud Series Layout

The following tables describe the attributes specific to the TagCloud chart definition (`<tagCloudSeries>`).

**Table 79:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>levelColoring</td>
<td>integer</td>
<td>The depth of coloring areas. Value range is [0, 64].</td>
</tr>
<tr>
<td>comparator</td>
<td>Weight</td>
<td>Names</td>
</tr>
<tr>
<td>mode</td>
<td>Row</td>
<td>Column</td>
</tr>
<tr>
<td>alignment</td>
<td>Left</td>
<td>Center</td>
</tr>
<tr>
<td>orientation</td>
<td>MainlyHorizontal</td>
<td>HorizontalAndVertical</td>
</tr>
<tr>
<td>fillRate</td>
<td>double</td>
<td>The screen fill rate with words. Value range is [0.100, 1.500] with a step of 0.025.</td>
</tr>
<tr>
<td>spacingBetweenTags</td>
<td>Auto</td>
<td>Fixed</td>
</tr>
<tr>
<td>spacingValue</td>
<td>double</td>
<td>The space value in unit [page 346] if spacingBetweenTags is Fixed.</td>
</tr>
</tbody>
</table>

The following table describes the `<tag>` attributes.

**Table 80:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxFontRatio</td>
<td>Auto</td>
<td>Fixed</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type or Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>maxFontSize</td>
<td>integer</td>
<td>The word maximum font size if maxFontRatio is Fixed. Value range is [1, 56].</td>
</tr>
<tr>
<td>minFontRatio</td>
<td>Auto</td>
<td>Fixed</td>
</tr>
<tr>
<td>minFontSize</td>
<td>integer</td>
<td>The word minimum font size if minFontRatio is Fixed. Value range is [1, 256].</td>
</tr>
<tr>
<td>minVisibleFontSize</td>
<td>integer</td>
<td>The font size threshold under which words are removed. Value range is [1, 50].</td>
</tr>
<tr>
<td>fontScaling</td>
<td>Linear</td>
<td>Exponential</td>
</tr>
</tbody>
</table>

<font> defines the title font properties (face, italic, bold, strikethrough, underline, and RGB color). Font size is ignored. See Title [page 61].

### Example

```xml
<plotArea>
  <tagCloudSeries levelColoring="45" comparator="Weight" mode="Wordle"
                   orientation="HorizontalAndVertical"
                   fillRate="0.65" spacingBetweenTags="Fixed" spacingValue="0.097">
    <tag maxFontRatio="Auto" minFontRatio="Auto" minVisibleFontSize="4"
         fontScaling="Logarithmic">
      <font size="6" face="Arial" italic="false" bold="true"
             strikethrough="false" underline="false" rgb="#555555"/>
    </tag>
  </tagCloudSeries>
</plotArea>
```

### 6.2.2.6.7 Waterfall Series Layout

```xml
<waterfallSeries>
  <line>
    <color>
```

The following table describes the attributes specific to the Waterfall chart definition.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashedLines</td>
<td>Boolean</td>
<td>Specifies whether lines are displayed as dashed lines</td>
</tr>
<tr>
<td>referenceLine</td>
<td>Boolean</td>
<td>The reference line</td>
</tr>
<tr>
<td>spacingBetweenItems</td>
<td>double</td>
<td>The space between groups. Value range is [-1.000, 1.000], with a step of 0.01.</td>
</tr>
</tbody>
</table>

<line> thickness is defined via the width attribute with a range of [1, 7] and the color by using a <color> child element.
6.2.2.7 Graphics
Table 82:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;coloring&gt;</td>
<td>Specifies the palette and coloring method used by the chart. Both built-in [page 249] or custom [page 251] palettes can be used.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dual charts can define two palettes:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Point, Map, TagCloud and Waterfall Series have specific coloring methods. See Colorings [page 77].</td>
</tr>
<tr>
<td>&lt;rendering&gt;</td>
<td>Attributes for common rendering:</td>
</tr>
<tr>
<td></td>
<td>• filter, which defines the light and shadow rendering filter (None</td>
</tr>
<tr>
<td></td>
<td>• look3D (Boolean)</td>
</tr>
<tr>
<td></td>
<td>Attributes specific to Bar3D charts:</td>
</tr>
<tr>
<td></td>
<td>• shading is the 3D lighting mode (Faceted</td>
</tr>
<tr>
<td></td>
<td>• shape is the 3D bar shape (Box</td>
</tr>
<tr>
<td></td>
<td>You can set further options depending on the chart type. See Rendering [page 80].</td>
</tr>
<tr>
<td>&lt;effects&gt;</td>
<td>The rendering effects of the graphics. They are specific to each chart series. See Effects [page 82].</td>
</tr>
<tr>
<td></td>
<td>i Note</td>
</tr>
</tbody>
</table>
6.2.2.7.1 Colorings

Point Series

The following table shows the `<pointSeries>` children elements.

Table 83:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;colorGroup&gt;</code></td>
<td>The color of a group</td>
</tr>
<tr>
<td>Attributes:</td>
<td></td>
</tr>
<tr>
<td>● type (Fixed</td>
<td>Deepest)</td>
</tr>
<tr>
<td>● depth, whose minimum is 1</td>
<td></td>
</tr>
<tr>
<td><code>&lt;shapeGroup&gt;</code></td>
<td>The shape of a group</td>
</tr>
<tr>
<td>Attributes:</td>
<td></td>
</tr>
<tr>
<td>● type (Fixed</td>
<td>Deepest)</td>
</tr>
<tr>
<td>● depth, whose minimum is 1</td>
<td></td>
</tr>
</tbody>
</table>

Example

```xml
<coloring>
  <palettes>
    <palette alpha="0" refId="SAP Standard 2011"/>
  </palettes>
  <pointSeries>
    <colorGroup type="Fixed" depth="1"/>
    <shapeGroup type="Deepest"/>
  </pointSeries>
</coloring>
```

Map Series and TagCloud Series

Some specific coloring methods are provided for the Map Series and the TagCloud chart (`<mapSeries>` and `<tagCloudSeries>`).

Table 84:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>Palette</td>
<td>GradientBased</td>
</tr>
<tr>
<td>useInternalPalette</td>
<td>Boolean</td>
<td>Specifies whether the built-in palette is used</td>
</tr>
</tbody>
</table>

The following table describes their common children elements.
<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<gradientPalette>` | Defines the gradient color palette if method="GradientBased". **Attributes:**  
- **type**, which defines the number of colors in gradient. Possible values are `Colors2` for two colors and `Colors3` for 3 colors.  
- **start**, which defines the first color of the gradient color palette  
- **middle**, which defines the middle color of the gradient color palette  
- **end**, which defines the last color of the gradient color palette |
| `<OutOfRange>` | The color of the out of ranges values |
| `<nullOrEmpty>` | The color of the null or empty values |
| `<measurePolarityGradientPalette>` | Defines the gradient palette if method="GradientBasedNeutralPolarity". **Attributes:**  
- **type**  
- **start**  
- **middle**  
- **end** |
| `<data>` | Attributes:  
- **distributionMode** that defines how data is distributed in the coloring zone (ByValues|ByQuantiles)  
- **intervalSyntax** that defines how data intervals display in the legend (Basic|US|ISO31-11) |
| `<ranges>` | Defines the zones and the associated colors. **Attributes if method is not "CustomRange":**  
- **number ([0, 64])**  
- **from** that defines the beginning of the interval values from which the color method is applied  
- **to** that defines the end of the interval values to which the color method is applied  
**Attributes if method="CustomRange":**  
- **percentage(Boolean)**  
**Child element if method="CustomRange":** `<range>` |
| `<range>` | Defines the minimum and maximum values for the color of the zone defined by the interval if method="CustomRange". **Attributes:**  
- **from** that defines the beginning of the interval values from which the custom color method is applied  
- **to** that defines the end of the interval values to which the custom color method is applied |
Example

Map Series

```
<mapSeries method="CustomRange" useInternalPalette="true">
  <OutOfRange>
    <color rgb="#c0c0c0" alpha="117"/>
  </OutOfRange>
  <nullOrEmpty>
    <color rgb="#e0e0e0" alpha="125"/>
  </nullOrEmpty>
  <data distributionMode="ByQuantiles" intervalSyntax="ISO31-11"/>
  <ranges percentage="true">
    <range from="0.0" to="33.0">
      <color rgb="#ff0000" alpha="255"/>
    </range>
    <range from="33.0" to="67.0">
      <color rgb="#000000" alpha="255"/>
    </range>
    <range from="67.0" to="100.0">
      <color rgb="#00ff00" alpha="255"/>
    </range>
  </ranges>
</mapSeries>
```

Example

Tag Cloud Series

```
<tagCloudSeries method="Palette">
  <OutOfRange>
    <color rgb="#c0c0c0" alpha="117"/>
  </OutOfRange>
  <nullOrEmpty>
    <color rgb="#e0e0e0" alpha="125"/>
  </nullOrEmpty>
  <data distributionMode="ByQuantiles" intervalSyntax="ISO31-11"/>
  <ranges number="5" from="-4.497" to="220000.0">
    <color rgb="#00ff00" alpha="255"/>
  </ranges>
</tagCloudSeries>
```

Waterfall Series

```
<waterfallSeries> contains all settings to configure the color of some specified values. Each of the following
<waterfallSeries> children elements has the type attribute (Auto|Fixed). If type is Fixed, then a <color>
has to be set. Gradients of colors are allowed for the start and total values only.
```
Table 86:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;start&gt;</td>
<td>The color of the start value</td>
</tr>
<tr>
<td>&lt;total&gt;</td>
<td>The color of the total value</td>
</tr>
<tr>
<td>&lt;negative&gt;</td>
<td>The color of the negative values</td>
</tr>
<tr>
<td>&lt;positive&gt;</td>
<td>The color of the positive values</td>
</tr>
</tbody>
</table>

Example

```xml
<coloring>
  ...
  <waterfallSeries>
    <start type="Fixed">
      <color rgb="#00ff00" alpha="130"/>
    </start>
    <total type="Fixed">
      <color rgb="#0000ff" alpha="110"/>
    </total>
    <negative type="Auto"/>
    <positive type="Auto"/>
  </waterfallSeries>
</coloring>
```

6.2.2.7.2 Rendering

Edge

<edge> allows you to configure the borders of the chart elements. The following table describes the <edge> attributes.

Table 87:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>None</td>
<td>Default</td>
</tr>
<tr>
<td>width</td>
<td>integer</td>
<td>The line thickness that you can set to Bar3D charts. Value range is [1, 7].</td>
</tr>
</tbody>
</table>

You can also set a color to an edge by using the <color> child element.

Rotation

The following table shows the charts that support the rotation of the graph and the corresponding <rotation> attributes.
Table 88:

<table>
<thead>
<tr>
<th>Chart</th>
<th>&lt;rotation&gt; Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bar3D</td>
<td>The rotation angle values for the 3D orientation:</td>
</tr>
<tr>
<td></td>
<td>● xAngle on the X axis. Value range is [-90, 90].</td>
</tr>
<tr>
<td></td>
<td>● yAngle on the Y axis. Value range is [-180, 180].</td>
</tr>
<tr>
<td>Pie series</td>
<td>● startAngle defines the start angle of the Pie in degrees. Value range is [0, 360] with a step of 1.</td>
</tr>
<tr>
<td></td>
<td>● clockwise is a Boolean that defines the Pie rotation</td>
</tr>
<tr>
<td>Polar Bubble and Polar Scatter</td>
<td>● startAngle defines the start angle of the chart in degrees. Possible values are 0, 90, 180, 270, and 360.</td>
</tr>
<tr>
<td></td>
<td>● clockwise is a Boolean that defines the chart rotation</td>
</tr>
<tr>
<td>Radar</td>
<td>clockwise is a Boolean that defines the Radar rotation</td>
</tr>
</tbody>
</table>

Shadow

<shadow> allows you to configure the shadow properties of a chart element. The following table describes the <shadow> attributes.

Table 89:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>effect</td>
<td>None</td>
<td>OneSided</td>
</tr>
<tr>
<td>xOffset</td>
<td>double</td>
<td>The horizontal distance or offset of the shadow from each chart element in unit [page 346].</td>
</tr>
<tr>
<td>yOffset</td>
<td>double</td>
<td>The vertical distance, or offset of the shadow from each chart element in unit [page 346].</td>
</tr>
<tr>
<td>filterPassCount</td>
<td>integer</td>
<td>The complexity of the effect. When a visual complexity is increased, performance may be decreased. Value range is [1, 9] with a step of 1.</td>
</tr>
<tr>
<td>filterWindowSize</td>
<td>integer</td>
<td>The smoothness of the shadow. The higher the value, the smoother the effect. Value range is [3, 9] with a step of 2.</td>
</tr>
<tr>
<td>lightPower</td>
<td>double</td>
<td>The light intensity. Value range is [-1.000, +1.000] with a step of 0.05.</td>
</tr>
</tbody>
</table>

You can also set a color to a shadow by using the <color> child element.

Example

```xml
<rendering>
  <edge type="Override" width="7">
    <color rgb="#ff00ff" alpha="189"/>
  </edge>
</rendering>
```
6.2.2.7.3 Effects

<effects> allows you to configure special effects for some chart series.

- Pie Series [page 82]
- Bar Series, Box Chart, and Waterfall Series [page 83]
- Line Series, Radar Series [page 84]
- Surface Series [page 85]
- Point Series [page 85]
- Map Series [page 86]
- Combined Charts [page 86]

Pie Series

<pieSeries> allows you to configure special effects for the Pie charts. The following table describes the <pieSeries> attributes.

Table 90:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>effect</td>
<td>None</td>
<td>Halo</td>
</tr>
<tr>
<td>material</td>
<td>None</td>
<td>BrushedMetal1</td>
</tr>
<tr>
<td>depth</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>outerRadius</td>
<td>integer</td>
<td></td>
</tr>
</tbody>
</table>

Example

<effects>
  <pieSeries effect="Button3D" material="Water" depth="1.208"/>
</effects>
Bar Series, Box Chart, and Waterfall Series

The following table describes the attributes common to `<barSeries>`, `<boxSeries>`, and `<waterfallSeries>`.

### Table 91:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>effect</td>
<td>None</td>
<td>Volume</td>
</tr>
<tr>
<td>progressiveAlpha</td>
<td>Boolean</td>
<td>Specifies whether there is progressive transparency along bars</td>
</tr>
<tr>
<td>roundedCorners</td>
<td>Boolean</td>
<td>Specifies whether bar corners are rounded</td>
</tr>
<tr>
<td>effectWidth</td>
<td>double</td>
<td>The width ratio value for the volume effect. Value range is [0.000, 0.500] with a step of 0.01.</td>
</tr>
<tr>
<td>colorBrighter</td>
<td>double</td>
<td>The lighting color adjustment value to apply when displaying volume effects on top of lines. Value range is [1.000, 2.000] with a step of 0.01.</td>
</tr>
<tr>
<td>colorLessBright</td>
<td>double</td>
<td>The lighting color adjustment value applied when displaying a volume effect on bottom of lines. Value range is [0.000, 1.000] with a step of 0.01.</td>
</tr>
</tbody>
</table>

#### Example

**Bar Series**

```xml
<effects>
  <barSeries effect="None" roundedCorners="false"/>
</effects>
```

**Box Series**

```xml
<effects>
  <boxSeries effect="Cylinder"/>
</effects>
```

**Waterfall Series**

```xml
<effects>
  <waterfallSeries effect="Gradient" progressiveAlpha="true"/>
</effects>
```
Line Series, Radar Series

The following table describes the attributes common to `<lineSeries>` and `<radarSeries>`. However, the `style` attribute is specific to `<lineSeries>`.

### Table 92:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>effect</td>
<td>None</td>
<td>Volume</td>
</tr>
<tr>
<td>width</td>
<td>integer</td>
<td>The line thickness. Value range is [0, 7].</td>
</tr>
<tr>
<td>style</td>
<td>Solid</td>
<td>Dot</td>
</tr>
<tr>
<td>spline</td>
<td>Boolean</td>
<td>Specifies whether data points are connected using cardinal spline curves (as opposed to straight lines)</td>
</tr>
<tr>
<td>colorBrighter</td>
<td>double</td>
<td>The lighting color adjustment value to apply when displaying volume effects on top of lines. Value range is [1.000, 2.000] with a step of 0.01.</td>
</tr>
<tr>
<td>colorLessBright</td>
<td>double</td>
<td>The lighting color adjustment value applied when displaying a volume effect on bottom of lines. Value range is [0.000, 1.000] with a step of 0.01.</td>
</tr>
<tr>
<td>transparency</td>
<td>integer</td>
<td>The transparency factor of the area defined by the lines in the Radar charts only. Value range is [0, 55].</td>
</tr>
</tbody>
</table>

`<marker>` defines the symbol to display at the measure points of the chart. The following table describes the `<marker>` attributes.

### Table 93:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>visible</td>
<td>Boolean</td>
<td>Specifies whether the marker is displayed</td>
</tr>
<tr>
<td>size</td>
<td>integer</td>
<td>The marker size. Value range is [4, 32].</td>
</tr>
<tr>
<td>autoResize</td>
<td>Boolean</td>
<td>Specifies whether the marker symbols are resized automatically according to the chart size</td>
</tr>
</tbody>
</table>

The following table describes the `<marker>` children elements.
Table 94:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;color&gt;</td>
<td>The symbol color</td>
</tr>
<tr>
<td>&lt;edge&gt;</td>
<td>The marker border. A color or a gradient of colors can also be defined.</td>
</tr>
<tr>
<td></td>
<td><strong>Attribute</strong>: type (None</td>
</tr>
<tr>
<td></td>
<td><strong>Child Element</strong>: &lt;color&gt;</td>
</tr>
<tr>
<td>&lt;symbols&gt;</td>
<td>The symbols set used as markers</td>
</tr>
<tr>
<td></td>
<td><strong>Child Element</strong>: &lt;symbol&gt;(Circle</td>
</tr>
</tbody>
</table>

**Example**

```xml
<effects>
  <lineSeries effect="Volume" style="Dash" width="7" spline="true"
  colorBrighter="1.5" colorLessBright="0.7">
    <marker visible="true" size="32">
      <edge type="Override">
        <color alpha="255" rgb="#ff00ff"/>
      </edge>
      <symbols>
        <symbol>Circle</symbol>
        <symbol>Star</symbol>
        <symbol>Diamond</symbol>
        <symbol>Square</symbol>
      </symbols>
    </marker>
  </lineSeries>
</effects>
```

**Surface Series**

The only `<surfaceSeries>` attribute is `progressiveAlpha` that specifies whether there is progressive transparency along the surface.

**Point Series**

The following table describes the `<pointSeries>` attributes.

Table 95:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>effect</td>
<td>None</td>
<td>Halo</td>
</tr>
<tr>
<td>showCenter</td>
<td>Boolean</td>
<td>Specifies whether a cross marker is displayed at the center of the symbol</td>
</tr>
</tbody>
</table>
A `<marker>` is also defined to configure the symbol used at the measure points of the chart. See the section above for details.

**Example**

```xml
<effects>
    <pointSeries effect="Moonlight" showCenter="false">
        <marker visible="true" size="16">
            <color alpha="38" rgb="#ff00ff"/>
            <edge type="Override">
                <color alpha="181" rgb="#000000"/>
            </edge>
            <symbols>
                <symbol>Star</symbol>
                <symbol>Square</symbol>
                <symbol>Circle</symbol>
                <symbol>Diamond</symbol>
            </symbols>
        </marker>
    </pointSeries>
</effects>
```

**Map Series**

The only `<mapSeries>` attribute is `zoneEmboss` that specifies whether an embossed zone is displayed.

**Combined Charts**

Combined charts can have the effects of more than one series. For example, the `DualCombinedBarLine` and `CombinedBarLine` charts combine the effects of both Bar and Line Series.

**Example**

```xml
<DualCombinedBarLine>
<effects>
    <lineSeries effect="None" width="2" spline="true">
        <marker visible="true" size="8">
            <edge type="Override">
                <color alpha="125" rgb="#00ff00"/>
            </edge>
            <symbols>
                <symbol>Circle</symbol>
                <symbol>Star</symbol>
                <symbol>Diamond</symbol>
                <symbol>Square</symbol>
            </symbols>
        </marker>
    </lineSeries>
</effects>
```
6.2.2.8 Axes

<axes>
  <axis role="string" visible="Boolean" optional="Boolean">
    <id>
    <name>
    <title>
    <layout>
    <coloring>
    <grid>
    <tick>
    <labels>
    <stacking>
    <scaling>
    <expressions>
  ...

Table 96:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>string</td>
<td>The unique role of the axis. See Roles and Identifiers [page 90] for values.</td>
</tr>
<tr>
<td>visible</td>
<td>Boolean</td>
<td>Specifies whether the axis is displayed in the chart</td>
</tr>
<tr>
<td>optional</td>
<td>Boolean</td>
<td>Specifies whether you must assign expressions to the axis</td>
</tr>
</tbody>
</table>

Table 97:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;id&gt;</td>
<td>The axis identifier. This positive integer is specific to each chart axis.</td>
</tr>
<tr>
<td>&lt;name&gt;</td>
<td>The internal name of the axis as a string. Cannot be changed.</td>
</tr>
<tr>
<td>&lt;title&gt;</td>
<td>The axis Title [page 61]. In addition, the &lt;separator&gt; element defines the character used as label separator in the axis title when no custom label is defined. Maximum length is 256.</td>
</tr>
<tr>
<td>&lt;layout&gt;</td>
<td>Attributes:</td>
</tr>
<tr>
<td></td>
<td>● reorder, a Boolean that defines whether labels order is reversed on the category axis</td>
</tr>
<tr>
<td></td>
<td>● continuous, a Boolean that defines the layout of the category axis in a continuous mode</td>
</tr>
</tbody>
</table>
|         | ● adjust (Boolean)
<p>|         | If adjust is true, then horizontal and vertical proportionalities can be defined. Their possible types are: |
|         | ● Auto. No value has to be set. |
|         | ● Fixed, which defines a value in unit [page 346] |
|         | ● Proportional, which defines a ratio whose range is [0.0, 1.0] with a step of 0.01 |
| &lt;coloring&gt; | The axis Colors [page 59]. |</p>
<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;grid&gt;</td>
<td>The color and background color of the grid around axis labels. See Colors [page 59].</td>
</tr>
<tr>
<td></td>
<td>i Note For category axes only.</td>
</tr>
</tbody>
</table>
| <tick> | Attributes:  
- color  
- length, whose value range is [2, 8]  
- margin between the tick and its label, with range [0, 8]  
- densityMode, that can be either Auto or Fixed  
- density, whose value range is [0, 4] if densityMode is Fixed |
| <labels> | The axis Labels [page 91]. |
| <stacking> | Attributes of the axis stacking mode:  
- mode (Unstacked|Stacked|GloballyStacked)  
- stacked100percent (true|false) |
|         | i Note For value axes only. |
| <scaling> | The axis scaling mode used to determine the axis ticks. See Scaling [page 92]. |
|         | i Note For value axes only. |
| <expressions> | The formulas assigned to the axis. The number of expressions depends on the chart type and axis role. For example, you can assign only one expression per axis in the TagCloud chart.  
Attribute of an expression of a Bar, Line, Surface, Dual, or Combined chart:  
- regionType (Default|Bars|Lines|Surfaces)  
Attribute of an expression of a HeatMap, TreeMap, or TagCloud chart for the measures on value axes:  
- polarity (Auto|Ascending|Descending|Neutral) |

**Example**

**Value Role Axis**

```xml
<axis role="Value" visible="true" optional="false">
  <id>2</id>
  <name>Value Axis 1</name>
  <title visible="true">
    <style>
      <border thickness="None"/>
      <color rgb="#000000" alpha="255"/>
    </style>
  </title>
</axis>
```
Example

Category Role Axis

```xml
<axis role="MainCategory" visible="true" optional="false">
  <id>0</id>
  <name>Main Category Axis</name>
  <title visible="true">
    <style>
      <border thickness="None">
        <color alpha="255" rgb="#000000"/>
      </border>
      <background>
        <color alpha="0" rgb="#000000"/>
      </background>
      <font size="8" face="Arial" italic="false" bold="true" strikethrough="false" underline="false" rgb="#707070"/>
    </style>
    <layout spacing="0"/>
    <separator>&amp;</separator>
    <layout reverseOrder="false" continuous="false" adjust="false"/>
    <coloring>
      <color alpha="255" rgb="#707070"/>
    </coloring>
    <grid>
      <color alpha="255" rgb="#dadada"/>
      <background>
        <color alpha="0" rgb="#000000"/>
      </background>
      <font size="8" face="Arial" italic="false" bold="true" strikethrough="false" underline="false" rgb="#707070"/>
    </grid>
    <tick length="4" margin="0"/>
    <labels visible="true" orientation="Auto" deleteMode="Auto" autoResize="false">
      <font size="8" face="Arial" italic="false" bold="true" strikethrough="false" underline="false" rgb="#707070"/>
      <numberFormat default="true" sample="1,234567E3" type="Custom">
        <template positive="SCIENTIFIC"/>
      </numberFormat>
      <stacking mode="GloballyStacked" stacked100percent="false"/>
      <scaling mode="Linear" unitScaleFactor="0" roundMinMaxValues="false">
        <minValue type="Auto"/>
        <maxValue type="Auto"/>
      </scaling>
      <expressions>
        <formula regionType="Default" dataType="Numeric" dataObjectId="DP0.DO7">={Revenue}</formula>
      </expressions>
    </labels>
  </title>
</axis>
```
### 6.2.2.8.1 Roles and Identifiers

A chart has a certain number of axes and each axis has a certain role. The following table describes the axis role per identifier (from 0 to 5) for every chart or chart series.

Table 98:

<table>
<thead>
<tr>
<th>Chart</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bar Series, Line, Surface, and CombinedBarLine</td>
<td>Color</td>
<td>Category</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pie</td>
<td>PieSectorSize</td>
<td>PieSectorColor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PieWidthDepth and Doughnut</td>
<td>PieSectorSize</td>
<td>PieDepthSize</td>
<td>PieSectorColor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scatter</td>
<td>Value1</td>
<td>Value2</td>
<td>Color</td>
<td>Shape</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bubble</td>
<td>Value1</td>
<td>Value2</td>
<td>BubbleWidth</td>
<td>BubbleHeight</td>
<td>Color</td>
<td>Shape</td>
</tr>
<tr>
<td>PolarScatter</td>
<td>AngularValue</td>
<td>RadialValue</td>
<td>Color</td>
<td>Shape</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PolarBubble</td>
<td>AngularValue</td>
<td>RadialValue</td>
<td>BubbleWidth</td>
<td>Color</td>
<td>Shape</td>
<td></td>
</tr>
<tr>
<td>BoxPlot</td>
<td>Category</td>
<td>Value</td>
<td>Color</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radar</td>
<td>Color</td>
<td>Category</td>
<td>Shape</td>
<td>Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HeatMap</td>
<td>MainCategory</td>
<td>Color</td>
<td>SecondCategory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TreeMap</td>
<td>RectangleTitle</td>
<td>RectangleWeight</td>
<td>Color</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TagCloud</td>
<td>Category</td>
<td>TagsWeight</td>
<td>TagsFamily</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Waterfall</td>
<td>Category</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dual</td>
<td>Category</td>
<td>Value1</td>
<td>Value2</td>
<td>Color</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note**

There can only be up to two dimensional axes in Radar charts. For example, if data is displayed on "Color" and "Category", then you cannot assign data to "Shape".
6.2.2.8.2 Labels

Table 99:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>visible</td>
<td>Boolean</td>
<td>Specifies whether a label is displayed on the axis</td>
</tr>
<tr>
<td>orientation</td>
<td>Auto</td>
<td>Angle30</td>
</tr>
<tr>
<td>deleteMode</td>
<td>Auto</td>
<td>Fixed</td>
</tr>
<tr>
<td>deleteFactor</td>
<td>integer</td>
<td>The deletion factor if deleteMode is Fixed. Minimum is 1.</td>
</tr>
<tr>
<td>autoResize</td>
<td>Boolean</td>
<td>Specifies whether font size is reduced automatically to display all labels</td>
</tr>
<tr>
<td>staggered</td>
<td>Boolean</td>
<td>Specifies whether labels are staggered</td>
</tr>
<tr>
<td>textPolicy</td>
<td>No Wrap</td>
<td>Wrap</td>
</tr>
</tbody>
</table>

Table 100:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;font&gt;</td>
<td>The label font properties (size, face, italic, bold, strikethrough, underline, and RGB color)</td>
</tr>
<tr>
<td>&lt;numberFormat&gt;</td>
<td>The format of the tick values defined through a template and a series of properties.</td>
</tr>
</tbody>
</table>

Example

```xml
<labels visible="true" orientation="Auto" staggered="false">
  <font size="8" face="Arial" italic="false" bold="true" strikethrough="false"
       underline="false" rgb="#707070"/>
  <numberFormat default="true" sample="1,234567E3" type="Custom">
    <template positive="SCIENTIFIC"/>
  </numberFormat>
</labels>
```
6.2.2.8.3 Scaling

```
<scaling>
  <minValue/>
  <maxValue/>
</scaling>
```

Table 101:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Linear</td>
<td>Exponential</td>
</tr>
<tr>
<td>originInRange</td>
<td>Always</td>
<td>Auto</td>
</tr>
<tr>
<td>unitScaleFactor</td>
<td>integer</td>
<td>The scaling axis unit scale exponent to be able to display very small or very large tick values in a small area. Value range is [-24, 24].</td>
</tr>
<tr>
<td>roundMinMaxValues</td>
<td>Boolean</td>
<td>Specifies whether limit values of the scaling axis are rounded to the nearest tick value</td>
</tr>
</tbody>
</table>

Table 102:

<table>
<thead>
<tr>
<th>Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;minValue&gt;</td>
<td>Defines the way the maximal value of the scaling axis is managed</td>
</tr>
<tr>
<td></td>
<td><strong>Attribute:</strong> type (Auto</td>
</tr>
<tr>
<td>&lt;maxValue&gt;</td>
<td>Defines the way the maximal value of the scaling axis is managed</td>
</tr>
<tr>
<td></td>
<td><strong>Attribute:</strong> type (Auto</td>
</tr>
</tbody>
</table>

**Example**

```
<scaling mode="Linear" unitScaleFactor="0" roundMinMaxValues="false">
  <minValue type="Auto"/>
  <maxValue type="Auto"/>
</scaling>
```

6.3 Date and Time Formats

The RESTful Web Service SDKs support `dateTime` data of the following format:

```
yyyy-mm-ddThh:mm:ss(.s+)(zzzzzz|Z)
```
Where:

Table 103:

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yyyy</td>
<td>A four digit that represents the year</td>
</tr>
<tr>
<td>-</td>
<td>Separators between parts of the date portion</td>
</tr>
<tr>
<td>First mm</td>
<td>A two-digit numeral that represents the month</td>
</tr>
<tr>
<td>dc</td>
<td>A two-digit numeral that represents the day</td>
</tr>
<tr>
<td>T</td>
<td>A separator indicating that time-of-day follows</td>
</tr>
<tr>
<td>hh</td>
<td>A two-digit numeral that represents the hour</td>
</tr>
<tr>
<td>:</td>
<td>A separator between parts of the time-of-day portion</td>
</tr>
<tr>
<td>Second mm</td>
<td>A two-digit numeral that represents the minute</td>
</tr>
<tr>
<td>ss</td>
<td>A two-integer-digit numeral that represents the whole seconds</td>
</tr>
<tr>
<td>.s+</td>
<td>The fractional seconds preceded by a dot separator</td>
</tr>
<tr>
<td>zzzzzz</td>
<td>zzzzzz represents the timezone according to the W3C recommendation. zzzzzz is of the form +</td>
</tr>
<tr>
<td>Z</td>
<td>The zero-length duration timezone, which is the UTC canonical representation</td>
</tr>
</tbody>
</table>

Example

2002-10-10T12:00:00.000-05:00 (noon on 10 October 2002, Central Daylight Savings Time as well as Eastern Standard Time in the U.S.) is equivalent to 2002-10-10T17:00:00.000Z.

Related Information

http://www.w3.org/TR/xmlschema-2/#dateTime

6.4 Object Full Paths

In XML request or response bodies such as a query specification or the description of a UNV or UNX universe, the objects of the universe are identified by the following attributes:

- **id**
  
  This is the object identifier as defined in the CMS repository.

- **path**
  
  This is the full path of the object. This attribute is optional and is mainly used by client tools to implement universe change source. If the universe data source must be replaced with another one containing different object IDs, then only the object paths can be used to map the objects in the query and the objects in the new universe data source.
You build the full paths by following the rules below:

- Each node in the path is made of the object name and type, separated by `|:<name>|<type>`.
- The escape character of `|` and `~` is `~`.
- The nodes of the path are concatenated with `\:"Age Group|folder\Age Max|dimension"`.
- The escape character of `\` and `§` is `§`.

The following table shows the possible types of objects manipulated in each of the provided SDKs:

<table>
<thead>
<tr>
<th>SDK</th>
<th>Object Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI Semantic Layer RESTful Web Service SDK</td>
<td>• attribute</td>
</tr>
<tr>
<td></td>
<td>• dimension</td>
</tr>
<tr>
<td></td>
<td>• filter</td>
</tr>
<tr>
<td></td>
<td>• folder</td>
</tr>
<tr>
<td></td>
<td>• measure</td>
</tr>
<tr>
<td>Web Intelligence RESTful Web Service SDK</td>
<td>• analysisDimension</td>
</tr>
<tr>
<td></td>
<td>• attribute</td>
</tr>
<tr>
<td></td>
<td>• aggregationAttribute</td>
</tr>
<tr>
<td></td>
<td>• calculatedMember</td>
</tr>
<tr>
<td></td>
<td>• dimension</td>
</tr>
<tr>
<td></td>
<td>• filter</td>
</tr>
<tr>
<td></td>
<td>• folder</td>
</tr>
<tr>
<td></td>
<td>• hierarchy</td>
</tr>
<tr>
<td></td>
<td>• level</td>
</tr>
<tr>
<td></td>
<td>• measure</td>
</tr>
<tr>
<td></td>
<td>• namedSet</td>
</tr>
</tbody>
</table>

**Example**

The following examples illustrate the naming rules:

<table>
<thead>
<tr>
<th>Path</th>
<th>Object</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root folder</td>
<td>&quot;CustomerName&quot; dimension</td>
<td>&quot;CustomerName\dimension&quot;</td>
</tr>
<tr>
<td>&quot;Customer&quot; folder</td>
<td>&quot;Name&quot; dimension</td>
<td>&quot;Customer\folder\Name\dimension&quot;</td>
</tr>
<tr>
<td>&quot;Customer&quot; dimension in &quot;Contact&quot; folder</td>
<td>&quot;Name&quot; attribute</td>
<td>&quot;Contact\folder\Customer\dimension\Name\attribute&quot;</td>
</tr>
<tr>
<td>&quot;Customer\Large&quot; dimension in &quot;Country\US&quot;</td>
<td>&quot;First~Name&quot; attribute</td>
<td>&quot;Country~\US\folder\Customer\Large\dimension\First~\Name\attribute&quot;</td>
</tr>
</tbody>
</table>
6.5 Parameters

The term "parameter" refers to the element defined in the request body manipulated by the web service. It represents contexts, @Prompts, and object parameters.

Table 106:

<table>
<thead>
<tr>
<th>SDK</th>
<th>Functionality</th>
</tr>
</thead>
</table>
| BI Semantic Layer RESTful Web Service SDK                | • Resolve contexts in universes  
|                                                          | • Answer @Prompts in universes  
|                                                          | • Answer object parameters defined in data foundations and business layers, and used as filters or in @Prompts. |
| Web Intelligence RESTful Web Service SDK                 | • Resolve contexts of queries based on universes  
|                                                          | • Identify and fill in prompts with values to refresh documents. Prompts may come from @Prompts in universes, object parameters defined in data foundations and business layers, SAP BW parameters or be defined in the query. |

Supported parameters and list of values include:

- Index-aware parameters
- Optional and non optional parameters
- Cascading parameters
- Index-aware lists of values
- Lists of values with multiple columns for UNX universes only
- Hierarchical lists of values for UNX universes only
- Object parameters

See the Information Design Tool User Guide for more information on contexts, @Prompts, and object parameters.

Parameter Response Body Schemas [page 95]
Lists of Values [page 98]
Default Values [page 101]
Previous Values [page 103]
Incompatible Contexts [page 103]
Answer Request Body Schemas [page 104]

6.5.1 Parameter Response Body Schemas

Response Body Schemas (XML)

(GET and PUT .../parameters)

This schema contains the parameters and their properties, such as the expected answers.
i Note

See examples in Getting the List of Parameters [page 159] (BI Semantic Layer RESTful Web Service SDK) and Getting the Refresh Parameters [page 464] (Web Intelligence RESTful Web Service SDK).

```xml
<parameters>
  <parameter type="context|prompt|sapVariable" optional="Boolean" dpId="string">
    <id></id>
    <technicalName></technicalName>
    <name></name>
    <answer type="Text|Numeric|Date|Unknown" constrained="Boolean">
      <info cardinality="Single|Multiple|Interval" keepLastValues="Boolean">
        LOV
        [DefaultValues|PreviousValues]
        ContextIncompatibilities
        Values
      </info>
    </answer>
  </parameter>
</parameters>
```

### Parameter

**Table 107:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;id&gt;</code></td>
<td>integer</td>
<td>Parameter identifier. The web service generates it automatically.</td>
</tr>
<tr>
<td><code>&lt;technicalName&gt;</code></td>
<td>string</td>
<td>Optional technical name of the parameter</td>
</tr>
<tr>
<td><code>&lt;name&gt;</code></td>
<td>string</td>
<td>Prompt question in the user locale</td>
</tr>
<tr>
<td><code>&lt;description&gt;</code></td>
<td>string</td>
<td>Optional description of the parameter</td>
</tr>
<tr>
<td><code>&lt;answer&gt;</code></td>
<td>N/A</td>
<td>The expected answer</td>
</tr>
</tbody>
</table>

**Table 108:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>context</td>
<td>prompt</td>
</tr>
<tr>
<td>optional</td>
<td>Boolean</td>
<td>Specifies whether the parameter is optional (false if type=&quot;context&quot;)</td>
</tr>
<tr>
<td>dpId</td>
<td>string</td>
<td>A mandatory attribute that defines the data provider identifier used for this parameter. Only used in the Web Intelligence RESTful Web Service SDK.</td>
</tr>
</tbody>
</table>
Answer

Table 109:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;info&gt;</td>
<td>Contains:</td>
</tr>
<tr>
<td></td>
<td>• The list of values associated with the parameter (LOV)</td>
</tr>
<tr>
<td></td>
<td>• The DefaultValues or PreviousValues</td>
</tr>
<tr>
<td></td>
<td>• The incompatible context values (ContextIncompatibilities)</td>
</tr>
<tr>
<td></td>
<td>See the next sections for more information.</td>
</tr>
</tbody>
</table>

| Values | The values associated with the parameter.  |
|        | In the GET .../parameters call result, they are either DefaultValues or PreviousValues.  |

➤ Remember

PreviousValues are only returned by the Web Intelligence RESTful Web Service SDK.

Table 110:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Text</td>
<td>Numeric</td>
</tr>
<tr>
<td>constrained</td>
<td>Boolean</td>
<td>Defines if the user can type a new value for the parameter (false) or if the user must select only the values from the associated list of values (true).</td>
</tr>
</tbody>
</table>

Info

Table 111:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cardinality</td>
<td>Single</td>
<td>Multiple</td>
</tr>
<tr>
<td>keepLastValues</td>
<td>Boolean</td>
<td>Indicates whether the parameter of type prompt keeps the previous answered values.</td>
</tr>
</tbody>
</table>

➤ Remember

When working with the BI Semantic Layer RESTful web service, you must implement the storage of the previous answered values, because the web service does not store them. When working with the Web Intelligence RESTful web service, the values are kept in the Web Intelligence document.
Related Information

Lists of Values [page 98]
Default Values [page 101]
Previous Values [page 103]
Incompatible Contexts [page 103]

6.5.2 Lists of Values

LOV represents the list of values associated with the parameter. It describes the possible values of the answer.

The <lov> element defines a list of values itself.

```
<lov hierarchical="Boolean"
    partial="Boolean"
    refreshable="Boolean"
    searchable="Boolean"
    mandatorySearch="Boolean"
    path="[0|1|2, 
              second_level_value],[0|1|2, 
              third_level_value],[...]]">
```

List of Values

Table 112:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hierarchical</td>
<td>Boolean</td>
<td>Specifies whether the list of values associated with the parameter is hier-</td>
</tr>
<tr>
<td>partial</td>
<td>Boolean</td>
<td>Specifies whether the entire list of values is displayed. The size of the</td>
</tr>
<tr>
<td>refreshable</td>
<td>Boolean</td>
<td>Specifies whether the list of values can be refreshed. This attribute can</td>
</tr>
<tr>
<td>searchable</td>
<td>Boolean</td>
<td>Specifies whether values of the list of values can be searched. This attrib-</td>
</tr>
<tr>
<td>mandatorySearch</td>
<td>Boolean</td>
<td>Specifies whether values of the list of values are restricted to those filtered</td>
</tr>
</tbody>
</table>

This is an optional attribute that you can use only when searchable="true". See Answer Request Body Schemas [page 104].
### Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>[0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0 represents a string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1 represents a date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2 represents a number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See example in Example - Responding to a Hierarchical Parameter [page 170].</td>
</tr>
</tbody>
</table>

### One Column

**LOV** is made of the following elements if the list of values contains one column:

```xml
<lov>
  <id>
  <updated>
  {Intervals | Values}
</lov>
```

**Note**

*Columns* does not appear if the parameter is of type context.

### Table 113:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;id&gt;</td>
<td>string</td>
<td>The list of values identifier</td>
</tr>
<tr>
<td>&lt;updated&gt;</td>
<td>DateTime</td>
<td>The date of last update of the list of values</td>
</tr>
<tr>
<td>Intervals</td>
<td>N/A</td>
<td>The element block that describes values as intervals when the number of values is too large. The first interval is returned by default. The default number of values in an interval is 50. Values of a context parameter can have a description.</td>
</tr>
</tbody>
</table>

```xml
<intervals>
  <interval id="integer">
    <value id="integer" description="string" final="Boolean">
```

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td>N/A</td>
<td>The element block that describes the possible values of the list of values. Values of a context parameter can have a description.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;values&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;value id=&quot;integer&quot; description=&quot;string&quot; final=&quot;Boolean&quot;&gt;</code></td>
</tr>
<tr>
<td>Columns</td>
<td>N/A</td>
<td>The element block that describes the column to be mapped to the list of values. <code>&lt;column&gt;</code> defines the column name. Its type attribute is mandatory.</td>
</tr>
</tbody>
</table>

### Multiple Columns

`LOV` is made of the following elements if the list of values contains multiple columns:

```
<lov>
  <id>
  <updated>
    [Intervals_MC | Values_MC]
  Columns
</lov>
```

**Note**

Columns does not appear if the parameter is of type context.

### Table 114:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;id&gt;</td>
<td>string</td>
<td>The list of values identifier</td>
</tr>
<tr>
<td>&lt;updated&gt;</td>
<td>DateTime</td>
<td>The date of last update of the list of values</td>
</tr>
<tr>
<td>Intervals_MC</td>
<td>N/A</td>
<td>The element block that describes values as intervals when the number of values is too large. The first interval is returned by default. An interval can contain 50 values.</td>
</tr>
<tr>
<td>Element</td>
<td>Type or Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Values_MC</td>
<td>N/A</td>
<td>The element block that describes the possible values of the list of values. <code>&lt;cvalue&gt;</code> defines the values of multiple columns. <code>&lt;column&gt;</code> defines the column value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;cvalues&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;cvalue id=&quot;integer&quot; final=&quot;Boolean&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;column id=&quot;integer&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;column id=&quot;integer&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;column id=&quot;integer&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;column id=&quot;integer&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;column id=&quot;integer&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;column id=&quot;integer&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;column id=&quot;integer&quot;</code></td>
</tr>
</tbody>
</table>

**Cascading Parameters**

In the case of cascading parameters, the value given to a parameter depends on the answer to a previous parameter.

**LOV** is made of the following elements:

```xml
<lov>
  <id>
    <parameters>
      <id>
      <id>
      ...
</parameters>
  </id>
</lov>
```

Under `<parameters>`, each `<id>` corresponds to the identifier of a parameter, on which depend the values of the current parameter.

### 6.5.3 Default Values

**DefaultValues** represents the default values that the parameter can accept.

**Note**

If the parameter type is context, then there is no default value, and **DefaultValues** is not required.
One Column

DefaultValues is structured as follows if the parameter accepts values of only one column:

```xml
<values>
  <value id="string" path="[[0|1|2, second_level_value],[0|1|2, third_level_value], [...]]">
    ...
  </value>
</values>
```

The following table describes the `<value>` attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The value identifier. It is used if the parameter is of context type or if the values are indexed either from the SAP system or from index awareness.</td>
</tr>
</tbody>
</table>
| path      | [0|1|2, value] | Optional. In the case of hierarchical parameters, specifies the values and types of the hierarchy, starting with the second level. The syntax [0|1|2, value] describes the data type and the data value of a level.  
  - 0 represents a string  
  - 1 represents a date  
  - 2 represents a number |

Multiple Columns

DefaultValues is structured as follows if the parameter accepts values of multiple columns:

```xml
<cvalues>
  <cvalue id="string" final="Boolean">
    <column id="integer" type="String|Date|Numeric">
      ...
    </column>
  </cvalue>
  ...
</cvalues>
```

<cvalue> defines the values of multiple columns. The following table describes its attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The value identifier. It is used if the values are indexed either from the SAP system or from index awareness.</td>
</tr>
<tr>
<td>final</td>
<td>Boolean</td>
<td>Attribute specific to hierarchical parameters that defines if the parameter is used to answer a value for a node (final=&quot;false&quot;) or a leaf (final=&quot;true&quot;) in the hierarchy</td>
</tr>
</tbody>
</table>
**6.5.4 Previous Values**

*PreviousValues* represents the previous values that the parameter has accepted.

```
<i>Note</i>
They are only returned by the Web Intelligence RESTful Web Service SDK.
```

*PreviousValues* is structured as default values:

```
<previous>
  <value id="string" path="[[0|1|2, second_level_value], [0|1|2, third_level_value]], ...
```

The GET .../parameters request adheres to the following rules regarding the management of previous and default values:

- Previous values are returned inside `<answer>/values` when they exist.
- If there are no previous values but default values, then default values are returned inside `<answer>/values`.
- If there are no previous nor default values, then nothing is returned.

**6.5.5 Incompatible Contexts**

*ContextIncompatibilities* represents the contexts that are incompatible.

*ContextIncompatibilities* is structured as follows:

```
<incompatibility>
  <values>
    <value id="string">
    <value id="string">
    ...
```

Context values are grouped by pair.
Table 118:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;value&gt;</td>
<td>string</td>
<td>The context value</td>
</tr>
</tbody>
</table>

The following table describes the <value> attribute:

Table 119:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The context value identifier</td>
</tr>
</tbody>
</table>

6.5.6 Answer Request Body Schemas

Request Body Schemas (XML)

(PUT .../parameters)

This schema contains the actual answers to pass to contexts or prompts, and the query of a list of values for the parameters that remain to be answered.

```xml
<parameters>
  <parameter>
    <answer type="Text|Numeric|Date|Unknown" constrained="Boolean">
      <info>
        <values>
          <value path="[0|1|2,\ value]" id="integer">
            <info>
              <lov>
                <query intervalId="integer" intervalSize="Integer|-1|Unlimited|Server" refresh="Boolean">
                  <sort order="Boolean"/>
                  <search>
                    <path> <value id="integer" type="String|Date|Numeric">
```

Parameter

Table 120:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;id&gt;</td>
<td>integer</td>
<td>Parameter identifier</td>
</tr>
<tr>
<td>&lt;answer&gt;</td>
<td>N/A</td>
<td>The actual answer</td>
</tr>
</tbody>
</table>
### Answer

Table 121:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;info&gt;</td>
<td>Contains one of the following:</td>
</tr>
<tr>
<td></td>
<td>● The possible values of the list of values. See its description in Lists of Values [page 98].</td>
</tr>
<tr>
<td></td>
<td>● The query used to retrieve a list of values for unanswered parameters. The query can specify how values will be returned, and/or refreshed if the list of values allows it.</td>
</tr>
</tbody>
</table>

➤ Remember

Not mandatory in the request bodies.

| <values> | The list of actual values expressed as a sequence of <value> elements. Is ignored by the service in the request bodies. |

### Values

Table 122:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;value&gt;</td>
<td>string</td>
<td>An actual value</td>
</tr>
</tbody>
</table>

**Attributes:**

- `id`: Defines the context identifier or the value index. Mandatory if the value contains an answer to a context or if the prompt is index-aware.
- `path`: Optional. In the case of hierarchical parameters, specifies the values and types of the hierarchy, starting with the second level. The call needs this information to reach the answer value.

The syntax `[0|1|2, \second_level_value],[0|1|2, \third_level_value],[...]]` describes the data type and the data value of a level.

- `0` represents a string
- `1` represents a date
- `2` represents a number

You can also use `String`, `Date`, or `Numeric` explicitly instead of numbers.

See examples in Example - Responding to a Hierarchical Parameter [page 170] and Example - Refreshing a Document with a Hierarchical Parameter [page 474].
Table 123:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intervalId</td>
<td>integer</td>
<td>The index of the interval that should be returned. If not specified, the first interval is returned. An error is returned if this index is out of range (depending on values count).</td>
</tr>
</tbody>
</table>
| intervalSize | integer|-1|Unlimited|Server | Specifies the number of values in the interval to return. If not specified, 50 is used. Possible values are:  
- A strictly positive integer  
- -1 or Unlimited indicates the whole list of values is returned  
- Server indicates that the Information Engine Service property List of Values Batch size (entries) of the CMC defines the number of values to return. |
| refresh      | Boolean               | Specifies whether the list of values is refreshed. Optional. An error is returned if the list of values does not allow refreshing. |

Table 124:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;sort&gt;</td>
<td>Boolean</td>
<td>Specifies whether the list of values must be returned sorted. Optional.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attribute: order (Ascending</td>
</tr>
<tr>
<td>&lt;search&gt;</td>
<td>string</td>
<td>Optional. Defines a search pattern in the list of values as a string. The following wildcard characters may be used in the pattern string: ? for zero or one character, and * for zero or n characters. For example, M?Gregor yields to the value McGregor, and M*Gregor to the values McGregor and MacGregor.</td>
</tr>
<tr>
<td>&lt;path&gt;</td>
<td>N/A</td>
<td>Optional. In the case of a hierarchical parameter, it specifies the intermediate answer value. It also specifies the node for which the list of values is requested. This path is expressed as a sequence of &lt;value&gt; elements. See examples in Example - Responding to a Hierarchical Parameter [page 170] and Example - Refreshing a Document with a Hierarchical Parameter [page 474].</td>
</tr>
</tbody>
</table>
Query Path

Table 125:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| <value>  | string        | The value for a node of the path. Attributes:  
- id: defines the context identifier or the value index. Mandatory if the value contains an answer to a context or if the prompt is index-aware.  
- type: The value type (String|Date|Numeric). Default is String. |

6.6 Queries

A query specification is an XML document that describes thoroughly the query to be run using the BI Semantic Layer RESTful Web Service SDK. The query specification is based on a query model designed to standardize the data extraction from data providers by products that use the concepts of the Semantic Layer.

This query model allows you to define any type of SQL query to extract and return data from any data source, such as relational databases, OLAP data providers, XML files or web services.

A query specification supports the following components of a query:

- Query options
- Result objects
- Sort objects
- Query filters

The query model also allows you to define advanced queries such as combined queries.

- Query Specification Body Schema [page 108]
- Query Options [page 109]
- Result Objects [page 110]
- Sort Objects [page 111]
- Query Filters [page 112]
- Combined Queries [page 125]
- Query Workflows [page 126]
- The OData Query Service [page 128]
6.6.1 Query Specification Body Schema

Body Schema (XML)

(POST /queries and GET /queries/<queryID>)

```
<query xmlns="http://www.sap.com/rws/sl/universe" id="string" dataSourceType="unv\nunx" dataSourceId="integer">
  <querySpecification version="1.0">
    <queryOptions>
      <queryOption>
        <queryData>
          <resultObjects>
            <resultObject>
              <sortObjects>
                <sortObject>
                  <filterPart>
                    QueryFilters
                  </filterPart>
                </sortObject>
              </sortObjects>
            </resultObject>
          </resultObjects>
        </queryData>
      </queryOption>
    </queryOptions>
  </querySpecification>
</query>
```

QueryFilters stands for the filter conditions contained by the query. See the next sections for a detailed description of the query specification elements.

Query

Table 126:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;querySpecification&gt;</td>
<td>The query specification</td>
</tr>
<tr>
<td></td>
<td>Attribute: version(1.0)</td>
</tr>
<tr>
<td>&lt;queryOptions&gt;</td>
<td>The list of query options</td>
</tr>
<tr>
<td>&lt;queryData&gt;</td>
<td>The objects used in the query as result, sort and filter objects. There is at least one &lt;queryData&gt;</td>
</tr>
</tbody>
</table>

Table 127:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The query identifier</td>
</tr>
<tr>
<td>dataSourceType</td>
<td>unv</td>
<td>unx</td>
</tr>
<tr>
<td>dataSourceId</td>
<td>integer</td>
<td>The universe identifier</td>
</tr>
</tbody>
</table>

Related Information

Query Filters [page 112]
Creating a Query [page 155]
6.6.2 Query Options

You can specify in the query some of the options that you can set in the universe design tool for UNV universes or the information design tool for UNX universes.

```xml
<queryOptions>
    <queryOption name="string" activated="Boolean" value="string"/>
</queryOptions>
```

➤ Remember
An option that is not present explicitly in the specification is not validated.

Table 128:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string (values in table below)</td>
<td>The option name</td>
</tr>
<tr>
<td>activated</td>
<td>Boolean</td>
<td>The option does not apply if activated is false. Is optional. If not present, the option is activated.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>The option value</td>
</tr>
</tbody>
</table>

Table 129:

<table>
<thead>
<tr>
<th>name Attribute Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duplicatedRows</td>
<td>If true, the query returns all related rows, even if there are duplicated rows.</td>
</tr>
<tr>
<td>maxRetrievalTime</td>
<td>It defines the maximum time that a query can run before the query is stopped.</td>
</tr>
<tr>
<td>InSeconds</td>
<td></td>
</tr>
<tr>
<td>maxRowsRetrieved</td>
<td>If true, the query returns all the possible rows, but only displays the first n rows, where n is the maximum number of rows set for this option.</td>
</tr>
<tr>
<td></td>
<td>If the user only needs a certain amount of data, you can set this option to restrict the number of rows of data displayed in reports.</td>
</tr>
<tr>
<td>samplingResultSetSize</td>
<td>It defines the maximum number of rows that a query returns as a sample.</td>
</tr>
<tr>
<td>samplingResultSetFixed</td>
<td>It defines the type of sampling used. Option values are true for fixed sampling and false for random.</td>
</tr>
</tbody>
</table>

➤ Remember

samplingResultSetSize can be set without samplingResultSetFixed. In that case, the type of sampling is random.

Example

The following snippet of a query specification for a UNIX universe shows the query options.

```xml
<query xmlns="http://www.sap.com/rws/sl/universe" id="589789982204141561" dataSourceType="unx" dataSourceId="5808">  
    <querySpecification version="1.0">  
        <queryOptions>
```

SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer
Concepts and Schemas of the RESTful Web Service SDKs © 2015 SAP SE or an SAP affiliate company. All rights reserved.
6.6.3 Result Objects

The result objects compose the query.

Table 130:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The object identifier</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>The full path of the object in the universe. Optional.</td>
</tr>
</tbody>
</table>

➤ Remember

The object order in <resultObjects> is important, because it reflects the object order in the SQL query. For example, the following query:

```xml
<queryData>
  <resultObjects>
    <resultObject id="A" .../>
    <resultObject id="B" .../>
  </resultObjects>
  ...
</queryData>
```

gives different results from the following one:

```xml
<queryData>
  <resultObjects>
    <resultObject id="B" .../>
    <resultObject id="A" .../>
  </resultObjects>
  ...
</queryData>
```

Example

The following snippet of a query specification shows the result objects of the query for a UNX universe.

```xml
<query xmlns="http://www.sap.com/rws/s1/universe" id="5897899822041415615" dataSourceType="unx" dataSourceId="5808">
  <querySpecification version="1.0">
    ...
  </querySpecification>
</query>
```
6.6.4 Sort Objects

The sort objects are used to sort the query result. They can be dimensions, attributes or measures. Result objects can be used as sort objects. This mainly depends on the query capabilities of the universe.

```
   <sortObjects>
       <sortObject id="string" path="string" sortType="Ascending\|Descending"/>
   </sortObjects>
```

Table 131:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The object identifier, defined when retrieving the data source metadata</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>The full path of the object in the universe. Optional.</td>
</tr>
<tr>
<td>sortType</td>
<td>Ascending|Descending</td>
<td>The sort type</td>
</tr>
</tbody>
</table>

**Example**

The following snippet of a query specification shows the sort objects used in the query for a UNX universe. Two of them are also defined as result objects.

```
<query xmlns="http://www.sap.com/rws/sl/universe" id="9168123992538053733"
   dataSourceType="unx" dataSourceId="5808">
   <querySpecification version="1.0">
       ...
   </querySpecification>
   ...
   <resultObjects>
       <resultObject path="Product\folder\Category\dimension"
          id="IB8eG7hEeCk0Y1v-tlf2Q"/>
       <resultObject path="Product\folder\Product\dimension"
          id="IB8eGb1hEeCk0Y1v-tlf2Q"/>
       <resultObject path="Time\folder\Calendar\folder\Calendar Year\Month\dimension"
          id="IB08G7hEeCk0Y1v-tlf2Q"/>
       <resultObject path="Inventory\folder\Stock Level\measure"
          id="IB8eFr1hEeCk0Y1v-tlf2Q"/>
       <resultObject path="Product\folder\Minimum Stock\measure"
          id="IB8eHr1hEeCk0Y1v-tlf2Q"/>
   </resultObjects>
</query>
```
6.6.5 Query Filters

The query filters define the filtering conditions for query results.

The query specification model defines the following types of filters:

- Predefined filters
- Custom filters

The following custom filters are supported:

- Comparison filters based on constant values
- Comparison filters based on objects
- Ranking filters
- Subquery filters
- Combined filters, which are made of multiple comparison filters

Predefined Filters [page 112]

Custom Filters - Constant Comparison Filters [page 113]

Custom Filters - Object Comparison Filters [page 117]

Custom Filters - Ranking Filters [page 118]

Custom Subquery Filters [page 122]

Combined Custom Filters [page 124]

6.6.5.1 Predefined Filters

```xml
<filterPart>
  <predefinedFilter id="string" path="string"/>
</querySpecification>
```
Table 132:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The object identifier, as defined when retrieving the data source metadata</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>The full path of the object in the universe</td>
</tr>
</tbody>
</table>

Example

The following snippet of a query specification shows a predefined filter in the query.

```xml
<query xmlns="http://www.sap.com/rws/sl/universe" id="589789982041415615" dataSourceType="unx" dataSourceId="5808">
  <querySpecification version="1.0">
    <queryOptions>
      ...          
    </queryOptions>
    <queryData>
      <resultObjects>
        <resultObject path="Product\folder\Product\dimension" id="_IB8eGb1hEeCk0Ylv-t1P2Q"/>
        <resultObject path="Inventory\folder\Stock Level\measure" id="_IB8eFr1hEeCk0Ylv-t1P2Q"/>
        <resultObject path="Product\folder\Minimum Stock\measure" id="_IB8eHr1hEeCk0Ylv-t1P2Q"/>
      </resultObjects>
      <filterPart>
        <predefinedFilter path="Inventory\folder\Stock Below Minimum|filter" id="_V5_GkLR_EeCZotjyucy1A"/>
      </filterPart>
    </queryData>
  </querySpecification>
</query>
```

6.6.5.2 Custom Filters - Constant Comparison Filters

For each `<comparisonFilter>`, you can add up to two `<constantOperand>`. Each operand contains an `<answerValue>` that represents the constant value used as right operand.

Comparison Filter

Table 133:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The identifier of the object used as left operand</td>
</tr>
</tbody>
</table>
### Attribute Types or Value Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>string</td>
<td>The full path of the object in the universe</td>
</tr>
<tr>
<td>operator</td>
<td>string (values in table below)</td>
<td>The operator</td>
</tr>
</tbody>
</table>

### Table 134: Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>operator Attribute Value</th>
<th>Number of Right Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is Null</td>
<td>IsNull</td>
<td>Zero</td>
</tr>
<tr>
<td>Is Not Null</td>
<td>IsNotNull</td>
<td>Zero</td>
</tr>
<tr>
<td>Equal To (=)</td>
<td>EqualTo</td>
<td>One</td>
</tr>
<tr>
<td>Not Equal To (&lt;&gt; )</td>
<td>NotEqualTo</td>
<td>One</td>
</tr>
<tr>
<td>Less Than (&lt;)</td>
<td>LessThan</td>
<td>One</td>
</tr>
<tr>
<td>Greater Than (&gt;)</td>
<td>GreaterThan</td>
<td>One</td>
</tr>
<tr>
<td>Less Than or Equal To (&lt;=)</td>
<td>LessThanOrEqualTo</td>
<td>One</td>
</tr>
<tr>
<td>Greater Than or Equal to (&gt;=)</td>
<td>GreaterThanOrEqualTo</td>
<td>One</td>
</tr>
<tr>
<td>Like</td>
<td>Like</td>
<td>One</td>
</tr>
<tr>
<td>Not Like</td>
<td>NotLike</td>
<td>One</td>
</tr>
<tr>
<td>Except</td>
<td>Except</td>
<td>One</td>
</tr>
<tr>
<td>Between</td>
<td>Between</td>
<td>Two</td>
</tr>
<tr>
<td>Not Between</td>
<td>NotBetween</td>
<td>Two</td>
</tr>
<tr>
<td>Both</td>
<td>Both</td>
<td>Two</td>
</tr>
<tr>
<td>In</td>
<td>InList</td>
<td>List</td>
</tr>
<tr>
<td>Not In</td>
<td>NotInList</td>
<td>List</td>
</tr>
</tbody>
</table>
Constant Operand

Table 135:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchPattern</td>
<td>Boolean</td>
<td>Specifies how the &lt;answerValue&gt; value is parsed without knowing the escape and mask characters of the underlying database. Can be used with the Like operator only. If set to true, the &lt;answerValue&gt; value is parsed: ● The * character is replaced with the multicharacter mask of the database in the resulting SQL. ● The ? character is replaced with the single-character mask of the database in the resulting SQL. ● If the pattern contains ? or \*, then ? or * are escaped. Strings containing ? or * are actually found by the operator. ● The _ and % wildcard characters are escaped. If set to false, the value is not parsed. Default is false.</td>
</tr>
</tbody>
</table>

Answer Value

Table 136:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataType</td>
<td>String</td>
<td>Numeric</td>
</tr>
</tbody>
</table>

Example

IsNull Operator (No Constant Operand)

The following snippet of a query specification shows an IsNull comparison filter with no constant operand in the query.

```xml
<query dataSourceId="5909" dataSourceType="unx" xmlns="http://www.sap.com/rws/sl/universe">
  <querySpec version="1.0">
    <queryData>
      <resultObjects>
        <resultObject id="_IB8eVbJhEeCk0Ylv-tJF2Q" path="Customer|folder\Geography|folder\Continent|dimension"/>
        <resultObject id="_IB8eWbJhEeCk0Ylv-tJF2Q" path="Customer|folder\Geography|folder\Country|dimension"/>
      </resultObjects>
      <filterPart>
        <comparisonFilter id="_IB8eVbJhEeCk0Ylv-tJF2Q" path="Sales Orders|folder\Number of Orders|measure" operator="IsNull"/>
      </filterPart>
    </queryData>
  </querySpec>
</query>
```
Example

Equal To Operator (One Constant Operand)

The following snippet of a query specification shows an *EqualTo* comparison filter with one constant operand in the query.

```xml
<query xmlns="http://www.sap.com/rws/sl/universe" id="5897899822041415615"
   dataSourceType="unx" dataSourceId="5808">
   <querySpecification version="1.0">
       <queryOptions>
       </queryOptions>
       <queryData>
           <resultObjects>
               <resultObject path="Customer|folder\Geography|folder\Continent|dimension" id="_IBo8M7IhEeCk0Ylv-t1F2Q"/>
               <resultObject path="Customer|folder\Geography|folder\Country|dimension" id="_IB8eVbIhEeCk0Ylv-t1F2Q"/>
               <resultObject path="SalesOrders|folder\Number of Orders\measure" id="_IB8eVbIhEeCk0Ylv-t1F2Q"/>
           </resultObjects>
           <filterPart>
               <comparisonFilter operator="EqualTo" path="Time|folder\Calendar|folder\CalendarYear|dimension" id="_IBo8FLIhEeCk0Ylv-t1F2Q">
                   <constantOperand searchPattern="false">
                       <answerValue dataType="String">2011</answerValue>
                   </constantOperand>
               </comparisonFilter>
           </filterPart>
       </queryData>
   </querySpecification>
</query>
```

Example

Between Operator (Two Constant Operands)

The following snippet of a query specification shows a *Between* comparison filter with two constant operands in the query.

```xml
<query dataSourceId="5909" dataSourceType="unx" xmlns="http://www.sap.com/rws/sl/universe">
   <querySpecification version="1.0">
       <queryData>
           <resultObjects>
               <resultObject id="_IBo8M7IhEeCk0Ylv-t1F2Q" path="Customer|folder\Geography|folder\Continent|dimension"/>
               <resultObject id="_IB8eVbIhEeCk0Ylv-t1F2Q" path="Customer|folder\Geography|folder\Country|dimension"/>
               <resultObject id="_IB8eVbIhEeCk0Ylv-t1F2Q" path="Sales Orders|folder\Number of Orders\measure"/>
           </resultObjects>
           <filterPart>
               <comparisonFilter id="_IB8eVbIhEeCk0Ylv-t1F2Q" path="Sales Orders\folder\Number of Orders\measure" operator="Between">
                   <constantOperand searchPattern="false">
                       <answerValue dataType="Numeric">200</answerValue>
                   </constantOperand>
                   <constantOperand searchPattern="false">
                       <answerValue dataType="Numeric">700</answerValue>
                   </constantOperand>
               </comparisonFilter>
           </filterPart>
       </queryData>
   </querySpecification>
</query>
```
6.6.5.3 Custom Filters - Object Comparison Filters

<filterPart>
  <comparisonFilter id="string" path="string" operator="string">
    <objectOperand id="string" path="string">
      <objectOperand>
        represents the business object used as right operand.
      </objectOperand>
    </objectOperand>
  </comparisonFilter>
</filterPart>

Comparison Filter

Table 137:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The identifier of the object used as left operand, as defined when retrieving the data source metadata</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>The full path of the object in the universe</td>
</tr>
<tr>
<td>operator</td>
<td>string (values in table below)</td>
<td>The operator</td>
</tr>
</tbody>
</table>

Table 138:

<table>
<thead>
<tr>
<th>Operator</th>
<th>operator Attribute Value</th>
<th>Number of Right Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal To (=)</td>
<td>EqualTo</td>
<td>One object</td>
</tr>
<tr>
<td>Not Equal To (&lt;&gt;</td>
<td>NotEqualTo</td>
<td>One object</td>
</tr>
<tr>
<td>Less Than (&lt;)</td>
<td>LessThan</td>
<td>One object</td>
</tr>
<tr>
<td>Greater Than (&gt;)</td>
<td>GreaterThan</td>
<td>One object</td>
</tr>
<tr>
<td>Less Than or Equal To</td>
<td>LessThanOrEqualTo</td>
<td>One object</td>
</tr>
<tr>
<td>Greater Than or Equal to (&gt;=)</td>
<td>GreaterThanOrEqualTo</td>
<td>One object</td>
</tr>
</tbody>
</table>
### Object Operand

Table 139:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The identifier of the object used as right operand</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>The full path of the object in the universe</td>
</tr>
</tbody>
</table>

#### Example

The following snippet of a query specification shows an `EqualTo` comparison filter with one object as operand in the query.

```xml
<query xmlns="http://www.sap.com/rws/sl/universe" id="5897899822041415615"
  dataSourceType="unx" dataSourceId="5808">
  <querySpecification version="1.0">
    <queryOptions>...
    </queryOptions>
    <queryData>
      <resultObjects>
        <resultObject path="Customer|folder\Geography|folder\Continent|dimension" id="_IBo8W7hEeCk0Ylv-t1F2Q"/>
        <resultObject path="Customer|folder\Geography|folder\Country|dimension" id="_IBo8NrlHeeCk0Ylv-t1F2Q"/>
        <resultObject path="SalesOrders|folder\Number of Orders|measure" id="_IB8eVb1hEeCk0Ylv-t1F2Q"/>
      </resultObjects>
      <filterPart>
        <comparisonFilter path="Customer|folder\Geography|folder\Country|dimension" operator="EqualTo"
          id="_IBo8NrlHeeCk0Ylv-t1F2Q">
          <objectOperand id="_IBo8JzLlEeCk0Ylv-t1F2Q" path="Restrictions\Restricted Country"/>
        </comparisonFilter>
      </filterPart>
    </queryData>
  </querySpecification>
</query>
```

#### 6.6.5.4 Custom Filters - Ranking Filters

The ranking filter is defined by:
- A ranking order (ascending or descending)
• The question prompted to the end-user if the filter is triggered by a prompt
• The dimension and measure to rank
• The additional dimensions used to rank
• Any additional custom filter it may contain

Ranking Filter

Table 140:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>Top</td>
<td>Bottom</td>
</tr>
<tr>
<td>level</td>
<td>integer</td>
<td>The number of values to rank. Optional. In the case of a filter with prompt, it is the default value of the prompt. This behavior is specific to UNV universes.</td>
</tr>
</tbody>
</table>

Prompt

Table 141:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;prompt&gt;</td>
<td>string</td>
<td>Specifies the prompt question in the case of a filter with prompt</td>
</tr>
</tbody>
</table>

Dimension

Table 142:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The identifier of the dimension object to rank</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>Optional. The full path of the dimension object</td>
</tr>
</tbody>
</table>

Measure

Table 143:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The identifier of the measure object to rank</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type or Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>Optional. The full path of the measure object</td>
</tr>
</tbody>
</table>

**Ranked by Dimensions**

`<rankedByDimensions>` is optional. You add at least one `<rankedByDimension>` to `<rankedByDimensions>`.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The identifier of the additional dimension object used to rank</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>Optional. The full path of the dimension object</td>
</tr>
</tbody>
</table>

**Example**

**Without Prompt**

```xml
<query dataSourceType="unx" dataSourceId="91800" xmlns="http://www.sap.com/rws/sl/universe">
  <querySpecification version="1.0">
    <queryData>
      <resultObjects>
        <resultObject path="Dimcustomer|folder\Regionname|dimension" id="_60xHwQ-8Ee01RP--CtxScg"/>
        <resultObject path="Dimcustomer|folder\Countryname|dimension" id="_60xHwA-8Ee01RP--CtxScg"/>
        <resultObject path="Dimcustomer|folder\Customer Name|dimension" id="_72kd8A-8Ee01RP--CtxScg"/>
        <resultObject path="Custorderline|folder\Quantity|measure" id="_60Bg4g-8Ee01RP--CtxScg"/>
      </resultObjects>
      <filterPart>
        <rankingFilter level="3" function="Top">
          <dimension path="Dimcustomer|folder\Customer Name|dimension" id="_72kd8A-8Ee01RP--CtxScg"/>
          <basedOnMeasure path="Custorderline|folder\Quantity|measure" id="_60Bg4g-8Ee01RP--CtxScg"/>
        </rankingFilter>
      </filterPart>
    </queryData>
  </querySpecification>
</query>
```

**Example**

**With a Prompt**

```xml
<query dataSourceType="unx" dataSourceId="91800" xmlns="http://www.sap.com/rws/sl/universe">
  <querySpecification version="1.0">
  </querySpecification>
</query>
```
Example

With Additional Dimensions

```xml
<query dataSourceType="unx" dataSourceId="91800" xmlns="http://www.sap.com/rws/sl/universe">
  <querySpecification version="1.0">
    <queryData>
      <resultObjects>
        <resultObject path="Dimcustomer|folder\Regionname|dimension" id="_60xHwQ-8EeO1RP--CtxScg"/>
        <resultObject path="Dimcustomer|folder\Countryname|dimension" id="_60xHwA-8EeO1RP--CtxScg"/>
        <resultObject path="Dimcustomer|folder\Customer Name|dimension" id="_7Zkd8A-8EeO1RP--CtxScg"/>
        <resultObject path="Custorderline|folder\Quantity|measure" id="_60Bg4g-8EeO1RP--CtxScg"/>
      </resultObjects>
      <filterPart>
        <rankingFilter level="3" function="Top">
          <prompt>Enter the ranking level:</prompt>
          <dimension path="Dimcustomer|folder\Customer Name|dimension" id="_7Zkd8A-8EeO1RP--CtxScg"/>
          <basedOnMeasure path="Custorderline|folder\Quantity|measure" id="_60Bg4g-8EeO1RP--CtxScg"/>
        </rankingFilter>
      </filterPart>
    </queryData>
  </querySpecification>
</query>
```

Example

With a Predefined Filter

```xml
<query dataSourceType="unx" dataSourceId="91800" xmlns="http://www.sap.com/rws/sl/universe">
  <querySpecification version="1.0">
    <queryData>
      <resultObjects>
        <resultObject path="Dimcustomer|folder\Regionname|dimension" id="_60xHwQ-8EeO1RP--CtxScg"/>
        <resultObject path="Dimcustomer|folder\Countryname|dimension" id="_60xHwA-8EeO1RP--CtxScg"/>
        <resultObject path="Dimcustomer|folder\Customer Name|dimension" id="_7Zkd8A-8EeO1RP--CtxScg"/>
        <resultObject path="Custorderline|folder\Quantity|measure" id="_60Bg4g-8EeO1RP--CtxScg"/>
        <rankedByDimensions>
          <rankedByDimension path="Dimcustomer|folder\Regionname|dimension" id="_60xHwQ-8EeO1RP--CtxScg"/>
          <rankedByDimension path="Dimcustomer|folder\Countryname|dimension" id="_60xHwA-8EeO1RP--CtxScg"/>
        </rankedByDimensions>
      </resultObjects>
      <filterPart>
        <rankingFilter level="3" function="Top">
          <prompt>Enter ranking level :</prompt>
          <dimension path="Dimcustomer|folder\Customer Name|dimension" id="_7Zkd8A-8EeO1RP--CtxScg"/>
          <basedOnMeasure path="Custorderline|folder\Quantity|measure" id="_60Bg4g-8EeO1RP--CtxScg"/>
        </rankingFilter>
      </filterPart>
    </queryData>
  </querySpecification>
</query>
```
6.6.5.5 Custom Subquery Filters

See the *SAP BusinessObjects Web Intelligence User’s Guide* to learn the definition and benefits of subqueries.

### Subquery Filter

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator</td>
<td>string</td>
<td>The operator</td>
</tr>
<tr>
<td>correctionType</td>
<td>None</td>
<td>Any</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operator</th>
<th>operator Attribute Value</th>
<th>Number of Operands</th>
<th>Correlation Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal To (=)</td>
<td>EqualTo</td>
<td>One object</td>
<td>No</td>
</tr>
<tr>
<td>Operator</td>
<td>operator Attribute Value</td>
<td>Number of Operands</td>
<td>Correlation Supported</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------</td>
<td>--------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Not Equal To (&lt;&gt;)</td>
<td>NotEqualTo</td>
<td>One object</td>
<td>Yes</td>
</tr>
<tr>
<td>Less Than (&lt;)</td>
<td>LessThan</td>
<td>One object</td>
<td>Yes</td>
</tr>
<tr>
<td>Greater Than (&gt;)</td>
<td>GreaterThan</td>
<td>One object</td>
<td>Yes</td>
</tr>
<tr>
<td>Less Than or Equal To (&lt;=)</td>
<td>LessThanOrEqualTo</td>
<td>One object</td>
<td>Yes</td>
</tr>
<tr>
<td>Greater Than or Equal to (&gt;=)</td>
<td>GreaterThanOrEqualTo</td>
<td>One object</td>
<td>Yes</td>
</tr>
<tr>
<td>In</td>
<td>InList</td>
<td>List</td>
<td>No</td>
</tr>
<tr>
<td>Not In</td>
<td>NotInList</td>
<td>List</td>
<td>No</td>
</tr>
</tbody>
</table>

**Filter Object**

**Table 147:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The object identifier used as filter</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>The full path of the object in the universe</td>
</tr>
</tbody>
</table>

**Example**

The following snippet of a query specification shows a subquery filter with no correlation.

```xml
<query xmlns="http://www.sap.com/rws/sl/universe" id="5897899822041415615"
  dataSourceType="unx" dataSourceId="5808">
  <querySpecification version="1.0">
    <queryOptions>
      ...
    </queryOptions>
    <queryData>
      <resultObjects>
        <resultObject path="Time|folder\Calendar|folder\Calendar Year|dimension" id="_IBo8FLhEeCk0Ylv-t1F2Q"/>
        <resultObject path="Customer|folder\Geography|folder\Continent|dimension" id="_IBo8M7hEeCk0Ylv-t1F2Q"/>
        <resultObject path="Customer|folder\Geography|folder\Country|dimension" id="_IBo8NrhEeCk0Ylv-t1F2Q"/>
        <resultObject path="Sales Revenue|folder\Net Sales|measure id="_IB8eRLhEeCk0Ylv-t1F2Q"/>
      </resultObjects>
    </queryData>
    <filterPart>
      <subQueryFilter operator="EqualTo" correlationType="Any">
        <filterObjects>
          \Continent|dimension" id="_IBo8M7hEeCk0Ylv-t1F2Q"/>
          <filterObject path="Customer|folder\Geography|folder\Country|dimension" id="_IBo8NrhEeCk0Ylv-t1F2Q"/>
        </filterObjects>
      </subQueryFilter>
    </filterPart>
  </querySpecification>
</query>
```
6.6.5.6 Combined Custom Filters

In the query specification model, comparison filters can be combined using the AND and OR logical operators. These operators are defined using the `<and>` and `<or>` XML elements.

**Example**

The following snippet of a query specification shows the combination AND of two comparison filters.

```xml
<query xmlns="http://www.sap.com/rws/sl/universe" id="5897899822041415615"
   dataSourceType="unx" dataSourceId="5508">
   <querySpecification version="1.0">
      <queryOptions>
         ...
      </queryOptions>
      <queryData>
         <resultObjects>
            <resultObject path="Customer\folder\Geometry\folder\Country\dimension"
   id="_IBo88NrltEeck0Ylv-tlf2Q"/>
            <resultObject path="Customer\folder\Customer\dimension"
   id="_IBo8OrIhEeck0Ylv-tlf2Q"/>
            <resultObject path="Customer Satisfaction\folder\SatisfactionIndex\measure"
   id="_IB07ThEeck0Ylv-tlf2Q"/>
         </resultObjects>
         <filterPart>
            <and>
               <comparisonFilter path="Time\folder\Calendar\folder\CalendarYear Month\dimension"
   operator="EqualTo"
   id="_IBo8G7HhEeck0Ylv-tlf2Q">
                  <constantOperand>
                     <answerValue>CY2011-M10</answerValue>
                  </constantOperand>
               </comparisonFilter>
               <comparisonFilter path="Customer\folder\Geometry\folder\Continent\dimension"
   operator="EqualTo"
   id="_IBo8M7HhEeck0Ylv-tlf2Q">
                  <constantOperand>
                     <answerValue>"300000000</answerValue>
                  </constantOperand>
               </comparisonFilter>
            </and>
         </filterPart>
      </queryData>
   </querySpecification>
</query>
```
6.6.6 Combined Queries

The BI Semantic Layer RESTful Web Service SDK allows a client tool to build an XML query specification that contains multiple queries combined using operators. Only one query result is returned.

The following operators are supported:

<table>
<thead>
<tr>
<th>Operator</th>
<th>XML Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNION</td>
<td>union</td>
</tr>
<tr>
<td>MINUS</td>
<td>minus</td>
</tr>
<tr>
<td>INTERSECT</td>
<td>intersect</td>
</tr>
</tbody>
</table>

See the SAP BusinessObjects Web Intelligence User Guide for more information on these operators.

**Example**

The following snippet of a query specification shows two queries combined with the MINUS operator.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<query xmlns="http://www.sap.com/rws/sl/universe" id="5897899822041415615" dataSourceType="unx" dataSourceId="5808">
  <querySpecification version="1.0">
    <queryOptions>
      ...
    </queryOptions>
    <minus>
      <queryData>
        <resultObjects>
          <resultObject path="Customer|folder\Customer|dimension" id="_IBo8OrIhEeCk0Ylv-tlF2Q"/>
          <resultObject path="Customer|folder\Geography|folder\Country|dimension" id="_IBo8NhIeEk2Yvlv-t1F2Q"/>
          <resultObject path="Customer Satisfaction|folder\Satisfaction Index|measure" id="_IBytE7IehEck0Yvlv-t1F2Q"/>
          <resultObject path="Customer Satisfaction|folder\Achievement rate|measure" id="_OwM8LO1eCMD5vI8SKUZA"/>
        </resultObjects>
        <sortObjects>
          <sortObject path="Customer Satisfaction|folder\Achievement rate|measure" sortType="Ascending" id="_OwM8LO1eCMD5vI8SKUZA"/>
        </sortObjects>
      </queryData>
    </minus>
  </querySpecification>
</query>
```
6.6.7 Query Workflows

This section explains how you call the REST APIs to create and run queries with or without parameters against universes.

Simple Query Workflow

1. Create the query and send it to the server using POST .../query.
2. Run the query using GET .../data.svc.
3. Retrieve the query results using GET .../data.svc/<flowName>.

The GET .../data.svc request actually runs the query on the query engine, accesses the data source, and keeps the result set in memory. Consequently, any further GET call retrieves the cached data without running the query again.
Workflow for a Query With Contexts or Prompts

1. Create the query and send it to the server using POST .../query.
2. Get the query parameters that need to be responded using GET .../parameters.
3. Respond to all of the parameters using PUT .../parameters.
   1. If all parameters have been answered and the query can be run, then a successful message is returned.
   2. If not, the PUT call returns the parameters that remain to be answered. Then, repeat PUT .../parameters until all of the parameters get answers and a success status is returned.
4. Run the query with the answered parameters using GET .../data.svc.
5. Retrieve the query results using GET .../data.svc/<flowName>.

Remember
The query execution returns an error message if not all of the parameters have been answered.

Workflow for a Query With Different Parameter Values

You can run the same query multiple times with different values for contexts or parameters to get different result sets.

1. Create the query and send it to the server using POST .../query.
2. Get the query parameters that need to be responded using GET .../parameters.
3. Respond to all of the parameters using PUT .../parameters until all of them have been answered. Repeat the call if necessary.
4. Run the query with the answered parameters using GET .../data.svc.
5. Retrieve the query results using GET .../data.svc/<flowName>.
6. Get the query parameters that need to be responded using GET .../parameters.

Note
Skip this step if the parameters have been kept in memory.

7. Respond to all of the parameters with new values using PUT .../parameters until all of them have been answered. Repeat the call if necessary.
8. Run the query with the parameters answered in the previous step using GET .../data.svc.
9. Retrieve the query results using GET .../data.svc/<flowName>.

On step 4, the query is run and the result set is saved in memory. On step 5, the data is returned from the cache. On step 7, the call to PUT .../parameters resets the cache. Some parameters are required to be answered before the result set can be retrieved. On step 8, another call to GET .../data.svc runs the query with the answers and saves the new data set in the cache.
Related Information

Getting the List of Parameters [page 159]
Responding to Parameters [page 161]

6.6.8 The OData Query Service

The BI Semantic Layer RESTful Web Service SDK allows client tools to use the Open Data (OData) protocol to perform the following tasks:

- Running a query statement
- Retrieving the query results
- Exposing the query results as OData flows

The OData query service exposes the query results according to a schema based on the Entity Data Model (EDM). The schema is described with the help of CSDL. CSDL is an XML format that describes the structure and semantics of Entity Data Model schemas. To learn more about the concepts that rule an entity data model, see the CSDL specification.

The OData query service exposes the results of a query as one flow. It returns multiple flows in the following cases:

- If the query specification defines combined queries
- If the query uses unlinked objects

The schema defines the metadata used to build the query results. It is represented as a file of the EDMX format.

The query is running when the OData service is called. The OData service does not support POST HTTP requests.

Mapping Query Data [page 128]
Naming the Properties [page 129]
Mapping Query Data Types [page 130]

6.6.8.1 Mapping Query Data

The EDM schema used by the OData service maps the following EDM elements:

- The EntityContainer contains EntitySets.
- EntitySets are mapped to query flows. An EntitySet is defined by an EntityType.
- An EntityType consists of a series of Properties. A Property is mapped to a result object defined in the Query Specification and used in a query flow.

The following table describes the attributes defined for a Property. Attributes prefixed with "sap" are specific to SAP BusinessObjects universes.
### Table 149:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name used by the OData service to identify the object.</td>
</tr>
<tr>
<td>Type</td>
<td>Result type.</td>
</tr>
<tr>
<td>sap:label</td>
<td>Original name of the object in the universe.</td>
</tr>
<tr>
<td>sap:objectKey</td>
<td>Object ID in the universe.</td>
</tr>
<tr>
<td>sap:qualification</td>
<td>Object type. Possible values are Dimension, Measure, and Attribute.</td>
</tr>
<tr>
<td>sap:projectionFunction</td>
<td>Aggregation function applied to an object of type Measure. Possible values are: Average, Count, First, Last, Max, Min, Sum, Delegated, and None.</td>
</tr>
</tbody>
</table>

#### Example

The following `<Property>` element defines an object used in the query:

```xml
<Property Name="Year" Type="Edm.String" Nullable="true" Unicode="true"
          sap:objectKey="OBJ_49" sap:qualification="Dimension"
          sap:label="Year"/>
```

### Incomplete Result Set

The `<Schema>` XML element has a `sap:isPartial` attribute that indicates whether the result set of the query is complete.

`sap:isPartial` is set to `false` if the result set to return is greater than the value of the "Max rows retrieved" option or if the the execution query time takes more time than the maximum timeout defined in the "Max retrieval time" option. It is set to `true` otherwise.

### Related Information

- Query Options [page 109]
- Getting the OData Flow Metadata [page 181]

### 6.6.8.2 Naming the Properties

When generating the XML flow, the object names are automatically converted into Property names. The original object name is stored in the `sap:label` attribute of the Property.

In the EDM schema, the value of the `Name` attribute of a Property must adhere to the following rules:
It cannot start with a number or punctuation character.

- It cannot contain spaces.
- It cannot contain colons (:), periods (.), and @ characters.
- Any name can be used, no words are reserved.
- It is not recommended to start with xml.

If the generated name is an empty string, then the Property name is set to col.

Generated names must be unique. If they appear to be identical after conversion, the Property name is suffixed with _<increment>, where <increment> starts with 1.

### Property Names and sap:label Values

These examples illustrate the naming rules for Name and the corresponding, original object names set in sap:label.

```xml
<Property Name="Year" ... sap:label="Year"/>
<Property Name="Reservation_Year" ... sap:label="Reservation Year"/>
<Property Name="My_Name" ... sap:label="My_Name"/>
<Property Name="My_Name_1" ... sap:label="My Name"/>
<Property Name="inval_d3_name_" ... sap:label="inval!d3name$"/>
<Property Name="col" ... sap:label="$"/>
<Property Name="col_1" ... sap:label="!"/>
```

### 6.6.8.3 Mapping Query Data Types

The types of the objects used in the query are mapped to the following OData types:

<table>
<thead>
<tr>
<th>OData Data Types</th>
<th>BI Semantic Layer Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edm.Binary</td>
<td>BLOB</td>
</tr>
<tr>
<td>Edm.Boolean</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>Edm.DateTime</td>
<td>CALENDAR_DATE, DATE, DATE_TIME</td>
</tr>
<tr>
<td>Edm.Decimal</td>
<td>CURRENCY, FLOAT</td>
</tr>
<tr>
<td>Edm.Double</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>Edm.Int32</td>
<td>INT</td>
</tr>
<tr>
<td>Edm.Null</td>
<td>NULL</td>
</tr>
<tr>
<td>Edm.String</td>
<td>STRING, UNKNOWN</td>
</tr>
<tr>
<td>Edm.Time</td>
<td>TIME, TIMESTAMP</td>
</tr>
</tbody>
</table>
6.7 Report Structure

In a Web Intelligence document, the following items help identify the reports and report elements:

- **Id**
  The unique identifier of a report or report element inside a document. It is an integer.

- **Reference**
  The identifier of a report element with respect to the report it belongs to. It is a base64 string of the format "X.Y.A[...]", where X is the report identifier, Y the report element identifier, and A the identifier of an instance of the report element. Instance identifiers are used to identify the same report element used several times in a report, for example a section in a report, or a cell in a table.

Where to Use References?

You use the reference of a report element to get:

- The node references of its corresponding map and the associated data. See Getting the Map of a Report [page 297].
- Its datapath. See Getting the Datapaths of a Report Element [page 361].
- Its dataset. See Getting the Dataset of a Report Element [page 364].

**Note**

You can use references or datapaths to retrieve the data associated with a specific instance of a report element.

**Example**

```xml
<report>
  <id>1</id>
  <name>Chart Demo</name>
  <reference>1.RS</reference>
  <showDataChanges>false</showDataChanges>
  ...
</report>
```

In the current document, the report identifier is "1" and the string "1.RS" is the reference of the report in the document.

```xml
<elements>
  <element type="Cell">
    <id>3</id>
    <reference>1.K.3</reference>
    ...
  </element>
</elements>
```

The report element identifier is "3". The string "1.K.3" represents the reference of the element in the report.

In addition, the report structure can be viewed as a tree, which means there is a parent/child relationship between reports and report elements. This mechanism allows to add an element in the report structure with accuracy.

**Example**

```xml
<elements>
  <element type="Cell">
    <id>14</id>
  </element>
</elements>
```
The report element of type "Cell" belongs to the report of which identifier is "1".

6.8 Schedules

Body Schema (XML)

(GET .../documents/<documentID>/schedules/<scheduleID>)

(POST .../documents/<documentID>/schedules)

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;id&gt;</td>
<td>integer</td>
<td>The schedule identifier</td>
</tr>
<tr>
<td>Note</td>
<td></td>
<td>This element only appears in the response of the GET schedule details call.</td>
</tr>
<tr>
<td>&lt;name&gt;</td>
<td>string</td>
<td>The schedule name</td>
</tr>
<tr>
<td>&lt;format&gt;</td>
<td>N/A</td>
<td>The document format. <strong>Attribute:</strong> type (webi</td>
</tr>
<tr>
<td>Element</td>
<td>Type or Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>&lt;status&gt;</code></td>
<td><code>string</code></td>
<td>The schedule status. <strong>Attribute:</strong> <code>id</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each <code>id</code> can help you to test the schedule independently of your language locale.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>&lt;status id=&quot;0&quot;&gt;Running&lt;/status&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>&lt;status id=&quot;1&quot;&gt;Completed&lt;/status&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>&lt;status id=&quot;3&quot;&gt;Failed&lt;/status&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>&lt;status id=&quot;8&quot;&gt;Paused&lt;/status&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>&lt;status id=&quot;9&quot;&gt;Pending&lt;/status&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> This element only appears in the response of the <code>GET</code> schedule details call.</td>
</tr>
<tr>
<td><code>&lt;updated&gt;</code></td>
<td><code>string</code></td>
<td>The timestamp of the last schedule</td>
</tr>
<tr>
<td><code>&lt;destination</code></td>
<td>N/A</td>
<td>The document destination, which can be the BI launch pad inbox, an email address, a local file, or a file via FTP. Is optional. If no destination is specified, then the destination is the default one defined on the backend system. See the examples to learn how to set up the destination. <strong>Attribute:</strong> <code>keepInstanceInHistory(Boolean)</code> keeps the instance of the schedule document in history if it is <code>true</code> (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>&lt;useSpecificName&gt;</code> is the specific name of the scheduled document at destination. A name is assigned automatically if the element is not specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>fileExtension</code> specifies a file extension. Default is <code>true</code>.</td>
</tr>
<tr>
<td><code>&lt;error&gt;</code></td>
<td>N/A</td>
<td>Error information added in case of failure (<code>&lt;status id=&quot;3&quot;&gt;Failed&lt;/status&gt;</code>). <strong>Note</strong> This element can only appear in the response of the <code>GET</code> schedule details call.</td>
</tr>
<tr>
<td><code>recurrence_expression</code></td>
<td>N/A</td>
<td>The scheduling recurrence expression. See the next section to learn how to set it up.</td>
</tr>
</tbody>
</table>
### CSV Document Format

If the format type is CSV, the following options are supported:

```xml
<format type="csv">
  <properties>
    <property key="textQualifier">'</property> <!-- either ' or " -->
    <property key="columnDelimiter">,</property> <!-- either , or ; or tab -->
    <property key="charset">UTF-8</property> <!-- a valid server charset -->
    <property key="onePerDataProvider">false</property>
  </properties>
</format>
```

### Recurrence Expressions

The following table describes the possible recurrence expressions.

<table>
<thead>
<tr>
<th>Recurrence Expression</th>
<th>Schedule Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>No recurrence expression</td>
<td>Now</td>
</tr>
<tr>
<td>Recurrence Expression</td>
<td>Schedule Time</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>&lt;once retriesAllowed=&quot;2&quot; retryIntervalInSeconds=&quot;60&quot;&gt;</td>
<td>Once, run at &lt;startdate&gt;.</td>
</tr>
<tr>
<td></td>
<td>- retriesAllowed: the number of retries (integer)</td>
</tr>
<tr>
<td></td>
<td>- retryIntervalInSeconds: the time between two retries (integer)</td>
</tr>
<tr>
<td></td>
<td>- &lt;startdate&gt;: the date and time to run the schedule (dateTime)</td>
</tr>
<tr>
<td></td>
<td>- &lt;enddate&gt;: the date and time to stop the schedule (dateTime)</td>
</tr>
<tr>
<td>&lt;daily retriesAllowed=&quot;2&quot; retryIntervalInSeconds=&quot;60&quot;&gt;</td>
<td>Daily.</td>
</tr>
<tr>
<td></td>
<td>- &lt;dayinterval&gt; is the day interval to repeat the schedule (integer).</td>
</tr>
<tr>
<td>&lt;hourly retriesAllowed=&quot;2&quot; retryIntervalInSeconds=&quot;60&quot;&gt;</td>
<td>Hourly.</td>
</tr>
<tr>
<td></td>
<td>- &lt;hour&gt; (integer, [0, 24]) and &lt;minute&gt; (integer, [0, 59]) define the time interval to repeat the schedule.</td>
</tr>
<tr>
<td>&lt;weekly retriesAllowed=&quot;2&quot; retryIntervalInSeconds=&quot;60&quot;&gt;</td>
<td>Weekly. The schedule is run only the mentioned days.</td>
</tr>
<tr>
<td></td>
<td>- &lt;monday/&gt;., &lt;tuesday/&gt;., &lt;wednesday/&gt;., &lt;thursday/&gt;., &lt;friday/&gt;., &lt;saturday/&gt;., and &lt;sunday/&gt; are all optional. They must be mentioned in the specified order.</td>
</tr>
<tr>
<td>&lt;monthly retriesAllowed=&quot;2&quot; retryIntervalInSeconds=&quot;60&quot;&gt;</td>
<td>Monthly.</td>
</tr>
<tr>
<td></td>
<td>- &lt;month&gt; defines the month interval to repeat the schedule (integer, [1, 12]).</td>
</tr>
<tr>
<td>&lt;nthDayOfMonth retriesAllowed=&quot;2&quot; retryIntervalInSeconds=&quot;60&quot;&gt;</td>
<td>Only the specified day of each month.</td>
</tr>
<tr>
<td></td>
<td>- &lt;day&gt; defines the day of the month as a number between 1 and 31.</td>
</tr>
<tr>
<td>&lt;firstMondayOfMonth retriesAllowed=&quot;2&quot; retryIntervalInSeconds=&quot;60&quot;&gt;</td>
<td>The first Monday of each month.</td>
</tr>
<tr>
<td>Recurrence Expression</td>
<td>Schedule Time</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td><code>&lt;lastDayOfMonth retriesAllowed=&quot;2&quot; retryIntervalInSeconds=&quot;60&quot;&gt;</code></td>
<td>The last day of each month.</td>
</tr>
<tr>
<td><code>&lt;startdate&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;enddate&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xDayOfNthWeekOfTheMonth retriesAllowed=&quot;2&quot; retryIntervalInSeconds=&quot;60&quot;&gt;</code></td>
<td>The specified day of the specified week of each month.</td>
</tr>
<tr>
<td><code>&lt;startdate&gt;</code></td>
<td><code>&lt;enddate&gt;</code></td>
</tr>
<tr>
<td><code>&lt;day&gt;</code></td>
<td><code>&lt;week&gt;</code></td>
</tr>
<tr>
<td><code>&lt;calendar retriesAllowed=&quot;2&quot; retryIntervalInSeconds=&quot;60&quot;&gt;</code></td>
<td>The schedule runs according to the recurrence defined in the calendar that the end-user has created on the CMC.</td>
</tr>
<tr>
<td><code>&lt;startdate&gt;</code></td>
<td><code>&lt;enddate&gt;</code></td>
</tr>
<tr>
<td><code>&lt;id&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>
Response Body Schema (XML)

(GET .../universes/<universeID>)

```
<universe>
  <id>
  <cuid>
  <name>
  <description>
  <type>
  <folderId>
  <path>
  <maxRowsRetrieved>
  <maxRetrievalTime>
  <connected>
    <outline aggregated="Boolean">
      <businessViewName>
      <item>
        <folder>
          <id>
          <name>
          <description>
          <customProperty name="string">
          <item>
          <folder>
```

Universe

Table 152:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;id&gt;</td>
<td>integer</td>
<td>The universe identifier</td>
</tr>
<tr>
<td>&lt;cuid&gt;</td>
<td>string</td>
<td>The unique CUID of the universe in the CMS repository</td>
</tr>
<tr>
<td>&lt;name&gt;</td>
<td>string</td>
<td>The universe name in the requested locale</td>
</tr>
<tr>
<td>&lt;description&gt;</td>
<td>string</td>
<td>The universe description</td>
</tr>
<tr>
<td>&lt;type&gt;</td>
<td>unv</td>
<td>unx</td>
</tr>
<tr>
<td>&lt;folderId&gt;</td>
<td>integer</td>
<td>The ID of the folder that contains the universe in the CMS repository</td>
</tr>
<tr>
<td>&lt;path&gt;</td>
<td>string</td>
<td>The universe path in the CMS repository (returned if the GET call is made</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with the Web Intelligence RESTful Web Service SDK)</td>
</tr>
<tr>
<td>&lt;maxRowsRetrieved&gt;</td>
<td>integer</td>
<td>An option that defines the restrictions from the data source that can apply</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to queries</td>
</tr>
<tr>
<td>&lt;maxRetrievalTime&gt;</td>
<td>integer</td>
<td>An option that defines the restrictions from the data source that can apply</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to queries</td>
</tr>
<tr>
<td>&lt;connected&gt;</td>
<td>Boolean</td>
<td>true if the connection with the RDBMS is set (returned if the GET call is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>made with the Web Intelligence RESTful Web Service SDK)</td>
</tr>
</tbody>
</table>
### Outline

Table 153:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;businessViewName&gt;</code></td>
<td><code>string</code></td>
<td>The name of the default view if the master view is denied</td>
</tr>
<tr>
<td><code>&lt;item&gt;</code></td>
<td>N/A</td>
<td>An object of the universe</td>
</tr>
<tr>
<td><code>&lt;folder&gt;</code></td>
<td>N/A</td>
<td>A UNX universe folder or UNV universe class</td>
</tr>
</tbody>
</table>

### Item

```xml
<item type="string" dataType="string" hasLov="Boolean" forResult="Boolean"
forFilter="Boolean" forSort="Boolean">
  <id>
    <name>
      <description>
        <customProperty name="string">
          <item>
            <path>
              <aggregationFunction>
            </path>
          </item>
        </customProperty>
      </description>
    </name>
  </id>
</item>
```

Table 154:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;id&gt;</code></td>
<td><code>string</code></td>
<td>The object identifier</td>
</tr>
<tr>
<td><code>&lt;name&gt;</code></td>
<td><code>string</code></td>
<td>The object name</td>
</tr>
<tr>
<td><code>&lt;description&gt;</code></td>
<td><code>string</code></td>
<td>The object description</td>
</tr>
<tr>
<td><code>&lt;customProperty&gt;</code></td>
<td><code>string</code></td>
<td>Optional custom property for UNX universes only</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Attribute:</strong> name (string)</td>
</tr>
<tr>
<td><code>&lt;item&gt;</code></td>
<td>N/A</td>
<td>A child object</td>
</tr>
<tr>
<td><code>&lt;path&gt;</code></td>
<td><code>string</code></td>
<td>The object full path, which adheres to some particular rules. See Object Full Paths [page 93].</td>
</tr>
</tbody>
</table>
Element | Type or Value | Description
--- | --- | ---
<aggregationFunction> | Average|Sum|Count|First|Last|Max|Min|Delegated|none | The aggregation function used for measures only. Default is none.

**Note**
These attributes cannot be used with folders and predefined filters.

### Table 155:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>The object type in the universe (dimension, measure, attribute, and so on)</td>
</tr>
<tr>
<td>dataType</td>
<td>String</td>
<td>Numeric</td>
</tr>
<tr>
<td>hasLov</td>
<td>Boolean</td>
<td>Specifies whether the object has a list of values (for dimensions, measures, attributes, levels and hierarchies)</td>
</tr>
<tr>
<td>forResult</td>
<td>Boolean</td>
<td>Specifies whether the object can be used as a result object (default is true)</td>
</tr>
<tr>
<td>forFilter</td>
<td>Boolean</td>
<td>Specifies whether the object can be used as filter (default is true)</td>
</tr>
<tr>
<td>forSort</td>
<td>Boolean</td>
<td>Specifies whether the object can be used to sort data (default is true)</td>
</tr>
</tbody>
</table>

### Folder

### Table 156:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;id&gt;</td>
<td>string</td>
<td>The folder identifier (for UNX universes only)</td>
</tr>
<tr>
<td>&lt;name&gt;</td>
<td>string</td>
<td>The folder name</td>
</tr>
<tr>
<td>&lt;description&gt;</td>
<td>string</td>
<td>The folder description</td>
</tr>
<tr>
<td>&lt;customProperty&gt;</td>
<td>string</td>
<td>Optional custom property for UNX universes only</td>
</tr>
<tr>
<td>Attribute:</td>
<td>name</td>
<td>(string)</td>
</tr>
<tr>
<td>&lt;item&gt;</td>
<td>N/A</td>
<td>An object of the folder</td>
</tr>
<tr>
<td>&lt;folder&gt;</td>
<td>N/A</td>
<td>A subfolder</td>
</tr>
</tbody>
</table>

### Related Information

Getting the Details of a Universe (Semantic Layer) [page 143]
6.10 User Rights

User Rights Applied to UNV Universes

User rights that applied to UNV universes fall into three categories:

- **Security rights**
  Security rights can be set at universe design tool, universe and relational connection level.

- **Universe overloads**
  The rights defined on universes are Connections, Controls, SQL, Objects, Rows, and Table Mapping.

- **Object access level**
  The object access level defines a restriction level for users (Private, Public, Confidential...) A user who does not have the required access level cannot view the objects beyond his scope.

These user rights are applied before an application user makes use of the REST APIs and therefore are transparent to the REST API user. For example, objects denied by the access level are not returned in the universe outline, when getting it through `GET .../universes/<universeID>`.

Connections, Objects, Rows, and Table Mapping rights are applied when getting the query results through the OData service or when getting the list of values through `GET .../queries/<queryID>/parameters`.

Only a few security rights can have an impact on the use of the REST APIs. The following table presents them.

Table 157:

<table>
<thead>
<tr>
<th>Right</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“View objects” at relational connection level</td>
<td>If this right is disabled, the user cannot see the connection and the query cannot be run.</td>
</tr>
<tr>
<td>“View objects” at universe level</td>
<td>If this right is disabled, the universe is not displayed in the universe list and any call with the universe ID returns an error.</td>
</tr>
<tr>
<td>“Data access”</td>
<td>This custom security right is defined at universe level or relational connection level. If this right is disabled, any call to the OData service returns an error.</td>
</tr>
</tbody>
</table>

User Rights Applied to UNX Universes

The same user rights apply to UNX universes. Universe overloads are managed through business and data security profiles. Business security profiles manage rights on metadata at the business layer level. Data security profiles manage rights at the data foundation level. These profiles are attached to users or groups of users. The same security rights can have an impact on the use of the REST APIs.
7 BI Semantic Layer REST API Reference

Reference of the BI Semantic Layer REST APIs.

Each API reference provides the following information:

- What the API does
- URL of the HTTP request
- Description of the request query parameters
- Request body if needed (XML or JSON example used)
- Response type, which determines the value of the `accept` header attribute
- Response body (not described if the response is a success or error message)
- Response examples

Managing Universes with the BI Semantic Layer RESTful Web Service SDK [page 141]
Managing Queries [page 154]
Managing Parameters [page 158]
Managing Query Results [page 179]

7.1 Managing Universes with the BI Semantic Layer RESTful Web Service SDK

The BI Semantic Layer RESTful Web Service SDK provides a series of APIs to perform on UNX and UNV universes.

- Getting the List of the Universes (Semantic Layer) [page 141]
- Getting the Details of a Universe (Semantic Layer) [page 143]
- Getting the Business Views of a Universe [page 146]
- Getting the Query Capabilities of a Universe (Semantic Layer) [page 149]
- Getting the Link Groups of a Universe [page 151]
- Getting the Object Parameters of a Universe [page 153]

7.1.1 Getting the List of the Universes (Semantic Layer)

Usage

Gets the list of UNX and UNV universes stored in the CMS repository.

The universe list can be displayed on pages.
Request

GET /universes?offset=<offset>&limit=<limit>

Where:

- <offset> indicates the position in the list, from which universes are returned. It must be greater than or equal to 0. The default value is 0. This parameter is optional.
- <limit> indicates the number of universes that you can list on one page. The range is [1, 50]. The default value is 10. This parameter is optional.

Response

Response type: application/xml

Response body: the list of the universes identified by the following elements:

- <id>
- <cuid>
- <name> in the requested locale
- <type>(unv or unx)
- <folderId>, which is the ID of the folder that contains the universe in the CMS repository

Example

GET /universes?offset=0&limit=50

```xml
<universes>
  <universe>
    <id>6773</id>
    <cuid>AXyRzvmRrJxLqUm6_Jbf71E</cuid>
    <name>efashion.unv</name>
    <type>unv</type>
    <folderId>6771</folderId>
  </universe>
  <universe>
    <id>5808</id>
    <cuid>AUW2qRdU0IdPkyhlpZWrxvo</cuid>
    <name>Warehouse.unx</name>
    <type>unx</type>
    <folderId>5807</folderId>
  </universe>
  ...
</universes>
```
7.1.2 Getting the Details of a Universe (Semantic Layer)

Usage

Gets the details of a UNX or UNV universe referenced by its ID.

Request

GET /universes/<universeID>?aggregated=<aggregated>

Where:

- `<aggregated>` is an optional, Boolean parameter that indicates if the outline must be aggregated. Default value is `false`.

Influence of the aggregated Parameter

The UNX universe details returned depend on the `<aggregated>` parameter value:

- If `false`, the call returns the master view if granted or the default view if the master view is denied. The default view name is returned in the outline using the `<businessViewName>` element. This behavior is the one implemented in the SDK versions prior to 4.1 SP5.
- If `true`, the call returns an aggregated outline containing all folders and objects granted to the user. This outline merges all granted objects from the granted business view and contains all objects properties such as ID, path, and name.

If the master view is granted, then the outline returned with the calls .../<universeID>?aggregated=false and .../<universeID>?aggregated=true are the same, except the value of aggregated outline attribute.

Even if UNV universes do not support the concept of view introduced with UNX universes, for API consistency, the BI Semantic Layer RESTful Web Service can also behave as if a UNV universe had only one master view containing all the universe content. When returning this view, the `<aggregated>` parameter has no impact on the response and the value of the aggregated outline attribute is `false` in the response.

Response

Response type: application/xml or application/json

Response body: the details of the universe. See Universes [page 136].

Example

The Master View is Granted

GET /universes/5808?aggregated=false
or

GET /universes/5808

Response:

```xml
<universe id="5808">
  <cuid>AUN2qRdU0IIdPkyhlpz2Wrxvo</cuid>
  <name>Warehouse.unx</name>
  <type>unx</type>
  <folderId>5807</folderId>
  <maxRowsRetrieved>5000</maxRowsRetrieved>
  <maxRetrievalTime>600</maxRetrievalTime>
  <outline aggregated="false">
    <folder id="_IBo8OLIhEeCk0Ylv-tlF2Q">
      <name>Customer</name>
      <item dataType="Numeric" type="Dimension" hasLov="false">
        <id>_IBo8OLIhEeCk0Ylv-tlF2Q</id>
        <name>Customer Id</name>
        <path>Customer</path>
      </item>
      <item dataType="String" type="Dimension" hasLov="false">
        <id>_IBo8ONIhEeCk0Ylv-tlF2Q</id>
        <name>Customer</name>
        <path>Customer</path>
      </item>
    </folder>
    <folder id="_IB8eE7IhEeCk0Ylv-tlF2Q">
      <name>Geography</name>
      <item dataType="Numeric" type="Dimension" hasLov="false">
        <id>_IB8eE7IhEeCk0Ylv-tlF2Q</id>
        <name>Continent Id</name>
        <path>Customer|folder|Geography</path>
      </item>
      <item dataType="String" type="Dimension" hasLov="false">
        <id>_IB8eE7IhEeCk0Ylv-tlF2Q</id>
        <name>Continent</name>
        <path>Customer|folder|Geography</path>
      </item>
    </folder>
    <folder id="_H4I54LaXEeCH2d6gof1MOA">
      <name>Quarterly KPIs</name>
      <item dataType="Numeric" type="Measure" hasLov="false">
        <id>_H4I54LaXEeCH2d6gof1MOA</id>
        <name>Quarterly KPIs</name>
      </item>
    </folder>
  </outline>
</universe>
```
<path>Quarterly KPIs|folder\Reference Quarter|measure</path>
<aggregationFunction>Sum</aggregationFunction>
</item>
</folder>
</outline>
<universe>
  <id>5808</id>
  <cuid>AUW2q8dU0IdPkyhlpZWrxvo</cuid>
  <name>Warehouse.unx</name>
  <type>unx</type>
  <folderId>5807</folderId>
  <maxRowsRetrieved>5000</maxRowsRetrieved>
  <maxRetrievalTime>600</maxRetrievalTime>
  <outline aggregated="false">
    <BusinessViewName>MyView</BusinessViewName>
    <folder>
      ...
    </folder>
  </outline>
</universe>

Example

The Master View is Denied

GET /universes/5808?aggregated=false

or

GET /universes/5808

Response:

Example

The Call Requests an Aggregated Outline

GET /universes/5808?aggregated=true

Response:
Chrome Frame is not supported.
Please use another browser.

Related Information

Getting the List of the Universes (Semantic Layer) [page 141]

7.1.3 Getting the Business Views of a Universe

Usage

Table 158:

<table>
<thead>
<tr>
<th>Universe</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNV</td>
<td>The API returns a view considered as the master view of a UNV universe.</td>
</tr>
<tr>
<td>UNX</td>
<td>The API returns the business views of a UNX universe. The method returns only the views that are granted to the user that has opened the session. The views, including the master view, which are denied because of security settings, are not returned.</td>
</tr>
</tbody>
</table>

Note

For each business view, the user can navigate through tree nodes to get the objects that can be displayed in the outline. Each tree node refers either to a folder or to an object of the business layer by its identifier.

Request

GET /universes/<universeID>/businessviews

Request type: application/xml

Response

Response type: application/xml

Response body:
### Table 159:

<table>
<thead>
<tr>
<th>Universe</th>
<th>Response Body</th>
</tr>
</thead>
</table>
| UNV      | A view considered as the master view (masterView="true") that contains all classes and objects of the universe with:  
  - `<description>`, if the description is not available, the `<description>` element does not display.  
  - `<folderRef>` refers to a class of the view.  
  - `<itemRef>` refers to an object of the business layer.  
  There is only one view. |
| UNX      | The master view (masterView="true") and custom views for a user (masterView="false"), identified with:  
  - `<id>`  
  - `<name>`  
  - `<description>`, if the description is not available, the `<description>` element does not display.  
  - `<folderRef>` refers to a folder of the business view.  
  - `<itemRef>` refers to an object of the business layer.  
  There is only one master view. There can be several custom views. |

### Example

**Getting the View of a UNV Universe**

GET /universes/11572/businessviews

```xml
<businessViews xmlns="http://www.sap.com/rws/sl/universe">
    <businessView masterView="true">
        <folderRef name="Resort">
            <itemRef id="DO39"/>
            <itemRef id="DO2"/>
            <itemRef id="DO4"/>
            <itemRef id="DFc"/>
            <itemRef id="DF1"/>
            <itemRef id="DF2"/>
        </folderRef>
        <folderRef name="Sales">
            <itemRef id="DO2f"/>
            <itemRef id="DO31"/>
            <itemRef id="DO32"/>
            <itemRef id="DO33"/>
            <itemRef id="DO17"/>
        </folderRef>
        <folderRef name="Customer">
            <itemRef id="DO11"/>
            <itemRef id="DO12"/>
            <itemRef id="DOd">
                <itemRef id="DO9"/>
                <itemRef id="DOe"/>
                <itemRef id="D08"/>
            </itemRef>
            <itemRef id="DO19"/>
            <itemRef id="DF4"/>
            <itemRef id="DFa"/>
            <itemRef id="DFb"/>
            <folderRef name="Sponsor">
                <itemRef id="DO2b"/>
                <itemRef id="DO26"/>
            </folderRef>
        </folderRef>
    </businessView>
</businessViews>
```
Example

Getting the Business Views of a UNX Universe

GET /universes/5808/businessviews

```xml
<businessViews xmlns="http://www.sap.com/rws/sl/universe">
  <businessView masterView="true">
    <id>PER__6FCF0NdGEeKfwPJJ7KJ5ow</id>
    <masterView name="UniverseName" />
    <folderRef id="_6HHwgtdGEeKfwPJJ7KJ5ow">
      <itemRef id="_6HHwg9dGEeKfwPJJ7KJ5ow"/>
      <itemRef id="_6HRhgNdGEeKfwPJJ7KJ5ow"/>
      <itemRef id="_6HRhgddGEeKfwPJJ7KJ5ow"/>
      <itemRef id="_6opwQNdGEeKfwPJJ7KJ5ow"/>
      <itemRef id="_rhz58Oo_EeKgAukdblEjyA"/>
    </folderRef>
    <folderRef id="_6HuNcddGEeKfwPJJ7KJ5ow">
      <itemRef id="_6H3XYNdGEeKfwPJJ7KJ5ow"/>
      <itemRef id="_6H3XYddGEeKfwPJJ7KJ5ow"/>
      <itemRef id="_6IBIYNdGEeKfwPJJ7KJ5ow"/>
    </folderRef>
  </businessView>
  <businessView masterView="false">
    <id>ouYozyJyEeOx7dXQSVnuKw</id>
    <description>View description</description>
    <folderRef id="_6HHwqtdGEeKfwPJJ7KJ5ow">
      <itemRef id="_6HHwq9dGEeKfwPJJ7KJ5ow"/>
      <itemRef id="_6HRhgNdGEeKfwPJJ7KJ5ow"/>
    </folderRef>
    <folderRef id="_6HuNcddGEeKfwPJJ7KJ5ow">
      <itemRef id="_6H3XYNdGEeKfwPJJ7KJ5ow"/>
      <itemRef id="_6H3XYddGEeKfwPJJ7KJ5ow"/>
    </folderRef>
  </businessView>
</businessViews>
```
7.1.4 Getting the Query Capabilities of a Universe (Semantic Layer)

Usage

Gets the query capabilities of a UNX or UNV universe, depending on the user rights.
These capabilities list the features supported by the data source when building the query from the query panel.

Request

GET /universes/<universeID>/capabilities

Request type: application/xml

Response

Response type: application/xml

Response body:

Table 160:

<table>
<thead>
<tr>
<th>Capabilities</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General query capabilities</td>
<td>&lt;combinedQueriesSupported&gt;</td>
<td>true if the data source supports the combined queries.</td>
</tr>
<tr>
<td></td>
<td>&lt;maxValuesForInList&gt;</td>
<td>The maximum number of items that can be set in a filter based on INLIST or NOT_INLIST operators.</td>
</tr>
<tr>
<td>Data processing capabilities</td>
<td>&lt;removeDuplicateRowsAvailable&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;samplingLevel&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;maxRowRetrievedAvailable&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;maxRetrievalTimeAvailable&gt;</td>
<td></td>
</tr>
<tr>
<td>Capabilities</td>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Filter capabilities</td>
<td>&lt;supportedComparisonOperators&gt;</td>
<td>The list of comparison operators supported for filters based on dimensions, attributes or measures.</td>
</tr>
<tr>
<td></td>
<td>&lt;supportedLogicalOperators&gt;</td>
<td>The list of logical operators supported to combine filters.</td>
</tr>
<tr>
<td></td>
<td>&lt;supportedObjects&gt;</td>
<td>The list of object qualifications (dimensions, attributes, and measures) supported to build queries.</td>
</tr>
<tr>
<td></td>
<td>&lt;supportedRankings&gt;</td>
<td>The list of supported ranking operators (top, bottom, topPercent, or bottomPercent)</td>
</tr>
<tr>
<td></td>
<td>&lt;resultObjectInFilterSupported&gt;</td>
<td>true if an object used as a result object can be used in a filter.</td>
</tr>
<tr>
<td></td>
<td>&lt;objectComparisonSupported&gt;</td>
<td>true if filter operands based on another object are supported.</td>
</tr>
<tr>
<td></td>
<td>&lt;constantComparisonSupported&gt;</td>
<td>true if filter operands based on constant values are supported.</td>
</tr>
<tr>
<td></td>
<td>&lt;subQueriesSupported&gt;</td>
<td>true if subquery filters are supported.</td>
</tr>
</tbody>
</table>
| Result object capabili-
| ties               | <useAttributeSeparately Available>               |                                                                             |
|                      | <sortObjectsAvailable>                           | true if query sorts from objects are supported.                             |
|                      | <sortNonResultObjectsAvailable>                  | true if query sorts from objects that are not used as result objects are supported. |

Example

GET /universes/5808/capabilities

```xml
<capabilities xmlns="http://www.sap.com/rws/sl/universe">
  <GeneralCapabilities>
    <combinedQueriesSupported>true</combinedQueriesSupported>
    <maxValuesForInList>-1</maxValuesForInList>
  </GeneralCapabilities>
  <DataProcessingCapabilities>
    <removeDuplicateRowsAvailable>true</removeDuplicateRowsAvailable>
    <samplingLevel>None</samplingLevel>
    <maxRowRetrievedAvailable>true</maxRowRetrievedAvailable>
    <maxRetrievalTimeAvailable>true</maxRetrievalTimeAvailable>
  </DataProcessingCapabilities>
  <filterCapabilities>
    <supportedComparisonOperators>
      <value>EqualTo</value>
      <value>NotEqualTo</value>
      <value>Between</value>
      <value>NotBetween</value>
      ...
    </supportedComparisonOperators>
    <supportedRankings>
      <value>top</value>
      <value>bottom</value>
      <value>topPercent</value>
      <value>bottomPercent</value>
    </supportedRankings>
  </filterCapabilities>
</capabilities>
```
Related Information

Getting the List of the Universes (Semantic Layer) [page 141]

7.1.5 Getting the Link Groups of a Universe

Usage

Gets the link groups of a UNX universe published to a CMS repository.

Note

Link groups of UNV universes are not supported in this release.

About Link Groups

In a relational universe, a link group is a set of business objects sharing a dependency link that is expressed in the universe data source model (data foundation) or the universe semantic layer model (business layer). A link group can be one of the following:

- A data link group
  It is based on the data foundation and it expresses relationships between objects, such as a dimension and another dimension, or a dimension and a measure. A data link group can only contain dimensions and measures. A data link group can be seen as a measure group if it contains at least one measure.

- A semantic link group
It is based on the business layer and it expresses relationships between objects, such as a dimension and an attribute. A semantic link group must contain one dimension and its associated attributes if any.

A link group cannot be included into another link group. Link groups can have common dimensions or measures. A universe contains at least one link group, otherwise it is empty.

**Request**

GET /universes/<universeID>/linkgroups

Request type: application/xml

**Response**

Response type: application/xml

Response body:

```xml
<linkGroups>
  <linkGroup type="semantic">
    <link name="string" id="string"/>
  </linkGroup>
</linkGroups>
```

Where:

- The possible values of the link group type are:
  - data, if the link group is based on the data foundation
  - semantic, if it is based on the business layer
- name is the business object path
- ID is the identifier of the business object identifier

**Example**

```xml
<linkGroups xmlns="http://www.sap.com/rws/sl/universe">
  <linkGroup type="semantic">
    <link name="Customer\folder\Customer\dimension" id="OBJ_13"/>
    <link name="Customer\folder\Customer\dimension\Age\attribute" id="OBJ_9"/>
  </linkGroup>
  <linkGroup type="data">
    <link name="Resort\folder\Service\dimension" id="OBJ_5"/>
    <link name="Sales\folder\Sales Date\dimension" id="OBJ_23"/>
    <link name="Customer\folder\Service Line\dimension" id="OBJ_4"/>
    <link name="Resort\folder\Resort\dimension" id="OBJ_13"/>
    <link name="Resort\folder\Resort Country\dimension" id="OBJ_6"/>
    <link name="Customer\folder\Customer Country\dimension" id="OBJ_17"/>
    <link name="Measures\folder\Revenue\measure" id="OBJ_7"/>
    <link name="Measures\folder\Number of guests\measure" id="OBJ_20"/>
  </linkGroup>
</linkGroups>
```

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer
BI Semantic Layer REST API Reference
7.1.6 Getting the Object Parameters of a Universe

Usage

Gets the list of object parameters used to prompt some values to the end-user and defined in data foundations and business layers of a UNX universe stored in the CMS repository.

Note

This only relates to UNX universes since UNV universes do not support object parameters.

Request

GET /universes/<universeID>/prompts

Response

Response type: application/xml or application/json

Response body: the list of the object parameters (<prompt>) identified by the following elements:

- <id>
- <dataType>
- <name>
- <question>
Example

GET /universes/6660/prompts

```xml
<prompts>
  <prompt>
    <id>_HDBaULaYeeCH2d6gof1MOA</id>
    <dataType>Numeric</dataType>
    <name>Reference Quarter Number</name>
    <question>Reference Quarter Number:</question>
  </prompt>
  <prompt>
    <id>_PImaULaYeeCH2d6gof1MOA</id>
    <dataType>Numeric</dataType>
    <name>Reference Year Number</name>
    <question>Reference Year Number:</question>
  </prompt>
  <prompt>
    <id>_capysLIYeCVLNpRXueEka</id>
    <dataType>String</dataType>
    <name>Period Type</name>
    <question>Period Type:</question>
  </prompt>
  <prompt>
    <id>_lBRDMLIYeCVLNpRXueEka</id>
    <dataType>String</dataType>
    <name>Time Window</name>
    <question>Time Window:</question>
  </prompt>
  <prompt>
    <id>_uPlz8LIYeCVLNpRXueEka</id>
    <dataType>Date</dataType>
    <name>Reference Date</name>
    <question>Reference Date:</question>
  </prompt>
</prompts>
```

7.2 Managing Queries

The BI Semantic Layer RESTful Web Service SDK provides a series of APIs to query UNV and UNX universes and get data. Universe data is exposed as an OData service.

Creating a Query [page 155]

Getting the List of Queries [page 156]

Getting the Details of a Query [page 157]

Deleting a Query [page 158]
7.2.1 Creating a Query

Usage

Creates a precompiled query statement in an instance of the open session.

This instance can be used to run this statement multiple times. The query is not stored to the CMS repository, but in memory.

Request

POST /queries

Request type: application/xml

Request body: see query specification body schema [page 107]

Response

Response type: application/xml

The response is a message stating the success or failure of the request.

Example

Creating a Query with Result Objects Only

POST /queries

Request body:

```
<query dataSourceType="unix" dataSourceId="5808" xmlns="http://www.sap.com/rws/sl/universe">
  <querySpecification version="1.0">
    <queryData>
      <resultObjects>
        <resultObject path="Customer|folder\Geography|folder\City|dimension" id="_IBo8L7hEeCeK0Ylv-tlF2Q"/>
        <resultObject path="Customer|folder\Customer|dimension" id="_IBo8O1hEeCeK0Ylv-tlF2Q"/>
        <resultObject path="Inventory|folder\Stock Level|measure" id="_IB8eFtEeCeK0Ylv-tlF2Q"/>
      </resultObjects>
    </queryData>
  </querySpecification>
</query>
```

Response body:

```
<success>
  <message>The resource of type "query" with identifier "6089913651317040730" has been successfully created.</message>
</success>
```
Related Information

Deleting a Query [page 158]

7.2.2 Getting the List of Queries

Usage

Gets the list of all query statements available in the instance of the open session.

Request

GET /queries

Request type: application/xml

Response

Response type: application/xml

Response body: a series of <query> elements with the following attributes:

- `dataSourceType` is the universe type (unv or unx)
- `dataSourceId` is the universe identifier

A <query> has an <id> child element that represents the corresponding query statement identifier.

Example

GET /queries

```xml
<queries xmlns="http://www.sap.com/rws/sl/universe">
  <query dataSourceType="unx" dataSourceId="5845">
    <id>5602099021259262832</id>
  </query>
  <query dataSourceType="unx" dataSourceId="5845">
    <id>7059150136676433395</id>
  </query>
</queries>
```
7.2.3 Getting the Details of a Query

Usage

Returns the query specification of a query statement created in an instance of the open session.

Request

GET /queries/<queryID>

Request type: application/xml

Response

Response type: application/xml

Response body: see query specification body schema [page 107]

Example

GET /queries/608913651317040730

<query xmlns="http://www.sap.com/rws/sl/universe" id="608913651317040730"
    dataSourceType="unx" dataSourceId="5808">
    <querySpecification version="1.0">
        <queryData>
            <resultObjects>
                <resultObject path="Customer\folder\Geography\folder\City|dimension" id="_IBo8L7hEeCk0Ylv-t1P2Q"/>
                <resultObject path="Customer\folder\Customer|dimension" id="_IBo8ORlHeCk0Ylv-t1P2Q"/>
                <resultObject path="Inventory\folder\Stock Level|measure" id="_IB8eFrlHeCk0Ylv-t1P2Q"/>
            </resultObjects>
        </queryData>
    </querySpecification>
</query>

Related Information

Getting the List of Queries [page 156]
7.2.4 Deleting a Query

Usage

Deletes a query statement from memory in the instance of the open session.

Note

You can delete all query statements created in this instance by closing the session.

Request

DELETE /queries/<queryID>

Request type: application/xml

Response

Response type: application/xml

The response is a message stating the success or failure of the request.

Example

DELETE /queries/6089913651317040730

success

<message>The resource of type "query" with identifier "6089913651317040730" has been successfully removed.</message>

{id>6089913651317040730</id>

</success>

Related Information

Getting the List of Queries [page 156]

7.3 Managing Parameters

The BI Semantic Layer RESTful Web Service SDK provides a series of APIs to perform tasks on contexts and prompts of a UNV or UNX universe.
7.3.1 Getting the List of Parameters

Usage

Returns the first contexts or parameters to be answered.

Request

GET /queries/<queryID>/parameters

Response

Response type: application/xml or application/json

The response provides the parameters with their expected answers and default values. See Parameter Response Body Schemas [page 95] to learn about the content structure and element details.

Example

No parameter

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<parameters/>

Example

Context

The parameter of type context provides two possible context values.

<parameters>
  <parameter type="context" optional="false" dpId="DP0">
    <id>0</id>
    <name>Select a context</name>
    <answer type="text" constrained="true">
      <info cardinality="Single">
        <lov partial="false">
          <values>
            <value id="C2">Reservations</value>
            <value id="C1">Sales</value>
            </values>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
Example

DateTime Prompt

The parameter of type prompt accepts only one answer value (cardinality Single).

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP0">
    <id>0</id>
    <name>Enter Open Date:</name>
    <answer constrained="false" type="DateTime">
      <info cardinality="Single">
        <lov refreshable="true" partial="false" hierarchical="false">
          <values>
            <value>2011-09-03T17:15:00.000+02:00</value>  <!-- possible values -->
            <value>2012-09-03T17:15:00.000+02:00</value>
            <value>2013-09-03T17:15:00.000+02:00</value>
          </values>
        </lov>
        <value>2011-09-03T17:15:00.000+02:00</value>  <!-- default value -->
      </info>
    </answer>
  </parameter>
</parameters>
```

Example

Prompt with Multiple Values

The parameter of type prompt accepts multiple answer values (cardinality Multiple).

```xml
<parameters>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>0</id>
    <technicalName>pmmultiple</technicalName>
    <name>Enter Cityname (multiple)</name>
    <answer type="Text" constrained="true">
      <info cardinality="Multiple">
        <lov refreshable="true" partial="false" hierarchical="false">
          <id>UNIVERSELLOV_DS0.DO22</id>
          <values>
            <value>Barcelona</value>  <!-- possible values -->
            <value>Berlin</value>
            <value>Copenhagen</value>
            <value>Firenze</value>
            <value>London</value>
            <value>Madrid</value>
            <value>Marseilles</value>
            <value>Oslo</value>
            <value>Paris</value>
            <value>Pragues</value>
          </values>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```
Related Information

Parameter Response Body Schemas [page 95]
Getting the List of Queries [page 156]

7.3.2 Responding to Parameters

Usage

Responds to contexts and prompts.

Returns additional parameters if found when the first parameters have been responded. Since these new parameters must also be responded, you must perform an additional call by using as request body the answers to the new parameters, and the answers to the parameters that have triggered the new parameters.

Request

PUT /queries/<queryID>/parameters

Request body: the answers to the parameters retrieved using the GET .../parameters call. See Answer Request Body Schemas [page 104] to learn about the XML content structure and element details.
Response

Response type: application/xml or application/json

When all parameters have been answered, the last PUT call returns a message stating the success of the request.

```xml
<success>
  <message>The parameters of the resource of type "Query" with identifier "XX" have been answered.</message>
  <id>XX</id>
</success>
```

Example - Responding to a Context [page 162]

Example - Responding to a Prompt with Multiple Columns [page 163]

Example - Responding to two Prompts that Define an Interval [page 164]

Example - Responding to Cascading Parameters [page 166]

Example - Responding to a Hierarchical Parameter [page 170]

Related Information

Getting the List of Queries [page 156]

7.3.2.1 Example - Responding to a Context

A GET .../parameters call returns a list of values for a specific parameter of type context:

```xml
<parameters>
  <parameter dpId="DP0" type="context" optional="false">
    <id>0</id>
    <technicalName>c_Sub_2</technicalName>
    <name>Select a context</name>
    <answer type="Text" constrained="true">
      <info cardinality="Single">
        <lov partial="false">
          <value id="CTX_2">Reservations</value>
          <value id="CTX_1">Sales</value>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

In the PUT call, the request body contains an answer to the context. The following request body does not show the <info> element, which is not mandatory:

```xml
<parameters>
  <parameter dpId="DP0" optional="false" type="context">
    <id>0</id>
  </parameter>
</parameters>
```
The PUT call returns a successful message.

```
<success>
  <message>The resource of type "Query" with identifier "XX" has not been modified.</message>
  <id>XX</id>
</success>
```

7.3.2.2 Example - Responding to a Prompt with Multiple Columns

The GET .../parameters call returns an XML that describes a parameter of type prompt and cardinality Single that accepts an answer made of two column values. The Employee Name column is the one used as reference. The default value is Clive Lord.

```
<parameters>
  <parameter dpId="DP0" optional="false" type="prompt">
    <id>1</id>
    <technicalName>psCustomLOV_2col</technicalName>
    <name>Enter CustomLOV_2col:</name>
    <answer constrained="false" type="Text">
      <info cardinality="Single">
        <lov hierarchical="false" partial="false" refreshable="true">
          <id>UNIVERSELOV_DS0.DO65</id>
          <cvalues>
            <cvalue>
              <column id="0">Alan Leroy</column> <!-- possible values -->
              <column id="1">Sales</column>
            </cvalue>
            <cvalue>
              <column id="0">Brian Lunn</column>
              <column id="1">Board</column>
            </cvalue>
            <cvalue>
              <column id="0">Brian Opel</column>
              <column id="1">Sales</column>
            </cvalue>
            <cvalue>
              <column id="0">Chris Hutchings</column>
              <column id="1">Sales</column>
            </cvalue>
            <cvalue>
              <column id="0">Clive Allen</column>
              <column id="1">Support</column>
            </cvalue>
            <cvalue>
              <column id="0">Clive Lord</column>
              <column id="1">Sales</column>
            </cvalue>
          </cvalues>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```
The PUT .../parameters call responds to the parameter. The following request body does not show the <info> element, which is not mandatory. The answer value is the default value.

```
<parameters>
  <parameter dpId="DP0" optional="false" type="prompt">
    <id>1</id>
    <technicalName>psCustomLOV_2col</technicalName>
    <name>Enter CustomLOV_2col:</name>
    <answer constrained="false" type="Text">
      <values>
        <value>Clive Lord</value>
      </values>
    </answer>
  </parameter>
</parameters>
```

Response:

```
<success>
  <message>The resource of type "Query" with identifier "5661410011133153584" has not been modified.</message>
  <id>5661410011133153584</id>
</success>
```

### 7.3.2.3 Example - Responding to two Prompts that Define an Interval

The GET .../parameters call returns an XML that describes two parameters of type prompt and cardinality Single that define a journey between two European cities:

```
<parameters>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>0</id>
    <technicalName>psEnter City of Departure</technicalName>
    <name>Enter City of Departure</name>
    <answer type="Text" constrained="true">
      <info cardinality="Single">
        <value>Clive Lord</value>
      </info>
    </answer>
  </parameter>
</parameters>
```
The PUT .../parameters call responds to the prompts with the following request body:

```xml
<parameters>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>0</id>
    <technicalName>psEnter City of Departure</technicalName>
    <name>Enter City of Departure</name>
    <answer constrained="true" type="Text">
      <values>
        <value>Barcelona</value>  <!-- possible value -->
      </values>
    </answer>
  </parameter>
</parameters>
```
7.3.2.4 Example - Responding to Cascading Parameters

The GET .../parameters call returns:

- The list of possible values for the answer to a parameter of id=0, type prompt, and cardinality Multiple
- A second parameter of id=1, type prompt, and cardinality Single

The first parameter needs to be answered so that you can get the list of values of the second parameter.

```xml
<parameters>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>0</id>
    <technicalName>pmEnter First Name of Customer</technicalName>
    <name>Enter First Name of Customer</name>
    <answer type="Text" constrained="false">
      <info cardinality="Multiple">
        <lov refreshable="true" partial="false" hierarchical="false">
          <id>UNIVERSELOV_DS0.DO5a</id>
          <values>
            <value>Andre</value>
            <value>Annie</value>
            <value>Brian</value>
            <value>Christine</value>
            <value>Dan</value>
            <value>Erika</value>
            <value>Franz</value>
            <value>George</value>
            <value>Habib</value>
            <value>Jack</value>
            <value>Jan</value>
            <value>Joe</value>
            <value>John</value>
            <value>Luke</value>
            <value>Mary</value>
            <value>Michele</value>
            <value>Paul</value>
            <value>Peter</value>
            <value>Philip</value>
            <value>Robert</value>
          </values>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```
You answer the parameter with the PUT .../parameters call. First, you query the list of values according to a certain pattern (a*).

Request body:

```
<parameters>
  <parameter dpId="DP0" optional="false" type="prompt">
    <id>0</id>
    <technicalName>pmEnter First Name of Customer</technicalName>
    <name>Enter First Name of Customer</name>
    <answer constrained="false" type="Text">
      <info cardinality="Multiple">
        <lov hierarchical="false" partial="false" refreshable="true">
          <query>
            <sort order="Ascending"/>
            <search>a*</search>
          </query>
        </lov>
      </info>
    </answer>
  </parameter>
  <parameter dpId="DP0" optional="false" type="prompt">
    <id>1</id>
    <technicalName>psInvoice Date</technicalName>
    <name>Enter Invoice Date:</name>
    <answer constrained="false" type="Date">
      <info cardinality="Single">
        <lov hierarchical="false" refreshable="true">
          <query>
            <sort order="Descending"/>
          </query>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

The PUT call returns the possible values for the parameter of id=0 that correspond to the search pattern in the descending order.
Another **PUT** call with the values for the parameter of **id=0** as input is needed to get the values for the second parameter of **id=1**.

**Request body:**

```xml
<parameters>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>0</id>
    <technicalName>pmEnter First Name of Customer</technicalName>
    <name>Enter First Name of Customer</name>
    <answer type="Text" constrained="false">
      <info cardinality="Multiple">
        <lov refreshable="true" partial="false" hierarchical="false">
          <id>UNIVERSELOV_DS0.DO5a</id>
          <values>
            <value>Annie</value>
            <value>Andre</value>
          </values>
          <columns mappingId="0">
            <column type="String" id="0">First Name </column>
          </columns>
        </lov>
      </info>
    </answer>
  </parameter>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>1</id>
    <technicalName>psInvoice Date</technicalName>
    <name>Enter Invoice Date:</name>
    <answer type="Date" constrained="false">
      <info cardinality="Single">
        <lov refreshable="true" hierarchical="false">
          <id>UNIVERSELOV_DS0.DO6b</id>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```
The last PUT call with the value chosen for the parameter of id=1 as input returns a successful message.

Request body:

```xml
<parameters>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>1</id>
    <technicalName>pmEnter First Name of Customer</technicalName>
    <name>Enter First Name of Customer</name>
    <answer type="Text" constrained="false">
      <info cardinality="Multiple">
        <lov hierarchical="false" refreshable="true">
          <query>
            <sort order="Descending"/>
            <search>a*</search>
          </query>
          <values>
            <value>John Doe</value>
            <value>Jane Smith</value>
            <value>Michael Johnson</value>
          </values>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```
7.3.2.5 Example - Responding to a Hierarchical Parameter

A GET .../parameters call returns a list of values for a specific parameter of type prompt that asks you to select a city. Values are continents.

Response:

```xml
<parameters>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>0</id>
    <technicalName>psSelect a city</technicalName>
    <name>Select a city for customer</name>
    <answer type="Text" constrained="true">
      <info refreshable="true" partial="false" hierarchical="true">
        <id>UNIVERSELOV_DS0.D06a</id>
        <updated>1970-01-01T00:00:00.000+01:00</updated>
        <values>
          <value final="false">Asia</value>
          <value final="false">Europe</value>
          <value final="false">Latin America</value>
          <value final="false">North America</value>
        </values>
        <columns mappingId="0">
          <column type="String" id="0"/>
        </columns>
      </info>
    </answer>
  </parameter>
</parameters>
```
In the first **PUT** call, the value of the first level of the list of values is given as answer to the parameter. The answer is **Europe**.

Request body:

```xml
<parameters>
  <parameter dpId="DP0" optional="false" type="prompt">
    <id>0</id>
    <technicalName>psSelect a city</technicalName>
    <name>Select a city for customer</name>
    <answer constrained="true" type="Text">
      <info cardinality="Single">
        <lov hierarchical="true" partial="false" refreshable="true">
          <query>
            <path>
              <value>Europe</value>
            </path>
          </query>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

The response contains the list of values of the second level of the parameter. Values represent countries.

Response:

```xml
<parameters>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>0</id>
    <technicalName>psSelect a city</technicalName>
    <name>Select a city for customer</name>
    <answer type="Text" constrained="true">
      <info cardinality="Single"/>
      <lov refreshable="true" partial="false" hierarchical="true" path="[[0, \ Europe]]">[0, UNIVERSELOV_DS0.DO6a]</lov>
    </answer>
  </parameter>
</parameters>
```

In a second **PUT** call, the value of the list of values of the second level is given as answer to the parameter (**France**). The first level value is recalled (**Europe**).
Request body:

```xml
<parameters>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>0</id>
    <technicalName>psSelect a city</technicalName>
    <name>Select a city for customer</name>
    <answer type="text" constrained="true">
      <info cardinality="Single">
        <lov hierarchical="true" partial="false" refreshable="true">
          <query>
            <path>
              <value>Europe</value>
              <value>France</value>
            </path>
          </query>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

The response contains the list of values of the third and last level of the parameter. Values represent cities.

Response:

```xml
<parameters>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>0</id>
    <technicalName>psSelect a city</technicalName>
    <name>Select a city for customer</name>
    <answer type="Text" constrained="true">
      <info cardinality="Single">
        <lov hierarchical="true" partial="false" refreshable="true" path="[[0, Europe], [0, France]]">
          <id>UNIVERSELOV_DS0.DO6a</id>
          <updated>1970-01-01T01:00:00.000+01:00</updated>
          <values>
            <value>Marseilles</value>
            <value>Paris</value>
          </values>
          <columns mappingId="0">
            <column type="String" id="0"/>
          </columns>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

In the third and last PUT call, values of the lists of values of the first and second levels, and values of the third level are given as answer to the parameter. Only the value of the third level is considered as actual parameter value (Paris).

Request body:

```xml
<parameters>
  <parameter dpId="DP0" type="prompt" optional="false">
    <id>0</id>
    <technicalName>psSelect a city</technicalName>
    <name>Select a city for customer</name>
    <answer type="text" constrained="true">
      <info cardinality="Single">
        <lov hierarchical="true" partial="false" refreshable="true">
          <query>
          </query>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```
7.3.3 Getting the Details of a Parameter

Usage

Returns the possible answers of a parameter specified by its identifier.

Request

GET /queries/<queryID>/parameters/<parameterID>

Response

Response type: application/xml or application/json

The response provides the parameter with its expected answers. See Parameter Response Body Schemas [page 95] to learn about the content structure and element details.

Example

Parameter of Type Prompt

GET /queries/8963412105615728541/parameters/0

Response:

<parameters>
    <parameter type="prompt" optional="false">
Enter value for Customer Id (End):

<table>
<thead>
<tr>
<th>Interval ID</th>
<th>Date Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1991-2002-02-09T01:00:00.000+01:00</td>
</tr>
<tr>
<td>1</td>
<td>2001-06-15T02:00:00.000+02:00</td>
</tr>
<tr>
<td>2</td>
<td>2001-08-14T02:00:00.000+02:00</td>
</tr>
<tr>
<td>3</td>
<td>2001-11-02T01:00:00.000+01:00</td>
</tr>
<tr>
<td>4</td>
<td>2001-12-14T01:00:00.000+01:00</td>
</tr>
<tr>
<td>5</td>
<td>2002-02-12T01:00:00.000+01:00</td>
</tr>
<tr>
<td>6</td>
<td>2002-04-10T02:00:00.000+02:00</td>
</tr>
<tr>
<td>7</td>
<td>2002-02-09T01:00:00.000+01:00</td>
</tr>
<tr>
<td>8</td>
<td>2002-02-26T01:00:00.000+01:00</td>
</tr>
<tr>
<td>9</td>
<td>2001-12-28T01:00:00.000+01:00</td>
</tr>
</tbody>
</table>

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
Getting the Details of a Parameter Restricted Through a Query

Usage

Returns the possible answers of a parameter of which values are restricted through a query.

Request

PUT /queries/<queryID>/parameters/<parameterID>

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
Request body: the details of a query. See Answer Request Body Schemas [page 104] to learn about the content structure and element details.

Response

Response type: application/xml or application/json

The response provides the parameter with its expected answers.

Example

Parameter of Type Prompt

First, you retrieve the prompt parameter of cardinality Single that accepts an answer made of three columns.

GET /queries/8963412105615728541/parameters

Response:

```xml
<parameters>
  <parameter type="prompt" optional="false">
    <id>0</id>
    <technicalName>psEnd</technicalName>
    <name>Enter value for Customer Id (End):</name>
    <answer type="Numeric" constrained="true">
      <info keepLastValues="true" cardinality="Single">
        <lov mandatorySearch="false" searchable="true" refreshable="true" partial="false" hierarchical="false">
          <id>UNIVERSELOV_DS0.DOa8</id>
          <updated>2015-08-19T17:18:53.000+02:00</updated>
          <intervals>
            <interval id="0">
              <cvalue>
                <column id="0">1991</column>
                <column id="1">2002-02-09T01:00:00.000+01:00</column>
              </cvalue>
              <cvalue>
                <column id="0">1616</column>
                <column id="1">2001-06-15T02:00:00.000+02:00</column>
              </cvalue>
              <cvalue>
                <column id="0">1614</column>
                <column id="1">2001-11-10T01:00:00.000+01:00</column>
              </cvalue>
            </interval>
            <interval id="1">
              <cvalue>
                <column id="0">1139</column>
                <column id="1">2001-09-03T02:00:00.000+02:00</column>
              </cvalue>
              <cvalue>
                <column id="0">2001-11-02T01:00:00.000+01:00</column>
              </cvalue>
            </interval>
          </intervals>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```
<table>
<thead>
<tr>
<th>Customer Id</th>
<th>From Date</th>
<th>To Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1132</td>
<td>2001-12-14T01:00:00.000+01:00</td>
<td>2002-02-12T01:00:00.000+01:00</td>
</tr>
<tr>
<td>718</td>
<td>2002-02-09T01:00:00.000+01:00</td>
<td>2002-04-10T02:00:00.000+02:00</td>
</tr>
<tr>
<td>714</td>
<td>2001-12-28T01:00:00.000+01:00</td>
<td>2002-02-26T01:00:00.000+01:00</td>
</tr>
<tr>
<td>297</td>
<td>2001-10-08T02:00:00.000+02:00</td>
<td>2001-12-07T01:00:00.000+01:00</td>
</tr>
<tr>
<td>293</td>
<td>2001-12-16T01:00:00.000+01:00</td>
<td>2002-02-14T01:00:00.000+01:00</td>
</tr>
<tr>
<td>1</td>
<td>2001-08-09T02:00:00.000+02:00</td>
<td>2001-10-08T02:00:00.000+02:00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer Id</th>
<th>From Date</th>
<th>To Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1991</td>
<td>2002-02-09T01:00:00.000+01:00</td>
<td>2002-04-10T02:00:00.000+02:00</td>
</tr>
<tr>
<td>1988</td>
<td>2001-07-14T02:00:00.000+02:00</td>
<td>2001-09-12T02:00:00.000+02:00</td>
</tr>
</tbody>
</table>

...
Then you retrieve the possible answer values restricted through a query.

PUT /queries/8963412105615728541/parameters/0

Request body:

```
<parameters>
  <parameter optional="false" type="prompt">
    <id>0</id>
    <technicalName>psEnd</technicalName>
    <name>Enter value for Customer Id (End):</name>
    <answer constrained="true" type="Numeric">
      <info cardinality="Single" keepLastValues="true">
        <lov hierarchical="false" mandatorySearch="false" partial="false" refreshable="true" searchable="true">
          <id>UNIVERSELOV_DS0.DOa8</id>
          <updated>2015-08-19T17:18:53.000+02:00</updated>
          <query>
            <sort order="Ascending"/>
            <search>80*</search>
          </query>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

Response:

```
<parameter type="prompt" optional="false">
  <id>0</id>
  <technicalName>psEnd</technicalName>
  <name>Enter value for Customer Id (End):</name>
  <answer type="Numeric" constrained="true">
    <info keepLastValues="true" cardinality="Single">
      <lov hierarchical="false" mandatorySearch="false" partial="false" refreshable="true">
        <id>UNIVERSELOV_DS0.DOa8</id>
        <updated>2015-08-19T17:25:38.000+02:00</updated>
        <cvalues>
          <cvalue>
            <column id="0">802</column>
            <column id="1">2001-12-21T01:00:00.000+01:00</column>
            <column id="2">2002-02-19T01:00:00.000+01:00</column>
          </cvalue>
          <cvalue>
            <column id="0">802</column>
            <column id="1">2000-08-10T02:00:00.000+02:00</column>
            <column id="2">2000-10-09T02:00:00.000+02:00</column>
          </cvalue>
        </cvalues>
      </lov>
    </info>
  </answer>
</parameter>
```
7.4 Managing Query Results

After the query has been posted and its parameters resolved, you can run the query and retrieve the resulting data set using some of the BI Semantic Layer RESTful Web Service APIs. The dataset is returned using an OData format. Several calls to OData allow you to retrieve the resulting data set of the query.

- Accessing the OData Service [page 179]
- Getting the OData Flow Metadata [page 181]
- Getting the OData Flow Content [page 182]
- OData Content - Getting All Data [page 184]
- OData Content - Getting the Row Count [page 186]
- OData Content - Getting the First Row [page 186]
- OData Content - Getting Property Content [page 187]
- OData Content - Getting Property Raw Content [page 187]
- OData Content - Getting Content After Offset [page 187]
- OData Content - Getting the First N Rows [page 189]
- OData Content - Getting the First N Rows After Offset [page 190]

7.4.1 Accessing the OData Service

Usage

Returns the available data flows for the specified query.

Request

GET /queries/<queryID>/data.svc

Response

Response type: application/xml or application/json

The response is an Atom feed or a collection of JSON objects.
Caution

The call returns an error message if at least one context or parameter has not been answered.

Example

One Flow

GET /queries/6089913651317040548/data.svc

Response type: application/xml

```xml
  <workspace>
    <atom:title>Default</atom:title>
    <collection href="Flows0">
      <atom:title>Flows0</atom:title>
    </collection>
  </workspace>
</service>
```

Response type: application/json

```json
{"d": {"EntitySets": ["Flows0"]}}
```

Example

Multiple Flows

GET /queries/6089913651317040730/data.svc

Response type: application/xml

```xml
  <workspace>
    <atom:title>Default</atom:title>
    <collection href="Flows0">
      <atom:title>Flows0</atom:title>
    </collection>
    <collection href="Flows1">
      <atom:title>Flows1</atom:title>
    </collection>
  </workspace>
</service>
```

Response type: application/json

```json
{"d": {"EntitySets": ["Flows0", "Flows1"]}}
```
7.4.2 Getting the OData Flow Metadata

Usage

Gets the metadata used to build the query results that the OData service exposes.

Request

GET /queries/<queryID>/data.svc/$metadata

Response

Response type: application/xml

The response that contains the schema has an EDMX format. The Id property does not map any object. It provides the current row number.

Example

Query with Multiple Flows

GET /queries/6089913651317040730/data.svc/$metadata

```xml
<?xml version="1.0"?>
<edmx:Edmx Version="1.0"
 <edmx:DataServices m:DataServiceVersion="1.0"
 xmlns:sap="http://www.sap.com/Protocols/SAPData">
 <Schema Namespace="Flows"
 xmlns="http://schemas.microsoft.com/ado/2008/09/edm"
 sap:isPartial="false">
 <EntityType Name="Flow0">
 <Key>
 <PropertyRef Name="Id"></PropertyRef>
 </Key>
 <Property Name="Id" Type="Edm.Int32" Nullable="false"></Property>
 <Property Name="City" Type="Edm.String" Nullable="false">
      sap:objectKey="DP0.DO8" sap:qualification="DIMENSION"></Property>
 <Property Name="Customer" Type="Edm.String" Nullable="false">
      sap:objectKey="DP0.DO2" sap:qualification="DIMENSION"></Property>
 ```
Related Information

Getting the List of Queries [page 156]

7.4.3 Getting the OData Flow Content

Usage

Returns the data of the specified flow.

Note

The OData service cannot return data page by page.

Request

GET /queries/<queryID>/data.svc/<flowName>

Response

Response type: application/xml or application/json
The data of a flow is represented by an Atom feed or an array of JSON objects.

Aside from a series of elements generated automatically (title, updated, author, link, and so on), the <feed> has the following children:

- `<id>`, which contains the request URL for the specified flow
- A series of `<entry>`, which represents a row of the flow

Aside from generated elements, an `<entry>` has the following children:

- `<id>`, which contains the request URL for a row specified by `<rowIndex>`(/queries/<queryID>/data.svc/<flowName>(<rowIndex>)). Row indexes are generated automatically.
- `<content>`, which contains the data.

The data appears in the `<m:properties>` tag. This tag has as many children as the query contains metadata. The tag name of each child is `<d:name>`, where name is a label returned by the OData flow metadata. See Getting the OData Flow Metadata [page 181].

### Specifying the Request

You can add several parameters to the request to get particular results. The following table describes the possible requests and the type and meaning of the corresponding responses. These parameters are optional. See examples in the next sections.

<table>
<thead>
<tr>
<th>Request</th>
<th>Response Description</th>
<th>Response Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>/queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;/?$count</td>
<td>The count of records for the specified query flow.</td>
<td>application/text</td>
</tr>
<tr>
<td>/queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;(&lt;rowIndex&gt;)</td>
<td>A row at the <code>&lt;rowIndex&gt;</code> index for the specified query flow. <code>&lt;rowIndex&gt;</code> is an integer.</td>
<td>● application/xml ● application/json</td>
</tr>
<tr>
<td>/queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;(&lt;rowIndex&gt;)/&lt;fieldName&gt;</td>
<td>Information about an object specified by <code>&lt;fieldName&gt;</code> from the row at <code>&lt;rowIndex&gt;</code> index in the specified query flow. <code>&lt;fieldName&gt;</code> corresponds to the Property name.</td>
<td>● application/xml ● application/json</td>
</tr>
<tr>
<td>/queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;(&lt;rowIndex&gt;)/&lt;fieldName&gt;/$value</td>
<td>The raw value of an object specified by <code>&lt;fieldName&gt;</code> from the row at <code>&lt;rowIndex&gt;</code> in the specified query flow.</td>
<td>application/text</td>
</tr>
<tr>
<td>/queries/&lt;queryID&gt;/data.svc/&lt;flowName&gt;/?$skip=&lt;offset&gt;</td>
<td>Query results in chunks, starting with the record number specified by the <code>&lt;offset&gt;</code> position in the specified query flow.</td>
<td>● application/xml ● application/json</td>
</tr>
</tbody>
</table>
### 7.4.4 OData Content - Getting All Data

#### Example

The following request returns all the data that the Flows0 flow contains.

**GET** /queries/6089913651317040730/data.svc/Flows0

**Response type:** application/xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xml:base="http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/">
  <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0/</id>
  <title type="text">Flows0</title>
  <updated>2013-10-15T16:52:16.181+02:00</updated>
  <author>
    <name></name>
  </author>
  <link href="Flows0" rel="self" title="Flows0"></link>
  <entry>
    <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0(0)/</id>
    <title type="text">Flows0</title>
</feed>
```
<updated>2013-10-15T16:52:16.181+02:00</updated>
<category term="Flows.Flow0" scheme="http://schemas.microsoft.com/ado/
2007/08/
dataservices/scheme"></category>
<link href="Flows0(0)" rel="edit" title="Flow0"></link>
<content type="application/xml">
<m:properties>
<d:Id>0</d:Id>
<d:City>Chicago</d:City>
<d:Customer>Steve &amp; Rob</d:Customer>
</m:properties>
</content>
</entry>
<entry>
<id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/
data.svc/Flows0(1)</id>
<title type="text">Flows0</title>
<updated>2013-10-15T16:52:16.182+02:00</updated>
<category term="Flows.Flow0" scheme="http://schemas.microsoft.com/ado/
2007/08/
dataservices/scheme"></category>
<link href="Flows0(1)" rel="edit" title="Flow0"></link>
<content type="application/xml">
<m:properties>
<d:Id>1</d:Id>
<d:City>Chicago</d:City>
<d:Customer>eMarket</d:Customer>
</m:properties>
</content>
</entry>
<entry>
...
</entry>
...
</feed>
Response type: application/json
{"d":[
{"__metadata":
{"uri": "http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/
data.svc/Flow0(0)",
"type": "Flows._Flow0"
},
"Id": 0,
"City": "Chicago",
"Customer": "Steve Rob",
},
{"__metadata":
{"uri": "http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/
data.svc/Flow0(1)",
"type": "Flows._Flow0"
},
"Id": 1,
"City": "Chicago",
"Customer": "eMarket",
},
{
...
},
...
]
}

SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI
Semantic Layer
BI Semantic Layer REST API Reference

© 2015 SAP SE or an SAP affiliate company. All rights reserved.

185


7.4.5 OData Content - Getting the Row Count

Example

The following request returns the numbers of rows in the specified flow. This is also the number of `<entry>` elements in the feed.

GET /queries/6089913651317040730/data.svc/Flow0/$count

Response type: application/text

```
120
```

7.4.6 OData Content - Getting the First Row

Example

The following request returns the data of the first row of the Flows0 flow.

Request:

GET /queries/6089913651317040730/data.svc/Flows0(0)

Response type: application/xml

```
<entry>
  <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0(0)</id>
  <title type="text">Flows0</title>
  <updated>2013-10-15T16:52:16.181+02:00</updated>
  <link href="Flows0(0)" rel="edit" title="Flow0"></link>
  <content type="application/xml">
    <m:properties>
      <d:Id>0</d:Id>
      <d:City>Chicago</d:City>
      <d:Customer>Steve &amp; Rob</d:Customer>
    </m:properties>
  </content>
</entry>
```

Response type: application/json

```
{"d": [  
  {"__metadata": {"uri": "http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0(0)","type": "Flows._Flow0"},
   "Id": 0,
   "City": "Chicago",
   "Customer": "Steve &amp; Rob",
  }
]}
```
7.4.7 OData Content - Getting Property Content

Example
The following request returns the value of the Customer property on the specified row of the Flows0 flow.

GET /queries/6089913651317040730/data.svc/Flows0(3)/Customer

Response type: application/xml

```xml
<?xml version="1.0" encoding="utf-8"?>
```

Response type: application/json

```json
{"d": [{"__metadata":
    "{uri": "http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/
data.svc/Flow0(3)",
    "type": "Flows._Flow0"
    },
    "Customer": "Chicago Mall",
}]
```

7.4.8 OData Content - Getting Property Raw Content

Example
The following request returns the value of the Customer property for the specified row of the Flows0 flow.

GET /queries/6089913651317040730/data.svc/Flows0(3)/Customer/$value

Response type: application/text

Chicago Mall

7.4.9 OData Content - Getting Content After Offset

Example
The following request returns the content of the Flows0 flow starting with the third row.

GET /queries/6089913651317040730/data.svc/Flows0?$skip=2
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
     xml:base="http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/"
     >
  <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0</id>
  <title type="text">Flows0</title>
  <updated>2013-10-15T16:52:17.946+02:00</updated>
  <author>
    <name></name>
  </author>
  <link href="Flows0" rel="self" title="Flows0"></link>
  <entry>
    <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0(2)</id>
    <title type="text">Flows0</title>
    <updated>2013-10-15T16:52:17.947+02:00</updated>
    <link href="Flows0(2)" rel="edit" title="Flow0"></link>
    <content type="application/xml">
      <m:properties>
        <d:Id>2</d:Id>
        <d:City>Chicago</d:City>
        <d:Customer>See You</d:Customer>
      </m:properties>
    </content>
  </entry>
  <entry>
    <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0(3)</id>
    <title type="text">Flows0</title>
    <updated>2013-10-15T16:52:17.947+02:00</updated>
    <link href="Flows0(3)" rel="edit" title="Flow0"></link>
    <content type="application/xml">
      <m:properties>
        <d:Id>3</d:Id>
        <d:City>Chicago</d:City>
        <d:Customer>Chicago Mall</d:Customer>
      </m:properties>
    </content>
  </entry>
  <entry>
    <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0(4)</id>
    <title type="text">Flows0</title>
    <updated>2013-10-15T16:52:17.947+02:00</updated>
    <link href="Flows0(4)" rel="edit" title="Flow0"></link>
    <content type="application/xml">
      <m:properties>
        <d:Id>4</d:Id>
        <d:City>Chicago</d:City>
        <d:Customer>National Supply</d:Customer>
      </m:properties>
    </content>
  </entry>
  ...
</feed>
7.4.10 OData Content - Getting the First N Rows

Example

The following request returns the content of the first four rows of the Flows0.

GET /queries/6089913651317040730/data.svc/Flows0?$top=4

Response type: application/xml

```xml
<feed xmlns="http://www.w3.org/2005/Atom"
     xml:base="http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc"></feed>
```
7.4.11 OData Content - Getting the First N Rows After Offset

Example

The following request returns the content of the first ten rows of the Flows0, starting with the second row.

GET /queries/6089913651317040730/data.svc/Flows0?$skip=1&$top=10

Response type: application/xml

```xml
<feed xmlns="http://www.w3.org/2005/Atom"
      xml:base="http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc">
  <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0</id>
  <title type="text">Flows0</title>
  <updated>2013-10-15T16:52:17.947+02:00</updated>
  <link href="Flows0" rel="edit" title="Flows0"></link>
  <entry>
    <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0(1)</id>
    <title type="text">Flows0</title>
    <updated>2013-10-15T16:52:17.947+02:00</updated>
    <link href="Flows0(1)" rel="edit" title="Flows0"></link>
    <content type="application/xml">
      <m:properties>
        <d:Id>1</d:Id>
        <d:City>Chicago</d:City>
        <d:Customer>See You</d:Customer>
      </m:properties>
    </content>
  </entry>
  <entry>
    <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0(2)</id>
    <title type="text">Flows0</title>
    <updated>2013-10-15T16:52:17.947+02:00</updated>
    <link href="Flows0(2)" rel="edit" title="Flows0"></link>
    <content type="application/xml">
      <m:properties>
        <d:Id>2</d:Id>
        <d:City>Chicago</d:City>
        <d:Customer>See You</d:Customer>
      </m:properties>
    </content>
  </entry>
  <entry>
    <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0(3)</id>
    <title type="text">Flows0</title>
    <updated>2013-10-15T16:52:17.947+02:00</updated>
    <link href="Flows0(3)" rel="edit" title="Flows0"></link>
    <content type="application/xml">
      <m:properties>
        <d:Id>3</d:Id>
        <d:City>Chicago</d:City>
        <d:Customer>Chicago Mall</d:Customer>
      </m:properties>
    </content>
  </entry>
</feed>
```
<title type="text">Flows0</title>  
<updated>2013-10-15T16:52:16.182+02:00</updated>  
2007/08/  
  dataservices/scheme"></category>  
  <link href="Flows0(1)" rel="edit" title="Flow0"></link>  
  <content type="application/xml">  
    <m:properties>  
      <d:Id>1</d:Id>  
      <d:City>Chicago</d:City>  
      <d:Customer>eMarket</d:Customer>  
    </m:properties>  
  </content>  
</entry>  
<entry>  
  <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0(2)</id>  
  <title type="text">Flows0(2)</title>  
  <updated>2013-10-15T16:52:16.182+02:00</updated>  
  2007/08/  
  dataservices/scheme"></category>  
  <link href="Flows0(2)" rel="edit" title="Flow0"></link>  
  <content type="application/xml">  
    <m:properties>  
      <d:Id>2</d:Id>  
      <d:City>Chicago</d:City>  
      <d:Customer>See You</d:Customer>  
    </m:properties>  
  </content>  
</entry>  
...  
<entry>  
  <id>http://w2k8x64sp2:6405/biprws/sl/v1/queries/6089913651317040730/data.svc/Flows0(10)</id>  
  <title type="text">Flows0(10)</title>  
  <updated>2013-10-15T16:52:17.947+02:00</updated>  
  2007/08/  
  dataservices/scheme"></category>  
  <link href="Flows0(10)" rel="edit" title="Flow0"></link>  
  <content type="application/xml">  
    <m:properties>  
      <d:Id>10</d:Id>  
      <d:City>Vancouver</d:City>  
      <d:Customer>Pacific Supermarket</d:Customer>  
    </m:properties>  
  </content>  
</entry>  
</feed>
8  Web Intelligence REST API Reference

Reference of the Web Intelligence REST APIs.

Each API reference provides the following information:

- What the API does
- URL of the HTTP request
- Description of the request query parameters
- Request body if needed (XML or JSON example used)
- Response type, which determines the value of the accept header attribute
- Response body (not described if the response is a success or error message)
- Response examples

Managing Documents [page 192]
Managing Reports [page 288]
Drilling on Report Data [page 331]
Managing Report Elements [page 345]
Managing Data Providers [page 404]
Managing Connections for Free-Hand SQL Data Providers [page 445]
Managing Personal Data Providers [page 448]
Managing SAP BW Connections and BEx Queries [page 455]
Refreshing Documents [page 463]
Refreshing Data Providers [page 489]
Scheduling Documents [page 491]
Searching for Resources [page 499]
Managing Universes with the Web Intelligence RESTful Web Service SDK [page 503]

8.1  Managing Documents

The Web Intelligence RESTful Web Service SDK provides a series of APIs to work with Web Intelligence documents.

⚠️ Restriction
Management of auto-save and auto-recovery configuration is currently not supported.

Creating a Document [page 193]
Getting the List of Documents [page 195]
8.1.1 Creating a Document

Usage

Creates an empty Web Intelligence document.

Request

POST /documents

Request type: application/xml or application/json

Request body:

```
<document>
  <name>
  <folderId>
```
Table 162:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;name&gt;</code></td>
<td>string</td>
<td>The name of the document to be created.</td>
</tr>
<tr>
<td><code>&lt;folderId&gt;</code></td>
<td>integer</td>
<td>The BI launch pad folder where the document is created. You can retrieve the folder ID from a GET call to <code>http://&lt;server-name&gt;:6405/biprws/infostore</code>.</td>
</tr>
</tbody>
</table>

Response

Response type: `application/xml` or `application/json`

The response is a message stating the success or failure of the request.

Example

**XML Example**

POST /documents

Request body:

```xml
<document>
    <name>My Document</name>
    <folderId>5151</folderId>
</document>
```

Response body:

```xml
<success>
    <message>The resource of type "Document" with identifier "5022" has been successfully created.</message>
    <id>5022</id>
</success>
```

An identifier has been assigned automatically.

Example

**JSON Example**

POST /documents

Request body:

```json
{"document": {
    "name": "my document",
    "folderId": "5151"
}}
```

Response body:

```json
{"success": {
    "message": "The resource of type "Document" with identifier "5022" has been successfully created.",
    "id": 5022
}}
```
8.1.2 Getting the List of Documents

Usage

Gets the list of Web Intelligence documents stored in the CMS repository.

The documents are sorted by name. The list returned depends on user access rights. You can also specify the number of documents to return and the first document to be used as the start document in the list.

Note

You can also search for documents using the /searches API.

Request

GET: /documents?offset=<offset>&limit=<limit>

Where:
- `<offset>` indicates the position in the list, from which documents are returned. It must be greater than or equal to 0. The default value is 0. This parameter is optional.
- `<limit>` indicates the number of documents in the list. The range is [0, 50]. The default value is 10. This parameter is optional.

Response

Response type: application/xml or application/json

Response body: the list of Web Intelligence documents sorted by name and identified by the following elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;id&gt;</td>
<td>integer</td>
<td>The document ID</td>
</tr>
<tr>
<td>&lt;cuid&gt;</td>
<td>string</td>
<td>The unique document ID</td>
</tr>
<tr>
<td>&lt;name&gt;</td>
<td>string</td>
<td>The document name</td>
</tr>
<tr>
<td>&lt;description&gt;</td>
<td>string</td>
<td>The optional document description</td>
</tr>
</tbody>
</table>
### Example

**To Get Six Documents from the CMS (XML)**

GET /documents?limit=6

**Response body:**

```xml
<documents>
  <document>
    <id>6804</id>
    <cuid>AbrVz6ES51NMtjZk4KLzPuM</cuid>
    <name>BOF-ROLE_AA_ActiveHierOnCtry - Levels 02 and 03</name>
    <folderId>7611</folderId>
    <state>Original</state>
  </document>
  <document>
    <id>6861</id>
    <cuid>AUwfvPdEG3xOoLNoNEzDguQ</cuid>
    <name>BOF-QRY_SIMPLE - Filter not in Result</name>
    <description>This is a sample for BOF-QRY_SIMPLE - Filter not in Result</description>
    <folderId>7611</folderId>
    <state>Unused</state>
  </document>
  <document>
    <id>6869</id>
    <cuid>Ac1wDFEOuvNHtvI4H.anSM</cuid>
    <name>BOF-QRY_SIMPLE - Filter on Country</name>
    <folderId>48162</folderId>
    <state>Original</state>
  </document>
  <document>
    <id>7048</id>
    <cuid>AUiVpegCKnFBh4AX8YxMT5A</cuid>
    <name>BOF-QRY_SIMPLE - Filter on Prompt not in Result</name>
    <folderId>7611</folderId>
    <state>Modified</state>
  </document>
  <document>
    <id>7125</id>
    <cuid>AWa48YHHT6FiUMP.T5wtBs</cuid>
    <name>BOF-QRY_SIMPLE - Query Prompt not in Result</name>
    <folderId>7611</folderId>
    <state>Unused</state>
  </document>
  <document>
    <id>5121</id>
    <cuid>AAnGKBoLv71DuK_UWMU15Q</cuid>
    <name>Chart demo</name>
    <description>Shows how data can be visualized on different kinds of charts. And the features supported on the charts.</description>
  </document>
</documents>
```
Example

To Get Three Documents from the CMS (JSON)

GET /documents?limit=3

Response body:

```json
{"documents": [ {
  "id": 5152,
  "cuid": "AZx1nlTLmCvJ6WUTC5I",
  "name": "Chart Demo",
  "description": "Shows how data can be visualized on different kinds of charts. And the features supported on the charts.",
  "folderId": 5151,
  "state": "Unused"}, {
  "id": 5169,
  "cuid": "AW4AVT1AUhVAogA6P7OQv9c",
  "name": "Charting Samples",
  "folderId": 5150,
  "state": "Unused"}, {
  "id": 5157,
  "cuid": "AS9ukIRdcIZLuUS6ESGVRBq",
  "name": "Drill Demo",
  "description": "This document demonstrates the drill functionality on tables and Charts",
  "folderId": 5151,
  "state": "Unused"} ]}
```

Related Information

Searching for Resources [page 499]

8.1.3 Getting the Details of a Document

Usage

Gets the details of a Web Intelligence document.
**Request**


Where:

- `<trackerDocumentID>`: identifier of a reference document for trackdata feature. Optional. Must be provided only when the document state is unused.

**Response**

Response type: application/xml or application/json

Response body: the details of the document identified by the following elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;id&gt;</td>
<td>Integer</td>
<td>The document ID</td>
</tr>
<tr>
<td>&lt;cuid&gt;</td>
<td>String</td>
<td>The unique document ID</td>
</tr>
<tr>
<td>&lt;name&gt;</td>
<td>String</td>
<td>The document name</td>
</tr>
<tr>
<td>&lt;description&gt;</td>
<td>String</td>
<td>The document description</td>
</tr>
<tr>
<td>&lt;folderId&gt;</td>
<td>Integer</td>
<td>The identifier of the folder of the CMS repository that contains the document</td>
</tr>
<tr>
<td>&lt;path&gt;</td>
<td>String</td>
<td>The path to the document in the CMS repository directory</td>
</tr>
<tr>
<td>&lt;updated&gt;</td>
<td>DateTime</td>
<td>The date and the time of the last update</td>
</tr>
<tr>
<td>&lt;scheduled&gt;</td>
<td>Boolean</td>
<td>true if the document has been scheduled</td>
</tr>
<tr>
<td>&lt;state&gt;</td>
<td>Unused</td>
<td>Original</td>
</tr>
<tr>
<td></td>
<td>Unused, if the document has not been loaded in the web service container</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Original, if the document has been loaded by the web service but not modified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modified, if the user sent requests which altered the state of the document</td>
<td></td>
</tr>
<tr>
<td>&lt;createdBy&gt;</td>
<td>String</td>
<td>The name of the document creator</td>
</tr>
<tr>
<td>&lt;lastAuthor&gt;</td>
<td>String</td>
<td>The name of the last person who modified the document</td>
</tr>
<tr>
<td>&lt;size&gt;</td>
<td>Integer</td>
<td>The size of the document in bytes</td>
</tr>
<tr>
<td>&lt;refreshOnOpen&gt;</td>
<td>Boolean</td>
<td>true if the document is set to be refreshed at the open time</td>
</tr>
</tbody>
</table>
Example

Getting the Details of a Document (XML)

GET /documents/4958

Response body:

```xml
<document>
  <id>4958</id>
  <cuid>AQtkbb8gNg4N0j3ydf.Sw1lY</cuid>
  <name>Formatting Sample</name>
  <description>This is a sample document showing formatting capabilities.</description>
  <folderId>8246</folderId>
  <path>My Favorites/web Intelligence</path>
  <updated>2012-09-05T14:25:05.277+02:00</updated>
  <scheduled>true</scheduled>
  <state>Unused</state>
  <createdBy>Administrator</createdBy>
  <lastAuthor>User1</lastAuthor>
  <size>29410</size>
  <refreshOnOpen>false</refreshOnOpen>
</document>
```

Example

Getting the Details of a Document (JSON)

GET /documents/5152

```json
{"document":
  "id":5152,
  "cuid":"AZx1nlTlCMdCvyJ6bWUTC5I",
  "name":"Chart Demo",
  "description":"Shows how data can be visualized on different kinds of charts. And the features supported on the charts.",
  "folderId":5151,
  "path":"Public Folders\Web Intelligence Samples\Mobile Samples",
  "updated":"2015-04-21T18:10:29.052+02:00",
  "scheduled":false,
  "state":"Unused",
  "createdBy":"Administrator",
  "lastAuthor":"Administrator",
  "size":298874,
  "refreshOnOpen":false}
}
```

Related Information

Getting the List of Documents [page 195]
8.1.4 Copying a Document

Usage

Copies a Web Intelligence document referenced by its ID.

Request

POST /documents?sourceId=<documentID>

Request type: application/xml or application/json

Request body (optional):

```
<document>
  <name>
  <folderId>
```

Table 165:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;name&gt;</td>
<td>string</td>
<td>The name of the copied document in the BI launch pad folder. If not specified, the service assigns a name to the document automatically.</td>
</tr>
<tr>
<td>&lt;folderId&gt;</td>
<td>integer</td>
<td>The BI launch pad folder where the document is copied. If not specified, the folder is the one of the original document.</td>
</tr>
</tbody>
</table>

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

Copying a Document (XML)

POST /documents?sourceId=4990

Request body:

```
<document>
  <name>Copy of 4990</name>
</document>
```

Response:

```
<success>
  <message>The resource of type "document" with identifier "5875" has been successfully created.</message>
```

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
The name "Copy of 4990" and ID "5875" have been assigned to the copied document. The folder is the same as the original one.

**Example**

**Copying a Document (JSON)**

```
POST /documents?sourceId=5152
```

The original document is named "Chart Demo".

Response:

```
{"success":
    "message":"The resource of type \"Document\" with identifier \"6585\" has been successfully created.",
    "id":6585}
```

The name "Chart Demo[1]" and ID "6585" have been assigned to the copied document. The folder is the same as the original one.

**Related Information**

- [Getting the List of Documents](page 195)

**8.1.5 Saving a Document**

**Usage**

Saves a Web Intelligence document to a folder of the user machine.

When you save a document, it is copied to the destination folder and an identifier is assigned automatically.

**Request**

```
POST /documents/<documentID>
```

Request type: application/xml or application/json

Request body:

```
<document>
    <name>
```

...
Table 166:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;name&gt;</td>
<td>String</td>
<td>The name of the document to save</td>
</tr>
<tr>
<td>&lt;folderId&gt;</td>
<td>Integer</td>
<td>The identifier of the BI launch pad destination folder</td>
</tr>
</tbody>
</table>

Note

Use -1 as folder identifier to save the document to the same folder.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/1234

Request body:

```xml
<document>
  <name>CopiedDocument</name>
  <folderId>2906</folderId>
</document>
```

Response:

```xml
<success>
  <message>The resource of type "Document" has been successfully created.</message>
  <id>5678</id>
</success>
```

An identifier has been assigned automatically.

8.1.6 Updating the State of a Document

Usage

Changes the state of a Web Intelligence document referenced by its ID.
About the State of a Document

The state of a document in the WACS server can be one of the following:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused</td>
<td>An unused document is a document that has not been loaded to the server.</td>
</tr>
<tr>
<td>Original</td>
<td>An original document is a document that has been loaded to the server, but has not been modified. Its state can be changed to Unused to release the available memory, closing the document.</td>
</tr>
<tr>
<td>Modified</td>
<td>A modified document is a document that has been loaded and modified. Its state can be changed to Unused. The document is closed to release the available memory.</td>
</tr>
</tbody>
</table>

As soon as you open a document, its status becomes Original. You can move a document from a Modified or Original to Unused state to discard all document changes and close the document. This releases the memory of the WACS server.

Request

PUT /documents/<documentID>

Request type: application/xml or application/json

Request body:

```xml
<document>
  <state/>
</document>
```

Remember

The request body is optional. If <state> is present, no other tag is accepted.

<table>
<thead>
<tr>
<th>Document State Change</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Original to Unused</td>
<td>The document is not modified and closed.</td>
</tr>
<tr>
<td>From Original with no body or an empty body</td>
<td>The document is not modified.</td>
</tr>
<tr>
<td>From Modified to Unused</td>
<td>The document is updated and closed.</td>
</tr>
<tr>
<td>From Modified with no body or an empty body</td>
<td>The document is updated and saved.</td>
</tr>
</tbody>
</table>
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

To Close an Unmodified Document

PUT /documents/8008

Request body:

```xml
<document>
  <state>Unused</state>
</document>
```

Response body:

```xml
<success>
  <message>The resource of type "document" with identifier "8008" has not been modified.</message>
  <id>8008</id>
</success>
```

Example

To Save a Modified Document

PUT /documents/9326

Response body:

```xml
<success>
  <message>The resource of type "document" with identifier "9326" has been successfully updated.</message>
  <id>9326</id>
</success>
```

Related Information

Getting the List of Documents [page 195]

8.1.7 Deleting a Document

Usage

Deletes a Web Intelligence document referenced by its identifier.
Request

DELETE /documents/<documentID>

Response

Response type: application/xml or application/json
The response is a message stating the success or failure of the request.

Example

DELETE /documents/8022

<success>
<message>The resource of type "Document" with identifier "8022" has been successfully removed.</message>
  <id>8022</id>
</success>

Related Information

Getting the List of Documents [page 195]

8.1.8 Getting the List of Values of a Data Object

Usage

Returns the list of values of a data object of a Web Intelligence document.
A data object is a data provider expression, a variable, or a linked dimension.

Request

GET /documents/<documentID>/dataobjects/<dataObjectID>/lov
Response

Response type: application/xml or application/json

Example

XML

GET /documents/6071/dataobjects/DP0.DO84/lov

Response:

```xml
<lov mandatorySearch="false" searchable="false" refreshable="false" partial="false" hierarchical="false">
  <values>
    <value>Germany</value>
    <value>Japan</value>
    <value>Madagascar</value>
    <value>middle East</value>
    <value>Nepal</value>
    <value>South Africa</value>
    <value>US</value>
  </values>
  <columns mappingId="0">
    <column type="String" id="0">Country</column>
  </columns>
</lov>
```

Example

JSON

GET /documents/5152/dataobjects/DP0.DO1fc/lov

Response:

```json
{"lov":{
  "@mandatorySearch":"false",
  "@searchable":"true",
  "@refreshable":"false",
  "@partial":"false",
  "@hierarchical":"false",
  "values":{
    "value":[
      "2 Pocket shirts",
      "Belts,bags,wallets",
      "Bermudas",
      "Boatwear",
      "Cardigan",
      "Casual dresses",
      "Day wear",
      "Dry wear",
      "Evening wear",
      "Fancy fabric",
      "Full length",
      "Hair accessories",
      "Hats,gloves,scarves",
      "Jackets",
      "Jeans",
      "Jewelry",
      "Long lounge pants",
      "Long sleeve",
      "Lounge wear",
    ]
  }
}}
```
Related Information

Getting the List of Documents [page 195]
Getting the Definition of an Input Control [page 308]

8.1.9 Getting the List of Values Under a Specific Data Object Value

Usage

Returns the list of values under a specified value of a data object in a hierarchy.

A data object is a data provider expression, a variable, or a linked dimension.

Request

PUT /documents/<documentID>/dataobjects/<dataObjectId>/lov

Request body: a specific value of the list of values of a data object.
Caution
Make sure the value you specify in the request body is correct. The Web Service SDK does not make any checking of the value. Incorrect values are ignored and replaced with the root values.

Response

Response type: application/xml or application/json

Example

PUT /documents/6070/dataobjects/DP0.DO9d/lov

Request body:

```xml
<lov hierarchical="true" partial="false" refreshable="false">
  <query>
    <path>
      <value id="\[Customer\].\[Customer Geography\].\[State-Province\].\[NSW\]&\[AU\]" final="false">Queensland</value>
    </path>
  </query>
</lov>
```

Response:

```xml
<lov hierarchical="true" partial="false" refreshable="false">
  <values>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Alexandria\].\[NSW\]" final="false">Alexandria</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Coffs Harbour\].\[NSW\]" final="false">Coffs Harbour</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Darlinghurst\].\[NSW\]" final="false">Darlinghurst</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Goulburn\].\[NSW\]" final="false">Goulburn</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Lane Cove\].\[NSW\]" final="false">Lane Cove</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Lavender Bay\].\[NSW\]" final="false">Lavender Bay</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Malabar\].\[NSW\]" final="false">Malabar</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Matraville\].\[NSW\]" final="false">Matraville</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Milsom Point\].\[NSW\]" final="false">Milsom Point</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Newcastle\].\[NSW\]" final="false">Newcastle</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[North Ryde\].\[NSW\]" final="false">North Ryde</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[North Sydney\].\[NSW\]" final="false">North Sydney</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Port Macquarie\].\[NSW\]" final="false">Port Macquarie</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Rhodes\].\[NSW\]" final="false">Rhodes</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Silverwater\].\[NSW\]" final="false">Silverwater</value>
    <value id="\[Customer\].\[Customer Geography\].\[City\].\[Springwood\].\[NSW\]" final="false">Springwood</value>
  </values>
</lov>
```
Related Information

Getting the List of Documents [page 195]

8.1.10 Working with Document Input Controls

You can apply input controls to one report, a series of reports, or a whole document. You can also move an input control from a report to a document, and vice versa.

The following input controls are supported:

- TextField
- ComboBox
- RadioButtons
- CheckBox
- ListBox
- Calendar
- Spinner
- Slider
- TreeList

Getting the Input Controls of a Document [page 210]
Getting the Definition of a Document Input Control [page 211]
Updating the Definition of a Document Input Control [page 212]
Adding an Input Control to a Document [page 213]
Deleting a Document Input Control [page 214]
Getting the Selection of a Document Input Control [page 215]
Setting the Selection of a Document Input Control [page 216]
Deleting the Selection of a Document Input Control [page 217]
Moving an Input Control from a Report to a Document [page 218]
8.1.10.1 Getting the Input Controls of a Document

Usage

Lists all the input controls of a document.

Request

GET /documents/<documentID>/inputcontrols

Response

Response type: application/xml or application/json
Response body: the list of input control identifiers and names.

Example

XML
GET /documents/6671/inputcontrols
Response:

```
<inputcontrols>
  <inputcontrol>
    <id>D.IF0</id>
    <name>Country</name>
  </inputcontrol>
</inputcontrols>
```

Example

JSON
GET /documents/5152/inputcontrols
Response:

```
{"inputcontrols":
  {"inputcontrol":
    {"id":"D.IF0",
     "name":"my checkbox_1"}
  }
}
```
8.1.10.2 Getting the Definition of a Document Input Control

Usage

Returns the details of an input control of a document.

Request

GET /documents/<documentID>/inputcontrols/<inputControlID>

Response

Response type: application/xml or application/json

Example

**ComboBox (XML)**

GET /documents/6671/inputcontrols/D.IF0

Response

```xml
<inputcontrol>
  <id>D.IF0</id>
  <name>Country</name>
  <comboBox allowAllValuesSelection="true" useCustom="false" operator="Equal"/>
  <assignedDataObject refId="DP0.DO39" kind="Expression"/>
  <assignedReportElements>
    <assignedReportElement refId="D" reference="D.IF0.T0"/>
  </assignedReportElements>
</inputcontrol>
```

**Example

**CheckBox (JSON)**

GET /documents/5152/inputcontrols/D.IF0

Response:

```json
{"inputcontrol":
  {"id":"D.IF0","name":"my checkbox_1",
   "checkBox":
      {"@numberOfLines":"5",
       "@useCustom":"false",
       "@allowAllValuesSelection":"true",
       "@operator":"InList"},
   "assignedDataObject":
      {"@kind":"Expression",
       "@refId":"DP0.DO1fc"},
   ...
  }
}
```
8.1.10.3 Updating the Definition of a Document Input Control

Usage

Updates the details of an input control of a document, such as a name or description.

Request

PUT /documents/<documentID>/inputcontrols/<inputControlID>

Request type: application/xml or application/json

The update of an input control must adhere to the following rules:

- You cannot change the widget of an input control.
- The input control must have at least one assigned report element.
- You can change the assigned data object, but it must be compatible with the widget.
- You can change the operator, but it must be compatible with the widget.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/6671/inputcontrols/D.IF0
8.1.10.4 Adding an Input Control to a Document

Usage

Adds a new input control to a document.

Request

POST /documents/<documentID>/inputcontrols

The request body must adhere to the following rules:

- The widget must be compatible with the assigned data object, for example: slide on a measure, radio buttons on a dimension.
- The widget must be compatible with the operator.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

XML

POST /documents/6671/inputcontrols

Request body:

```xml
<inputcontrol>
  <name>V_update</name>
  <comboBox allowAllValuesSelection="true" useCustom="false"
operator="GreaterOrEqual"/>
  <assignedDataObject refId="DP0.DO2"/>
</inputcontrol>
```

Response:

```xml
<success>
  <message>The resource of type "Input Control" with identifier "D.IF0" has been successfully updated."</message>
  <id>D.IF0</id>
</success>
```
Example

**JSON**

POST /documents/5152/inputcontrols

Request body:

```json
{"inputcontrol":{
    "name":"my checkbox_1",
    "checkBox":{
        "@allowAllValuesSelection":"true",
        "@useCustom":"false",
        "@operator":"InList"},
    "assignedDataObject":{"@refId":"DP0.DO39"}
}}
```

Response:

```json
{"success":{
    "message":"The resource of type "Input Control" with identifier "D.IF1" has been successfully created.",
    "id":"D.IF1"}
}
```

### 8.1.10.5 Deleting a Document Input Control

**Usage**

Removes an input control from a document.

**Request**

DELETE /documents/<documentID>/inputcontrols/<inputControlID>
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/6671/inputcontrols/D.IF0

Response:

<success>
  <message>The resource of type "Input Control" with identifier "D.IF0" has been successfully removed.</message>
  <id>D.IF0</id>
</success>

8.1.10.6 Getting the Selection of a Document Input Control

Usage

Gets the values selected for an input control of a document.

Request

GET /documents/<documentID>/inputcontrols/<inputControlID>/selection

Response

Response type: application/xml or application/json

The response is the list of values selected for the input control.

Example

XML

GET /documents/6671/inputcontrols/D.IF0/selection

Response:

<selection>
  <value>US</value>
</selection>
8.1.10.7 Setting the Selection of a Document Input Control

Usage

Sets the values selected for an input control of a document.

Request

PUT /documents/<documentID>/inputcontrols/<inputControlID>/selection

Request body: the values can be obtained by getting the list of values from the assigned data object using the call GET /documents/<documentID>/dataobjects/<dataObjectID>/lov.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

XML

PUT /documents/6671/inputcontrols/D.IFO/selection

Request body:

<selection>
  <value>France</value>
  <value>Germany</value>
</selection>

Response:

<success>
<message>The resource of type "Input Control Selection" with identifier "D.IF0" has been successfully updated.</message>
</success>

Example

**JSON**

**PUT** /documents/5152/inputcontrols/D.IF0/selection

Request body:

```
{"selection":
  {
    "value":["Bermudas","Boatwear"]
  }
}
```

Response:

```
{"success":
  {
    "message":"The resource of type "Input Control Selection" with identifier "D.IF0" has been successfully updated.",
    "id":"D.IF0"
  }
}
```

Related Information

Getting the List of Values of a Data Object [page 205]

**8.1.10.8 Deleting the Selection of a Document Input Control**

**Usage**

Removes the values selected from an input control of a document.

**Request**

DELETE /documents/<documentID>/inputcontrols/<inputControlID>/selection

**Response**

Response type: application/xml or application/json
The response is a message stating the success or failure of the request.

Example

DELETE /documents/6671/inputcontrols/D.IF0/selection

Response:

<success>
  <message>The resource of type "Input Control Selection" has been successfully removed.</message>
</success>

8.1.10.9 Moving an Input Control from a Report to a Document

Usage

Moves an input control from a report to a document. The input control obtains a new identifier after the move, but keeps the same data objects.

Request

PUT /documents/<documentID>/inputcontrols/<inputControlID>?fromId=<fromID>

Where:

- <fromID> is the input control ID in the report

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT documents/6660/inputcontrols?fromId=R1.IF3

Response:

<success>
  <message>The resource of type "Input Control" with identifier "R1.IF3" has been successfully moved.</message>
  <id>D.IF0</id>
</success>

The <id> is the new identifier of the input control in the document.
Another call to retrieve the definition of the input control shows a fake report element of identifier D associated with the control in the document.

GET documents/6660/inputcontrols/D.IF0

Response:

```xml
<inputcontrol>
  <id>D.IF0</id>
  <name>Revenue</name>
  <slider minValue="30000.0" maxValue="600000.0" increment="1000.0"
    cardinality="Interval" operator="Between">
    <default>
      <value>30000</value>
      <value>600000</value>
    </default>
  </slider>
  <assignedDataObject refId="DP0.DO7" kind="Expression"/>
  <assignedReportElements>
    <assignedReportElement refId="D" reference="D.IF0.T0"/>
  </assignedReportElements>
</inputcontrol>
```

8.1.11 Working with Properties

Use these methods to list or edit the document properties that are visible in the “Document Summary” of a document in Web Intelligence.

- Getting the Properties of a Document [page 219]
- Updating the Properties of a Document [page 221]

8.1.11.1 Getting the Properties of a Document

Usage

Gets the properties of a document referenced by its ID.

Request

GET /documents/<documentID>/properties

Response

Response type: application/xml or application/json
Response body: a series of `<property>` elements.

Example

GET /documents/7858/properties

```xml
<properties>
  <property key="SI_CUID">ATuas0Mb.mxHpfioi7Xrwfo</property>
  <property key="lastrefreshetime">1352999520</property>
  <property key="lastrefreshduration">1</property>
  <property key="lastsavedby">Administrator</property>
  <property key="creatorversion">14.0.6.894</property>
  <property key="disablequerypanel">false</property>
  <property key="lastrefreshdate">2012, Nov 15 18:12:00 GMT+01:00</property>
  <property key="enhancedViewing">false</property>
  <property key="current_doc_id">9939</property>
  <property key="osversion">?</property>
  <property key="modificationtime">1352999598</property>
  <property key="stripquery">true</property>
  <property key="modificationdate">2012, Nov 15 18:13:18 GMT+01:00</property>
  <property key="name">doc_unv_drill_enable</property>
  <property key="createdby">Administrator</property>
  <property key="tdcactivated">false</property>
  <property key="nameinrepo">doc_unv_drill_enable</property>
  <property key="permanentregionalformatting">false</property>
  <property key="reporterversion">14.0.6.894</property>
  <property key="repositorytype">C</property>
  <property key="locale">en_US</property>
  <property key="refreshonopen">false</property>
  <property key="docrepoid">9939</property>
  <property key="documenttype">WID</property>
  <property key="hassamplingresults">false</property>
  <property key="tdcmodeauto">true</property>
  <property key="effectiverefreshonopen">false</property>
  <property key="contentlocale">en_US</property>
  <property key="mergeprompts">true</property>
  <property key="documentsize">31070</property>
  <property key="extendmergedimension">false</property>
  <property key="autorefresh">false</property>
  <property key="creationtime">1352999595</property>
  <property key="creationdate">2012, Nov 15 18:13:15 GMT+01:00</property>
  <property key="ispartiallyrefreshed">false</property>
  <property key="nbqaawsconnection">0</property>
  <property key="documentversion">14.0.6.894</property>
</properties>
```

Related Information

Getting the List of Documents [page 195]
8.1.11.2 Updating the Properties of a Document

Usage

Updates the properties of a Web Intelligence document referenced by its identifier.

> Note

Certain settings are attributed automatically and cannot be set manually (for example, the last refresh time).

Request

PUT /documents/<documentID>/properties

Request body:

```xml
<properties>
  <property key="string">
  </property>
</properties>
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/9939/properties

Request body:

```xml
<properties>
  <property key="refreshonopen">true</property>
</properties>
```

Response:

```xml
<success>
  <message>The resource of type "properties" has been successfully updated.</message>
  <id>9939</id>
</success>
```
Related Information

Getting the List of Documents [page 195]

8.1.12 Working with ZIP files

You can archive the reports of a Web Intelligence document as HTML files into a ZIP file.

- Creating a ZIP File from a Document [page 222]
- Getting a Document as a ZIP file [page 224]
- Deleting a ZIP File [page 225]

8.1.12.1 Creating a ZIP File from a Document

Usage

Creates a ZIP file archiving the document reports of your choice as HTML files.

The file is created on the server.

⚠️ Caution

- You get the ZIP file by calling `GET /documents/<documentID>/zips/<zipID>?fileName=<fileName>`
- The ZIP file is deleted after the user session closes.

Request


Where:

- `<reportID[,...]>` is the list of the report identifiers to be archived in the ZIP file, separated with a comma. This parameter is optional. If not present, all the reports of the document are archived.

Request body: the script that allows the file archiving. See example below.

Response

Response type: `application/xml` or `application/json`
The response is a message stating the success or failure of the request.

**Example**

POST /documents/14695/zips?reportIds=1,3

Request body:

```
<zipfile>
  <zipentry name="index.html">
    <![CDATA[
      <html>
        <head>
          <title>Save As HTML (customized)</title>
          <script language="javascript" src="reports.js"></script>
        </head>
        <body bgcolor="#EEEEEE">
          <div id='content'></div>
          <script>
            var content = document.getElementById('content');
            content.innerHTML = '';
            for (var key in reportIds) {
              var reportLink = "<a href=" + key + "/report.html">";
              reportLink += reportIds[key]['name'];
              reportLink += '</a>'; reportLink += '<br/>';
              content.innerHTML += reportLink; }
          </script>
        </body>
      </html>
    ]]><zipentry>
  <zipentry name="report.html">
    <![CDATA[
      <html>
        <head></head>
        <body></body>
      </html>
    ]]><zipentry>
</zipfile>
```

Response:
```
<success>
  <message>The resource of type "ZipFile" with identifier "0" has been successfully created.</message>
  <id>0</id>
</success>
```

**Related Information**

- Getting the List of Documents [page 195]
- Getting a Document as a ZIP file [page 224]
8.1.12.2 Getting a Document as a ZIP file

Usage

Returns a ZIP file that archives the document reports of your choice as HTML files.

Note

- You can export a document with all of its reports as a ZIP file by calling `GET /documents/<documentID>?fileName=<fileName>`.
- Exports to HTML will be optimized for the end-user browser if you use the User-Agent HTTP header in the REST API call.

Request

GET /documents/<documentID>/zips/<zipID>?fileName=<fileName>

Where:

- `<fileName>` is a string that defines the name of the ZIP file. Maximum string length is 96. This parameter is optional. If set, the HTML output is zipped in a `fileName.zip` file and sent as attachment.

Response

Response type: application/zip

Response: the document as a ZIP file.

Example

GET /documents/14746/zips/0?fileName=myZipFile

Related Information

Getting the List of Documents [page 195]
Exporting a Document in Listing Mode [page 226]
8.1.12.3 Deleting a ZIP File

Usage

Deletes a ZIP file archiving reports of a Web Intelligence document.

Request

DELETE /documents/<documentID>/zips/<zipID>

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/14746/zips/0

Response:

<success>
  <message>The resource of type "ZipFile" with identifier "0" has been successfully removed.</message>
  <id>0</id>
</success>

Related Information

Getting the List of Documents [page 195]

8.1.13 Exporting Documents

You can export either a whole Web Intelligence document or some pages of the document.

  Exporting a Document in Listing Mode [page 226]
  Exporting a Document as a Series of Pages [page 228]
8.1.13.1 Exporting a Document in Listing Mode

Usage

Exports the whole Web Intelligence document to the desired format:

- XML
- zipped HTML
- PDF
- Microsoft Excel 2003
- Microsoft Excel 2007

**i Note**

Exports to HTML will be optimized for the end-user browser if you use the User-Agent HTTP header in the REST API call.

Request

GET /documents/<documentID>?<optional_parameters>

Table 169: Optional Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Supported Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpi</td>
<td>Resolution in dots per inch (dpi) for generated charts. Value between 75 and 9600. Default is 300 for PDF format, 96 for all other formats.</td>
<td>All</td>
</tr>
<tr>
<td>unit</td>
<td>A string that defines the unit that sizes will be reported in. Values are &quot;metric&quot;, &quot;millimeter&quot;, &quot;point&quot;, and &quot;pixel&quot; (default).</td>
<td>XML</td>
</tr>
<tr>
<td>rawValues</td>
<td>Boolean. Default is false. If true, the raw values and their types are exported with the formatted values.</td>
<td>XML</td>
</tr>
<tr>
<td>fileName</td>
<td>A string that defines the name of the ZIP file. If set, the HTML output is zipped to a fileName.zip file. Maximum string length is 96.</td>
<td>ZIP</td>
</tr>
<tr>
<td>optimized</td>
<td>Boolean. Default is false. If true, the generated output is optimized for calculations inside Microsoft Excel.</td>
<td>Microsoft Excel 2003 and Microsoft Excel 2007</td>
</tr>
</tbody>
</table>

Response

Response type:
• text/xml
• application/zip for zipped HTML
• application/pdf
• application/vnd.ms-excel for Microsoft Excel 2003
• application/vnd.openxmlformats-officedocument.spreadsheetml.sheet for Microsoft Excel 2007

The response is the file whose name is the document identifier with the output extension type.

**Example**

**XML**
Exports the XML result in the `xmlfile.xml` with a resolution of 150 dpi for any generated charts.

```
curl -G -s -H "accept:text/xml" -H X-SAP-LogonToken:""%tokenValue%"" 
"<base_webi_REST_URL>/documents/8022?dpi=150&size=pixel" > xmlfile.xml
```

**Example**

**PDF**
Exports the result with the file name `exportpdf.pdf`. The default resolution of 300 dpi for any generated charts is used.

```
"<base_webi_REST_URL>/documents/8022" > exportpdf.pdf
```

**Example**

**Microsoft Excel 2003**
Exports the result with the file name `excel2003.xls` file. The output is optimized for calculations inside Microsoft Excel, and any generated charts will have a resolution of 150 dpi.

```
curl -G -s -H "accept:application/vnd.ms-excel" -H X-SAP-LogonToken:""%tokenValue%""
```

**Example**

**Microsoft Excel 2007**
Exports the result in the `excel2007.xls` file. The output is optimized for calculations inside Microsoft Excel.

```
curl -G -s -H "accept:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
-H X-SAP-LogonToken:""%tokenValue%""
"<base_webi_REST_URL>/documents/8022?optimized=true" > excel2007.xls
```
Related Information

Getting the List of Documents [page 195]

8.1.13.2 Exporting a Document as a Series of Pages

Usage

Exports a Web Intelligence document as a series of pages in one of the following formats:

- XML
- PDF
- Microsoft Excel 2003
- Microsoft Excel 2007

Request

GET /documents/<documentID>/pages?optional_parameters>

Table 170: Optional Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Supported Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpi</td>
<td>Resolution in dots per inch (dpi) for generated charts. Value between 75 and 9600. Default is 300 for PDF format, 96 for all other formats.</td>
<td>All</td>
</tr>
<tr>
<td>mode</td>
<td>normal or quickDisplay.</td>
<td>All</td>
</tr>
<tr>
<td>orientation</td>
<td>Page orientation. Use to force a specific page orientation. Values are portrait and landscape.</td>
<td>If mode=normal</td>
</tr>
<tr>
<td>widthScaling</td>
<td>Number of pages per report displaying in width. The document setting applies by default.</td>
<td>If mode=normal</td>
</tr>
<tr>
<td>heightScaling</td>
<td>Number of pages per report displaying in height. The document setting applies by default.</td>
<td>If mode=normal</td>
</tr>
<tr>
<td>unit</td>
<td>A string that defines the unit that sizes will be reported in. Values are &quot;metric&quot;, &quot;millimeter&quot;, &quot;point&quot;, and &quot;pixel&quot; (default).</td>
<td>XML</td>
</tr>
<tr>
<td>rawValues</td>
<td>Boolean. Default is false. If true, the raw values and their types are exported with the formatted values.</td>
<td>XML</td>
</tr>
</tbody>
</table>
### Response

Response type:

- text/xml
- application/pdf
- application/vnd.ms-excel for Microsoft Excel 2003
- application/vnd.openxmlformats-officedocument.spreadsheetml.sheet for Microsoft Excel 2007

The response is a file whose name is the document identifier with the output extension type.

**Example**

**XML**

Exports the XML result in xmlfile.xml.

```bash
```

**Example**

**PDF**

Exports the PDF result in exportpdffile.pdf.

```bash
```

**Example**

**Microsoft Excel 2003**

Exports the result in excel2003.xls file.

```bash
```
Example

Microsoft Excel 2007
Exports the result in excel2007.xlsx file.

```
curl -G -s -H "accept:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
```

Related Information

Getting the List of Documents [page 195]

8.1.14 Working with Snapshots

A snapshot is identified by a snapshot ID token. A document can have the following states:

- Open
- Set Prompts
- Refresh
- Output

These states are affected by each change of the document (not only in the case of a prompt workflow like above). This means for example:

- Drill workflows
- Turn to
- Report filter manipulation
- Fold/unfold

Storage token can be used in scheduling workflows.

Creating a Document Snapshot [page 230]
Getting the List of Snapshots [page 231]
Restoring a Document to a Snapshot [page 232]

8.1.14.1 Creating a Document Snapshot

Usage

Creates a snapshot of a Web Intelligence document. The call returns the identifier of the newly created storage token.
You can customize one step and go back and forth between initial state and resulting state. This can be used as undo/redo or implement custom prompt dialog boxes.

**Request**

POST /documents/<documentID>/snapshots

**Response**

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

```
Example

POST /documents/8022/snapshots

<success>
  <message>The resource of type "Snapshot" with identifier "we00000000e3667716ec10" has been successfully created.</message>
  <id>we00000000e3667716ec10</id>
</success>
```

**Related Information**

Getting the List of Documents [page 195]

### 8.1.14.2 Getting the List of Snapshots

**Usage**

Gets the list of snapshot identifiers for a specified document.

**Request**

GET /documents/<documentID>/snapshots
Response

Response type: application/xml or application/json

Response body: the list of snapshot IDs for the document. The `id` attribute of `<snapshot>` is the snapshot identifier.

```
GET /documents/8022/snapshots

<snapshots maxStackSize="10">
  <snapshot id="we00000000d447acb133a0"/>
  <snapshot id="we00000000e5df6062ca2a"/>
  <snapshot id="we00010000dc73921d7b5d"/>
</snapshots>
```

Related Information

Getting the List of Documents [page 195]

8.1.14.3 Restoring a Document to a Snapshot

Usage

Restores the document to the state corresponding to the given snapshot.

Request

```
PUT /documents/<!--documentID-->?snapshotId=<!--snapshotID-->
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.
Example

PUT /documents/5022?snapshotId=we0000000e5df6062ca2a

<success>
  <message>The resource of type "Document" with identifier "5022" has been successfully updated.</message>
  <id>5022</id>
</success>

Related Information

Getting the List of Documents [page 195]
Getting the List of Snapshots [page 231]

8.1.15 Managing Styles, Formats, Fonts, Skins, and Charsets

The following APIs allow you to manage the visual aspects of your documents and reports.

Getting the Configuration Formats [page 234]
Getting Custom Formats for Numbers [page 235]
Getting the Font Mappings [page 236]
Getting the Report Skins [page 237]
Getting the Styles of a Document [page 239]
Getting the Details of a Style [page 240]
Adding a Style to a Document [page 241]
Updating a Style [page 242]
Deleting a Style [page 244]
Getting the CSS of a Document [page 244]
Updating the CSS of a Document [page 245]
Getting the Charsets [page 246]
Getting the Chart Types [page 248]
8.1.15.1 Getting the Configuration Formats

Usage

Gets a list of all declared formats on the Web Intelligence Server for the default locale. Returns the list of default formats and available formats.

The results depends on the Preferred Viewing Locale (PVL). If you need a specific locale setting, the locale must be passed in the header of the HTTP request. See Supporting Multiple Languages [page 38].

Request

GET /configuration/formats

Response

Response type: application/xml or application/json

Response body:

```xml
<format default="Boolean" type="Number|Currency|DateTime|Boolean|Date|Time" sample="string">
  <template positive="string"/>
</format>
```

Where:

- **default** is true if it is a default format
- **type** is the type of any format
- **sample** is a sample value of the corresponding format

Example

```xml
<formats>
  <format default="true" type="Number" sample="1 234,57">
    <template positive=""/>
  </format>
  <format default="true" type="Currency" sample="1 234,57 €; -1 234,57 €">
    <template positive="#,##0.00 €"/>
  </format>
  <format default="true" type="DateTime" sample="21/09/2004 20:45:30">
    <template positive="dd'/MM'/yyyy HH':'mm':'ss"/>
  </format>
  <format default="true" type="Boolean" sample="vrai; faux">
    <template positive="BOOLEAN"/>
  </format>
  <format default="true" type="Date" sample="21/09/2004">
    <template positive="dd'/MM'/yyyy"/>
  </format>
  <format default="true" type="Time" sample="20:45:30">
    <template positive="HH':'mm':'ss"/>
  </format>
</formats>
```
8.1.15.2 Getting Custom Formats for Numbers

Usage

Gets a list of the custom format numbers defined in a document.

Request

GET /documents/<documentID>/formats

Response

Response type: application/xml or application/json

Response body: description of the custom formats used in the Web Intelligence document.
Example

GET /documents/4326/formats

```xml
<formats>
  <format type="Custom" sample="1,235">
    <template positive="#,##0" negative="[red]#,##0" zero="No value" undefined="NaN"></template>
  </format>
</formats>
```

In this example, negative numbers are in red.

Related Information

Getting the List of Documents [page 195]

8.1.15.3 Getting the Font Mappings

Usage

Gets a list of all declared font mappings on a Web Intelligence server. There are three available platforms: HTML, Java, and Windows, with different font mappings. This call returns all mappings for all platforms.

Request

GET /configuration/fontmappings

Response

Response type: application/xml or application/json

Response body: the list of all declared font mappings on the Web Intelligence server:

- The `platform` attribute specifies the type of platform that uses the fonts
- The `<servername>` element is the name of the font server
- The `<platformName>` element lists the available fonts. Fonts with more than one word as their name are enclosed in quotes.
8.1.15.4 Getting the Report Skins

Usage

Gets a list of all skins of all types declared on a Web Intelligence Server.

Request

GET /configuration/skins

Response

Response type: application/xml or application/json
Response body: details of the available skins identified by:

```xml
<skin type="Cell|Block|Section|Report">
  <name>
  </name>
  <background>
  </background>
  <color>
  </color>
  <image>
  </image>
</skin>
```

Where:

- `<name>` is the unique name of the skin
- `<background>` is the background fill for the skin: a color, pattern, or a gif URL.

Example

```xml
<skins>
  <skin type="Cell">
    <name>Curve</name>
    <background>
      <color rgb="#ffffff"/>
      <image src="boimg://skin_marble.gif" display="Tile"/>
    </background>
  </skin>
  <skin type="Cell">
    <name>Business Objects</name>
    <background>
      <color rgb="#ffffff"/>
      <image src="boimg://skin_bo.gif" display="Stretch"/>
    </background>
  </skin>
  <skin type="Block">
    <name>Business Objects</name>
    <background>
      <color rgb="#ffffff"/>
      <image src="boimg://skin_bo.gif" display="Stretch"/>
    </background>
  </skin>
  <skin type="Section">
    <name>Business Objects</name>
    <background>
      <color rgb="#ffffff"/>
      <image src="boimg://skin_bo.gif" display="Stretch"/>
    </background>
  </skin>
  <skin type="Section">
    <name>Dots</name>
    <background>
      <color rgb="#ffffff"/>
      <image src="boimg://skin_listing.gif" display="Tile"/>
    </background>
  </skin>
  <skin type="Report">
    <name>Dots</name>
    <background>
      <color rgb="#ffffff"/>
      <image src="boimg://skin_listing.gif" display="Tile"/>
    </background>
  </skin>
</skins>
```
8.1.15.5 Getting the Styles of a Document

Usage

Gets the list of all defined styles in a given document.

Request

```
GET /documents/<documentID>/styles?unit=<unit>
```

Where:
- `<unit>` is an optional parameter of type string that defines the unit of measurement used for all dimensional values such as background width and height. Values are `metric` (default), `inch` and `centimeter`.

Response

Response type: application/xml or application/json


Example

```
GET /documents/3422/styles

<styles>
  <style>
    <id>1</id>
    <background>
      <color rgb="#ffffff"/>
    </background>
    <alignment horizontal="Left" vertical="Bottom"/>
  </style>
  <style>
    <id>2</id>
    <parentID>1</parentID>
    <border>
      <top thickness="None" rgb="#000000" style="None"/>
      <bottom thickness="Thin" rgb="#000000" style="Plain"/>
      <left thickness="None" rgb="#000000" style="None"/>
      <right thickness="None" rgb="#000000" style="None"/>
    </border>
    <font size="12" face="Arial" italic="false" bold="true" strikethrough="false"
      underline="false" rgb="#000000"/>
    <alignment horizontal="Left" vertical="Bottom"/>
  </style>
  <style>
    <id>3</id>
    <parentID>1</parentID>
    <border>
      <top thickness="Thin" rgb="#cac9d9" style="Plain"/>
      <bottom thickness="Thin" rgb="#cac9d9" style="Plain"/>
    </border>
  </style>
</styles>
```
Related Information

Getting the List of Documents [page 195]

8.1.15.6 Getting the Details of a Style

Usage

Gets the description of a style of a Web Intelligence document.

Request

GET /documents/<documentID>/styles/<styleID>?unit=<unit>

Where:

- `<unit>` is an optional parameter of type string that defines the unit of measurement used for all dimensional values such as background width and height. Values are metric (default), inch and centimeter.

Response

Response type: application/xml or application/json

Response body: details of a document style identified by `<style>`

Example

GET /documents/5022/styles/3

<style>
  <id>3</id>
</style>
Related Information

Getting the List of Documents [page 195]
Getting the Styles of a Document [page 239]

8.1.15.7 Adding a Style to a Document

Usage

Adds a style to a Web Intelligence document referenced by its ID.

Request

POST /documents/<documentID>/styles?unit=<unit>

Where:

- `<unit>` is an optional parameter of type string that defines the unit of measurement used for all dimensional values such as background width and height. Values are `metric` (default), `inch`, and `centimeter`.

Request body:

```
<style>
  <border>
    <background>
      <font>
        <alignment>
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/3422/styles?unit=inch

Request body:

```xml
<style>
  <background width="2.3" height="1.6"/>
</style>
```

Response:

```xml
<success>
  <message>The resource of type "Style" with identifier "64" has been successfully created.</message>
  <id>64</id>
</success>
```

Related Information

Getting the List of Documents [page 195]

8.1.15.8  Updating a Style

Usage

Updates a style of a Web Intelligence document.

Request

PUT /documents/<documentID>/styles/<styleID>?unit=<unit>

Where:

- `<unit>` is an optional parameter of type string that defines the unit of measurement used for all dimensional values such as background width and height. Values are metric (default), inch and centimeter.

Request body:

```xml
<style>
```

© 2015 SAP SE or an SAP affiliate company. All rights reserved.

SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer

Web Intelligence REST API Reference
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/5022/styles/24

Request body:

```xml
<style>
  <id>3</id>
  <parentId>58</parentId>
  <border>
    <top thickness="thin" rgb="#cacad9" style="plain"/>
    <bottom thickness="thin" rgb="#cacad9" style="plain"/>
    <left thickness="thin" rgb="#cacad9" style="plain"/>
    <right thickness="thin" rgb="#cacad9" style="plain"/>
  </border>
  <background>
    <color rgb="#5175b9"/>
  </background>
  <font size="9" face="Arial" italic="false" bold="true" strikethrough="false" underline="false" rgb="#ffffff"/>
  <alignment horizontal="left" vertical="bottom"/>
</style>
```

Response:

```xml
<success>
  <message>The resource of type "Style" with identifier "24" has been successfully updated.</message>
  <id>24</id>
</success>
```

Related Information

Getting the List of Documents [page 195]
Getting the Styles of a Document [page 239]
8.1.15.9 Deleting a Style

Usage

Deletes a style from a Web Intelligence document.

Request

```
DELETE /documents/<documentID>/styles/<styleID>
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

```
<success>
  <message>The resource of type "Style" with identifier "L4" has been successfully removed.</message>
  <id>L4</id>
</success>
```

Related Information

- Getting the List of Documents [page 195]
- Getting the Styles of a Document [page 239]

8.1.15.10 Getting the CSS of a Document

Usage

Returns the CSS attached to a Web Intelligence document.
8.15.11 Updating the CSS of a Document

Usage

Updates the CSS attached to a Web Intelligence document.

Request

PUT /documents/<documentID>/css

Request type: text/css
Request body: a CSS file

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

```
```

Request body described in file css2.xml:

```css
FORM CELL.ia-form-header {    /* Settings for header cells in a form */
    background-fill:color;
    font-size:9pt;
    color:#1D7DB3;
    background-color:#f8fbfc;
    font-weight-bold:yes;
    never-alternate:yes;
}
CELL.ia-form-separator {    /* Settings for cells separating two form instances */
    background-fill:none;    /* Invisible cell making the separation */
    border-top-style:none;
    border-right-style:none;
    border-bottom-style:none;
    border-left-style:none;
    border-top-width:0;
    border-right-width:0;
    border-bottom-width:0;
    border-left-width:0;
}
```

Response:

```xml
<success>
  <message>The resource of type "Document" with identifier "5022" has been successfully updated.</message>
  <id>5022</id>
</success>
```

8.1.15.12 Getting the Charsets

Usage

Gets a list of all supported charsets on a Web Intelligence Server. Charset is used as a parameter when exporting a document as CSV format.
Request

GET /configuration/charsets

Response

Response type: application/xml or application/json

Response body: details of the supported charsets, where:

- `<charset>` contains the description of a supported charset
- `<name>` is the charset name
- `<description>` is the charset description

Example

```
<charsets>
  <charset>
    <name>EUC-KR</name>
    <description>Korean (EUC-KR)</description>
  </charset>
  <charset>
    <name>ISO-8859-8</name>
    <description>Hebrew (ISO-8859-8)</description>
  </charset>
  <charset>
    <name>ISO-8859-5</name>
    <description>Cyrillic (ISO-8859-5)</description>
  </charset>
  . . .
  <charset>
    <name>UTF-8</name>
    <description>UTF-8</description>
  </charset>
  <charset>
    <name>HZ-GB-2312</name>
    <description>Chinese Simplified (HZ-GB2312)</description>
  </charset>
  <charset>
    <name>CNS-11643</name>
    <description>Chinese Traditional (EUC-TW)</description>
  </charset>
  <charset>
    <name>CP437</name>
    <description>IBM Latin US (CP437)</description>
  </charset>
</charsets>
```
8.1.15.13 Getting the Chart Types

Usage

Gets the list of available chart types.

Request

GET /configuration/visualizations

Response

Response type: application/xml or application/json

Response body: details of the supported chart type, where:

- <visualization type> contains the description of a supported chart type Tree map, Heat map, Pie chart, Tag cloud and so on.
- <name> is the chart type name
- <description> is the description of the chart type
- <category> describes the category of chart. For example, a Tree map and a Heat map are in the Map category.

Example

```xml
<visualizations>
  
  <visualization type="TreeMap">
    <name>Tree Map</name>
    <description>This chart displays values within nested rectangles that can be colored. The levels of nesting correspond to the level of hierarchical breakdown. The size of the rectangles and their color both express a set of values.</description>
    <category>Map</category>
  </visualization>
  
  <visualization type="HeatMap">
    <name>Heat Map</name>
    <description>This map displays values that are represented by colors in a map using a category axis and optionally a second category axis. The colors of the rectangles are determined by a measure value.</description>
    <category>Map</category>
  </visualization>
  
  <visualization type="Pie">
    <name>Pie Chart</name>
    <description>A circular chart made up of sectors. The area of the circle represents a whole, and the sectors of the circle represent the parts of a whole.</description>
  </visualization>

</visualizations>
```
Related Information

Chart Types [page 58]

8.1.16 Working with Color Palettes

Below are the main tasks that you can perform on color palettes.

A color palette defines a series of colors that can be used in a Web Intelligence document.

You can provide built-in color palettes as in the Web Intelligence user interfaces. You can also provide custom palettes.

- Getting the Default Color Palettes [page 249]
- Getting the List of Custom Palettes [page 251]
- Getting the Details of a Custom Palette [page 252]
- Creating a Custom Palette [page 253]
- Updating the Definition of a Custom Palette [page 254]
- Deleting a Custom Palette [page 255]

8.1.16.1 Getting the Default Color Palettes

Usage

Gets the list of default color palettes. This does not return any custom palettes.

A color is expressed using the RGB color model and an opacity. The RGB color value is the # sign followed by an hexadecimal value. The color opacity value is in the range [0, 255] from opaque to invisible.
Request

GET /configuration/palettes

Response

Response type: application/xml or application/json

Table 172:

<table>
<thead>
<tr>
<th>Response Body</th>
<th>Description</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;palette name=&quot;string&quot;&gt;</td>
<td>The name is the default palette name. The RGB and alpha values define a palette color.</td>
<td>Before 4.1 SP5</td>
</tr>
<tr>
<td>&lt;color rgb=&quot;#...&quot; alpha=&quot;number&quot;&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;palette&gt;</td>
<td>&lt;id&gt;</td>
<td>Since 4.1 SP5</td>
</tr>
<tr>
<td>&lt;id&gt;</td>
<td>&lt;colors&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;color rgb=&quot;#...&quot; alpha=&quot;number&quot;&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

Before 4.1 SP5

```xml
<palletes>
   <palette name="SAP Standard 2011">
       <color rgb="#008fd3" alpha="255"/>
       <color rgb="#99d101" alpha="255"/>
       ...
       <color rgb="#61209a" alpha="255"/>
   </palette>
   ...
   <palette name="black&amp;white">
       <color rgb="#5b5b5b" alpha="255"/>
       <color rgb="#d2d2d2" alpha="255"/>
       ...
       <color rgb="#050505" alpha="255"/>
   </palette>
</palettes>
```

Example

Since 4.1 SP5

```xml
<palletes>
   <palette>
       <id>SAP Standard 2011</id>
       <colors>
           <color rgb="#008fd3" alpha="255"/>
           <color rgb="#99d101" alpha="255"/>
           ...
       </colors>
   </palette>
</palettes>
```
8.16.2 Getting the List of Custom Palettes

**Usage**

Returns the palettes defined by the end-user and attached to a Web Intelligence document.

**Request**

```
GET /documents/<documentID>/palettes
```

**Response**

Response type: application/xml or application/json

The response is the list of custom palettes. A custom palette is defined as follows:

```
<palette>
  <id>
  <name>
  <colors>
    <color>
```

**Example**

```
GET /documents/1234/palettes
```

```
<palettes>
  <palette>
    <id>5f95e34e-b5cl-49d5-ac60-eb73ee5527c2</id>
    <name>custom palette 1</name>
    <colors>
      <color rgb="#5b5b5b" alpha="100"/>
      <color rgb="#d2d2d2" alpha="100"/>
    </colors>
  </palette>
</palettes>
```
8.16.3 Getting the Details of a Custom Palette

Usage

Returns the color information of a custom palette attached to a Web Intelligence document.

Request

GET /documents/<documentID>/palettes/<paletteID>

Response

Response type: application/xml or application/json

Example

GET /documents/1234/palettes/f5d53881-6467-4572-a616-c20bc8d1a687

<palette>
  <id>f5d53881-6467-4572-a616-c20bc8d1a687</id>
  <name>custom palette 3</name>
  <colors>
    <color rgb="#5b5b5b" alpha="100"/>
    <color rgb="#d2d2d2" alpha="100"/>
    <color rgb="#ebeeff" alpha="100"/>
    <color rgb="#848484" alpha="100"/>
    <color rgb="#aeadae" alpha="100"/>
    <color rgb="#0f0f0f" alpha="100"/>
  </colors>
</palette>
Related Information

Getting the List of Documents [page 195]

8.1.16.4 Creating a Custom Palette

Usage

Adds a custom palette to the current Web Intelligence document.

Request

POST /documents/<documentID>/palettes

Request body:

```xml
<palette>
  <name>
    <colors>
      <color>

    ...

  </colors>
</palette>
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request. An ID is assigned to the new custom palette.

Example

POST /documents/1234/palettes

Request body:

```xml
<palette>
  <name>
    <colors>
      <color>

    ...

  </colors>
</palette>
```
8.1.16.5 Updating the Definition of a Custom Palette

Usage

Updates the name and color definitions of a custom palette attached to a Web Intelligence document.

Request

PUT /documents/<documentID>/palettes/<paletteID>

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.
8.1.16.6 Deleting a Custom Palette

Usage

Deletes a custom palette attached to a Web Intelligence document.

Request

DELETE /documents/<documentID>/palettes/<paletteID>

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/1234/palettes/ac468f5c-2035-42ca-8e6f-76700657fa91

8.1.17 Managing Functions, Operators, and Variables

Below are the main tasks you can perform on formula engine functions and operators, and variables.

Getting the Formula Engine Functions [page 256]
8.17.1 Getting the Formula Engine Functions

Usage

Gets all functions of the available formula engine. This can be used to create formulas in the Report Specification or define variables in the document dictionary.

Request

GET /configuration/functions

Response

Response type: application/xml or application/json

Response body: a list of functions, with the following information:

- `<function category="Logical" returnType="Boolean">` Function type
- `<id>`
- `<name>`
- `<description>`
- `<syntax>`

Example

```xml
<functions>
  <function category="Logical" returnType="Boolean">
    <id>EVEN</id>
    <name>Even</name>
    <description>Determines whether a number is even</description>
    <syntax>bool Even(number)</syntax>
  </function>
  <function category="Document" returnType="String">
    <id>DOCUMENTAUTHOR</id>
    <name>DocumentAuthor</name>
    <description>Returns the InfoView logon of the document creator</description>
  </function>
</functions>
```
8.1.17.2 Getting the Formula Engine Operators

Usage

Gets all operators of the formula engine. This can be used to create formulas in the Report Specification or define variables in the document dictionary.

Request

GET /configuration/operators

Response

Response type: application/xml or application/json

Response body: details of the document operators available from the formula engine. Each operator is described as follows:

- <id>
- <name>
- <description>
- <syntax>

Example

```xml
<operators>
  <operator>
    <id>LINEAR</id>
    <name>Linear</name>
    <description>Tells the Interpolation function to use linear regression to supply missing measure values</description>
    <syntax>Linear</syntax>
  </operator>
  <operator>
    <id>INLIST</id>
    <name>InList</name>
    <description>Determines if a value is in a list</description>
    <syntax>bool test_value Inlist(value_list)</syntax>
  </operator>
</operators>
```
8.1.17.3 Getting the Variables of a Document

Usage

Gets the content of a variable dictionary of a Web Intelligence document.

Request

GET /documents/<documentID>/variables

Response

Response type: application/xml or application/json

Response body: a list of variables used by the document, where:

- dataType and qualification attributes of <variable> are the type and qualification of the variable (measure, attribute or dimension)
- grouping="true" attribute is only for grouping variables
- <id> is the identifier of the variable
- <name> is the name of the variable, as used in the document

Example

GET /documents/4326/variables

```
<variables>
    <variable dataType="Numeric" qualification="Measure">
        <id>L6</id>
        <name>Min Revenue</name>
    </variable>
    <variable dataType="Numeric" qualification="Measure">
        <id>L7</id>
        <name>RevenueThreshold</name>
    </variable>
    <variable dataType="Numeric" qualification="Measure">
        <id>L8</id>
        <name>Threshold factor</name>
    </variable>
    <variable dataType="Numeric" qualification="Measure">
        <id>L9</id>
        <name>Threshold Max</name>
    </variable>
    <variable dataType="Numeric" qualification="Measure">
        <id>LA</id>
        <name>Threshold Min</name>
    </variable>
    <variable grouping="true" dataType="String" qualification="Dimension">
        <id>LB</id>
        <name>Month+</name>
    </variable>
</variables>
```
Related Information

Getting the List of Documents [page 195]

8.1.17.4 Getting the Definition of a Variable

Usage

Gets the definition of a variable from the variable dictionary of a Web Intelligence document.

Request

GET /documents/<documentID>/variables/<variableID>

Response

Response type: application/xml or application/json

Response body: definition of the variable, described as follows:

- `dataType` and `qualification` attributes of `<variable>` are the type and qualification of the variable (measure, attribute or dimension)
- `<id>`
- `<name>` is the name of the variable, as used in the document
- `<description>` optional, can be used to describe how the variable is used
- `<formulaLanguageId>`
- `<definition>` is the formula used by the variable

Example

Regular Variable

GET /documents/1234/variables/L9

```
<variable dataType="Numeric" qualification="Measure">
  <id>L9</id>
  <name>Threshold Max</name>
  <description></description>
  <formulaLanguageId>[Threshold Max]</formulaLanguageId>
```
<definition>=\([RevenueThreshold]\)*(1+\[Threshold factor]\))</definition>

Example

**Grouping Variable**

GET /documents/6409/variables/L2

```xml
<variable grouping="true" dataType="String" qualification="Dimension">
  <id>L1</id>
  <name>Month+</name>
  <formulaLanguageId>[Month+]</formulaLanguageId>
  <dimensionId>DP0.DOb9</dimensionId>
  <groups>
    <group>
      <name>Q1</name>
      <values>
        <value>1</value>
        <value>2</value>
        <value>3</value>
      </values>
    </group>
    <group>
      <name>Q3</name>
      <values>
        <value>7</value>
        <value>8</value>
        <value>9</value>
      </values>
    </group>
    <group ungroupedValues="AutomaticallyGrouped">
      <name>Others</name>
      <values>
        <value>4</value>
        <value>5</value>
        <value>6</value>
        <value>10</value>
        <value>11</value>
        <value>12</value>
      </values>
    </group>
  </groups>
</variable>
```

Related Information

Getting the List of Documents [page 195]
Getting the Variables of a Document [page 258]
8.1.17.5 Adding a Variable

Usage

Adds a regular or grouping variable definition to a Web Intelligence document.

Request

POST /documents/<documentID>/variables

Request body in the case of a regular variable:

```
<variable qualification="Measure|Attribute|Dimension">
  <name>
  <definition>
```

Where:
- qualification is the type of variable
- <definition> is the formula used by the variable

➤ Remember

The regular variables must adhere to the following rules:
- The formula must be valid.
- You can only create a measure, an attribute or a dimension.
- When you create an attribute, the associated dimension is mandatory.

Request body in the case of a grouping variable:

```
<variable grouping="true">
  <dimensionId>
  <groups>
    <group>
      <name>
      <values>
        <value>
```

The name of the variable is computed if not provided by the user or in case of name conflict.

➤ Remember

The grouping variables must adhere to the following rules:
- The grouping attribute must be present and set to true.
- The <dimensionId> element should point to a valid attribute or dimension.
- Only one group can have the ungroupedValues attribute. Values assigned to the ungroupedValues group are ignored.
- A group must have a name and contain at least one value, except the ungroupedValues group.
- A value can be assigned to only one group and must belong to the parent list of values.
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

Regular Variable

POST /documents/4326/variables

Request body:

```
<variable qualification="Measure">
  <name>new variable</name>
  <definition>=[RevenueThreshold]*[Threshold factor]</definition>
</variable>
```

Response:

```
<success>
  <message>The resource of type "Variable" with identifier "LB" has been successfully created.</message>
  <id>LB</id>
</success>
```

Example

Grouping Variable

POST /documents/6409/variables

Request body:

```
<variable grouping="true">
  <dimensionId>DP0.DOb9</dimensionId>
  <groups>
    <group>
      <name>From January to April</name>
      <values>
        <value>1</value>
        <value>2</value>
        <value>3</value>
        <value>4</value>
      </values>
    </group>
    <group>
      <name>Summer Holidays</name>
      <values>
        <value>7</value>
        <value>8</value>
      </values>
    </group>
  </groups>
</variable>
```

Response:

```
<success>
  <message>The resource of type "Variable" with identifier "L3" has been successfully created.</message>
</success>
```
8.17.6 Editing a Variable

Usage

Modifies a regular or grouping variable from the variable dictionary of a Web Intelligence document. You can change the qualification, name or definition of a regular variable. You can change the groups and group values of a grouping variable.

Request

PUT /documents/<documentID>/variables/<variableID>

Request body in the case of a regular variable:

```xml
<variable qualification="Measure|Attribute|Dimension">
  <name>
  </name>
  <definition>
</definition>
</variable>
```

👉 Remember

The regular variables must adhere to the following rules:

- The formula must be valid.
- When you change the definition of the variable, you must refresh the document to commit your change to the report.

Request body in the case of a grouping variable:

```xml
<variable grouping="true">
  <groups>
    <group>
      <name>
      </name>
      <values>
        <value>
```

Related Information

- Getting the List of Documents [page 195]
- Getting the List of Values of a Data Object [page 205]
Remember

The grouping variables must adhere to the following rules:

- The grouping attribute must be present and set to true.
- The <dimensionId> element is not required.
- Only one group can have the ungroupedValues attribute. Values assigned to the ungroupedValues group are ignored.
- A group must have a name and contain at least one value, except the ungroupedValues group.
- A value can be assigned to only one group and must belong to the parent list of values.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

Regular Variable

PUT /documents/8022/variables/L9

Request body:

```xml
<variable qualification="Measure">
  <name>Updated variable</name>
  <definition>=([RevenueThreshold]*[Threshold factor])/10</definition>
</variable>
```

Response:

```xml
<success>
  <message>The resource of type "Variable" with identifier "L9" has been successfully updated.</message>
  <id>L9</id>
</success>
```

Example

Grouping Variable

PUT /documents/6409/variables/L1

Request body:

```xml
<variable grouping="true">
  <groups>
    <group>
      <name>First Semester</name>
      <values>
        <value>1</value>
        <value>2</value>
        <value>3</value>
        <value>4</value>
      </values>
    </group>
  </groups>
</variable>
```
Response:

<br/>

**Related Information**

- Getting the List of Documents [page 195](#)
- Getting the List of Values of a Data Object [page 205](#)
- Getting the Variables of a Document [page 258](#)

### 8.1.17.7 Deleting a Variable

**Usage**

Deletes a variable from a documents variable dictionary.

Ensure that the variable is no longer used by the queries in the document.

**Request**

```
DELETE /documents/<documentID>/variables/<variableID>
```
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/8022/variables/L9

```xml
<success>
  <message>The resource of type "Variable" with identifier "L9" has been successfully deleted.</message>
  <id>L9</id>
</success>
```

8.1.18 Managing Links

Note

Links refer to merged dimensions, which is the term used in SAP BusinessObjects Web Intelligence.

To create or modify a link, the following conditions must be respected:

- You can only link dimensions, attributes and hierarchies.
- The expressions to link must have the same data type.
- Each expression that you link to must be from a different data provider.
- The expressions that you link to must not be already used in another link.
- You cannot link to expressions that have been stripped (no query stripping allowed).
- The expressions must be compatible.

About compatibility:

- All custom formulas and variables will be considered as compatible to any other expressions and then ignored.
- Measures are always compatible with everything.
- If dimension A is linked, then A is compatible with all expressions in the flow lists that contain the link.
- Two dimensions or attributes from different flow lists are incompatible. Except when a dimension is linked, as mentioned above.

Below are the main tasks you can perform on the links of a report.

- Getting the Links of a Document [page 267]
- Getting the Details of a Link [page 268]
- Adding a Link [page 269]
- Editing a Link [page 270]
- Deleting a Link [page 271]
8.18.1 Getting the Links of a Document

Usage

Gets the content of a document's links dictionary.

Request

GET /documents/<documentID>/links

Response

Response type: application/xml or application/json

Response body: a series of <link> identified by <id> and <name>.

Example

GET /documents/8022/links

```xml
<link dataType="String" qualification="Dimension">
  <id>LB</id>
  <name>Category</name>
</link>
<link dataType="String" qualification="Dimension">
  <id>L8</id>
  <name>City</name>
</link>
... 
<link dataType="String" qualification="Dimension">
  <id>L6</id>
  <name>Year</name>
</link>
</links>
```

Related Information

Getting the List of Documents [page 195]
8.18.2 Getting the Details of a Link

Usage

Gets the description of a link of a document.

Request

GET /documents/<documentID>/links/<linkID>

Response

Response type: application/xml or application/json

Response body: a link described as follows:

- <id>
- <name>
- <description>
- <linkedExpressions>

Example

```
GET /documents/7738/links/L6

<link dataType="String" qualification="Dimension">
  <id>L6</id>
  <name>Year</name>
  <description>Year 2003 - 2006.</description>
  <dataSourceObjectId>DS0.DObc</dataSourceObjectId>
  <formulaLanguageId>[Year]</formulaLanguageId>
  <linkedExpressions>
    <linkedExpression id="DP0.DObc"/>
    <linkedExpression id="DP1.DObc"/>
  </linkedExpressions>
</link>
```

Related Information

Getting the List of Documents [page 195]
Getting the Links of a Document [page 267]
8.1.18.3 Adding a Link

Usage

Adds a link to the documents expressions dictionary.

Request

POST /documents/<documentID>/links

Request body:

```xml
<link>
  <name>new link</name>
  <linkedExpressions>
    <linkedExpression id="string"/>
  </linkedExpressions>
</link>
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/3422/links

Request body:

```xml
<link>
  <name>new link</name>
  <linkedExpressions>
    <linkedExpression id="DP0.DObc"/>
    <linkedExpression id="DP1.DObc"/>
  </linkedExpressions>
</link>
```

Response:

```xml
<success>
  <message>The resource of type "Link" with identifier "L6" has been successfully created.</message>
  <id>L6</id>
</success>
```
Related Information

Getting the List of Documents [page 195]

8.1.18.4 Editing a Link

Usage

Modifies a link of a Web Intelligence document.

Request

PUT /documents/<documentID>/links/<linkID>

Request body:

```xml
<link>
  <id />
  <name />
  <description />
  <dataSourceObjectId />
  <formulaLanguageId />
  <linkedExpressions>
    <linkedExpression id="string"/>
  </linkedExpressions>
</link>
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/3422/links/L2

Request body:

```xml
<link dataType="String" qualification="Dimension">
  <id>L6</id>
  <name>Year</name>
  <description>Year 2003 - 2006.</description>
  <dataSourceObjectId>DS0.DObc</dataSourceObjectId>
  <formulaLanguageId>[Year]</formulaLanguageId>
  <linkedExpressions>
    <linkedExpression id="DP0.DObc"/>
    <linkedExpression id="DP1.DObc"/>
  </linkedExpressions>
</link>
```
Response:

```xml
<success>
  <message>The resource of type "Link" with identifier "L2" has been successfully updated.</message>
  <id>67</id>
</success>
```

### Related Information

- Getting the List of Documents [page 195]
- Getting the Links of a Document [page 267]

### 8.1.18.5 Deleting a Link

#### Usage

Deletes a document referenced by its ID.

#### Request

```
DELETE /documents/<documentID>/links/<linkID>
```

#### Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

#### Example

```
DELETE /documents/8022/links/L4
```

```xml
<success>
  <message>The resource of type "Link" with identifier "L4" has been successfully removed.</message>
  <id>L4</id>
</success>
```
8.1.19 Managing Alerters and Tracker Settings

Alerters

You use the alerters to track the changes in selected data and to configure the display of changed data. Use the formula language to build your custom alerters for formatting data changes. You can include special calculations based on data changes. For example, you can include a calculation to show the difference between the previous value and the current value of a measure.

Trackdata

When you track data changes, you select a particular data refresh as a reference point. This data is known as the reference data. When you activate the data tracking, you see your data in relation to the reference data.

Formatting to track changes either automatically or manually. You can track the following types of data change:

- Inserted data
- Deleted data
- Changed data
- Increased values
- Decreased values

In automatic data tracking mode, you always compare the current data with the data before the last refresh. This is achieved by automatically setting the current data as the reference data just before each refresh. The reference data is always one refresh behind the current data. Automatic data tracking is appropriate for scheduled documents when you want to compare the current data with the data before the last refresh.

In manual data tracking mode, you select the reference data. You continue to use this data as a reference point until you update the reference point.

- Getting the List of Alerters [page 273]
- Getting the Details of an Alerter [page 274]
- Adding an Alerter [page 275]
- Editing an Alerter [page 277]
- Deleting an Alerter [page 278]
- Getting the Tracker Settings [page 279]
- Creating the Tracker Settings [page 280]
- Editing the Tracker Settings [page 282]
8.1.19.1 Getting the List of Alerters

Usage

Gets the list of all alerters defined in a document.

Request

GET /documents/<documentID>/alerters

Response

Response type: application/xml or application/json

Response body: a list of alerters, with the following information:

- <id>
- <name>
- <description> (optional)

Example

GET /documents/1223/alerters

<alerters>
    <alerter>
        <id>1</id>
        <name>Sales Revenue</name>
        <description>Test 1</description>
    </alerter>
    <alerter>
        <id>2</id>
        <name>Profit Margin</name>
        <description>Test 2</description>
    </alerter>
</alerters>

Related Information

Getting the List of Documents [page 195]
8.1.19.2 Getting the Details of an Alerter

Usage

Gets the full description of an alerter of a Web Intelligence document.

Request

GET /documents/<documentID>/alerters/<alerterID>

Response

Response type: application/xml or application/json

Response body: details of the document alerter identified by:

- <id>
- <name>
- <description> (optional)
- <rule>, which is the definition of the rule
- <action>, which is the definition of the affect on the document formatting

Example

GET /documents/8022/alerters/3

<alerter>
  <id>1</id>
  <name>Sales Revenue</name>
  <description>Test Raylight</description>
  <rule>
    <conditions>
      <condition expressionId="DP0.DO93" operator="Greater">
        <operand>2000000</operand>
      </condition>
    </conditions>
    <action>
      <data>
        <formula type="HyperLink">test</formula>
        <format type="Custom">
          <template positive="STANDARD"/>
        </format>
      </data>
      <style>
        <background>
          <color rgb="#ffff00"/>
        </background>
        <font rgb="#ff0000"/>
      </style>
    </action>
  </rule>
</alerter>
Related Information

Getting the List of Documents [page 195]
Getting the List of Alerters [page 273]

8.1.19.3 Adding an Alerter

Usage

Adds an alerter to a Web Intelligence document.

Request

POST /documents/<documentID>/alerters

Request body:

```xml
<alerter>
  <name>
   ...
  </name>
  <description>
   ...
  </description>
  <rule>
   ...
  </rule>
  <action>
   ...
  </action>
</alerter>
```

Note

You must provide an action when creating a new alerter. An empty alerter will result in an error.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/1223/alerters

Request body:

```xml
<name>ciomplexalerter</name>
```
<description>Add a complex alerter using Raylight</description>

<rule>
  <conditions>
    <condition expressionId="DP0.DO93" operator="Greater">
      <operand>10</operand>
      <operand>20</operand>
    </condition>
    <condition expressionId="DP0.DO93" operator="Greater">
      <target expressionId="DP0.DO93" />
    </condition>
  </conditions>
  <action>
    <data>
      <formula type="HyperLink">my hyperlink</formula>
      <format type="Custom" sample="1 234,57">
        <template positive="STANDARD"/>
      </format>
    </data>
    <style>
      <border>
        <top thickness="Medium" rgb="#800000" style="Plain"/>
        <bottom thickness="Medium" rgb="#800000" style="Plain"/>
        <left thickness="Medium" rgb="#800000" style="Plain"/>
        <right thickness="Medium" rgb="#800000" style="Plain"/>
      </border>
      <background width="0" height="0">
        <color rgb="#00ff00"/>
        <image src="bores://00002">
          <alignment horizontal="Left" vertical="Top"/>
        </image>
      </background>
      <font size="12" face="Arial" italic="false" bold="false" strikethrough="true" underline="true" rgb="#ff6600"/>
      <alignment horizontal="Center" vertical="Center"/>
    </style>
  </action>
</rule>

<i>Note</i>
The font size is expressed in "point".

Response:

<success>
  <message>The resource of type "Alerter" with identifier "2" has been successfully created.</message>
  <id>2</id>
</success>

Related Information

Getting the List of Documents [page 195]
Getting the List of Alerters [page 273]
8.1.19.4 Editing an Alerter

Usage

Updates the definition of an alerter of a Web Intelligence document.

Request

PUT /documents/<documentID>/alerters/<alerterID>

Request body:

```xml
<alerter>
  <id>
    <name>
      <description>Add a complex alerter</description>
      <rule>
        <action>
          <data>
            <formula type="HyperLink">my hyperlink</formula>
            <format type="Custom" sample="1 234,57">template positive="STANDARD"/></format>
          </data>
          <style>
            <border>
              <top thickness="Medium" rgb="#800000" style="Plain"/>
            </border>
          </style>
        </action>
      </rule>
    </name>
  </id>
</alerter>
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/5022/alerters/2

Request body:

```xml
<alerter>
  <name>Sample 2</name>
  <description>Add a complex alerter</description>
  <rule>
    <condition expressionId="DP0.DO93" operator="Greater">
      <operand>20</operand>
    </condition>
    <condition expressionId="DP0.DO93" operator="Greater">
      <target expressionId="DP0.DO93" />
    </condition>
  </rule>
  <action>
    <data>
      <formula type="HyperLink">my hyperlink</formula>
      <format type="Custom" sample="1 234,57">template positive="STANDARD"/></format>
    </data>
    <style>
      <border>
        <top thickness="Medium" rgb="#800000" style="Plain"/>
      </border>
    </style>
  </action>
</alerter>
```
Response:

```
<success>
  <message>The resource of type "Alerter" with identifier "2" has been successfully updated.</message>
  <id>2</id>
</success>
```

Related Information

- Getting the List of Documents [page 195]
- Getting the List of Alerters [page 273]

8.1.19.5 Deleting an Alerter

Usage

Deletes an alerter.

Request

DELETE /documents/<documentID>/alerters/<alerterID>
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/5022/alers/3

<success>
  <message>The resource of type "Alerter" with identifier "3" has been successfully removed.</message>
  <id>3</id>
</success>

Related Information

Getting the List of Documents [page 195]
Getting the List of Alerters [page 273]

8.1.19.6 Getting the Tracker Settings

Usage

Obtains the current trackdata definition for a document. Returns an error if the trackdata function is not activated on this document.

Request

GET /documents/<documentID>/tracker

Response

Response type: application/xml or application/json

Response body: definition of the trackdata information, described as follows:

- The mode attribute of <tracker> is the mode of Track Data changes:
  - Auto: display changes are based on comparison with last data refresh
- **Manual**: display changes are based on comparison with data refresh from user reference
- The **threshold** attribute specifies the display changes threshold in percent for numeric data. The attribute type is double.

**Example**

```xml
GET /documents/8022/tracker

<tracker mode="Manual" referenceDate="2012-11-16T10:56:21.951+01:00">
  <added active="true">
    <background rgb="#0000aa"/>
    <style italic="false" bold="true" strikethrough="false" underline="false" rgb="#aabbcc"/>
  </added>
  <changed active="true">
    <background rgb="#aabbcc"/>
    <style italic="false" bold="true" strikethrough="false" underline="false" rgb="#0000bb"/>
  </changed>
  <removed active="true">
    <background rgb="#0000cc"/>
    <style italic="false" bold="true" strikethrough="false" underline="false" rgb="#0000bb"/>
  </removed>
  <increasing threshold="5.0" applyThreshold="false" active="true">
    <background rgb="#33cc33"/>
    <style italic="false" bold="true" strikethrough="false" underline="false"/>
  </increasing>
  <decreasing threshold="8.0" applyThreshold="true" active="false">
    <background rgb="#33cc33"/>
    <style italic="true" bold="false" strikethrough="true" underline="true"/>
  </decreasing>
</tracker>
```

**Related Information**

- [Getting the List of Documents](#)

**8.1.19.7 Creating the Tracker Settings**

**Usage**

Activates a trackdata function on a document.

The request body is optional. If the request body is not provided, the trackdata will be activated with default settings or the previous settings if the trackdata function has been activated earlier.
Request

POST /documents/<documentID>/tracker

Request body:

```xml
<tracker mode="Manual" referenceDate="DateTime">
  <added>
    <changed>
      <removed>
        <increasing>
          <decreasing>
          </increasing>
          <decreasing>
        </removed>
      </changed>
    </added>
  </tracker>
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/8022/tracker

Request body:

```xml
<tracker mode="Manual" referenceDate="2012-11-16T10:56:21.951+01:00">
  <added active="true">
    <background rgb="#0000aa" />
    <style italic="false" bold="true" strikethrough="false" underline="false" />
  </added>
  <changed active="true">
    <background rgb="#aabbcc" />
    <style italic="false" bold="true" strikethrough="false" underline="false" />
  </changed>
  <removed active="true">
    <background />
    <style italic="false" bold="true" strikethrough="false" underline="false" />
  </removed>
  <increasing threshold="5.0" applyThreshold="false" active="true">
    <background rgb="#33cc33" />
    <style italic="false" bold="true" strikethrough="false" underline="false" />
  </increasing>
  <decreasing threshold="8.0" applyThreshold="true" active="false">
    <background rgb="#33cc33" />
    <style italic="true" bold="false" strikethrough="true" underline="true" />
  </decreasing>
</tracker>
```

Response:

```xml
<success>
  <message> The resource of type "Tracker" has been successfully created for the document.</message>
  <id>9917</id>
</success>
```
8.1.19.8 Editing the Tracker Settings

Usage

Updates a trackdata function on a document.

It updates the style applied on alerters. It is also possible to enable/disable the visualization of a change type. If the track data definition is not provided, the mode is set to Manual and the document is set as reference.

The request body is optional. If not provided, the trackdata will be activated with default settings or the previous settings if trackdata has been activated previously.

Request

PUT /documents/<documentID>/tracker

Request body:

```xml
<tracker mode="Manual" referenceDate="DateTime">
  <added>
  <changed>
  <removed>
  <increasing>
  <decreasing>
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/8022/tracker
Request body in an XML file:

```
<tracker mode="Manual" referenceDate="2012-11-16T10:56:21.951+01:00">
  <added active="true">
    <style italic="false" bold="true" strikethrough="false" underline="false" rgb="#0000aa"/>
  </added>
  <changed active="true">
    <background rgb="#aabbcc"/>
    <style italic="false" bold="true" strikethrough="false" underline="false" rgb="#0000bb"/>
  </changed>
  <removed active="true">
    <style italic="false" bold="true" strikethrough="false" underline="false" rgb="#0000cc"/>
  </removed>
</tracker>
```

Response:

```
<success>
  <message> The resource of type "Tracker" has been successfully updated for the document. </message>
  <id>9917</id>
</success>
```

**Related Information**

- Getting the List of Documents [page 195]
- Getting the Tracker Settings [page 279]

### 8.1.19.9 Deleting the Tracker Settings

**Usage**

Deletes the trackdata function on the document.

**Request**

```
DELETE /documents/<documentID>/tracker
```
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/5022/tracker

```xml
<success>
  <message>The resource of type "Tracker" has been successfully removed.</message>
  <id>9917</id>
</success>
```

Related Information

Getting the List of Documents [page 195]

8.1.20 Managing Attachments

Below are the tasks you can perform on the attachments of a report.

- Getting the List of Attachments [page 284]
- Adding an Attachment [page 285]

8.1.20.1 Getting the List of Attachments

Usage

Gets the list of attachments to a document.

Request

GET /documents/<documentID>/attachments
Response

Response type: application/xml or application/json

Response body: details of the attachments of the document. Each attachment is identified by:

- The attachment mime type `<mimeType>`
- Document name `<name>`
- Document description (if one exists, otherwise the tag is empty) `<description>`

Example

GET /documents/8022/attachments

```
<attachments>
  <attachment>
    <name>barometer.png</name>
    <size>13229</size>
    <releasemode>auto</releasemode>
    <md5hashcode>97B3E3B2745595A2CBA42CA825CCD656</md5hashcode>
    <mimeType>image/png</mimeType>
  </attachment>
  <attachment>
    <name>statistics.jpg</name>
    <size>18239</size>
    <releasemode>auto</releasemode>
    <md5hashcode>97B3E3B2745595A2CBA42CA825CCD656</md5hashcode>
    <mimeType>image/jpeg</mimeType>
  </attachment>
</attachments>
```

Related Information

- Getting the List of Documents [page 195]

8.1.20.2 Adding an Attachment

Usage

Adds an attachment to a document.

Request

POST /documents/<documentID>/attachments

Request type: multipart/form-data
Request body:

```
--------------------####boundary123456798
Content-Disposition: form-data; name="attachmentInfos"
Content-Type: application/xml
<attachment>
  <name>sales_report</name>
  <mimeType>image/png</mimeType> <!--Optional-->  
  <size>123</size>
</attachment>
--------------------####boundary123456798
Content-Disposition: form-data; name="attachmentContent";
filename="logo_picture.png"
Content-Type: image/png
(Content of file)
--------------------####boundary123456798
```

The multipart body request allows at least two parts:

- The information about the attachment (attachmentInfos)
- The attachment content (attachmentContent)

In case of image upload, the mime types accepted are the following:

- image/png
- image/jpeg
- image/gif
- image/bmp

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

CURL command:

```
curl -i -X "POST" -H "accept:application/xml"
-H "content-type:multipart/form-data" -H X-SAP-LogonToken:"%tokenValue%" -F
"attachmentInfos=attachment.xml" -F "attachmentContent=<file_to_be_uploaded>"
http://<serverName>:6405/biprws/raylight/v1/documents/3422/attachments
```

Request body:

```
<attachment>
  <mimeType>image/png</mimeType>
  <name>myResourceName</name>
  <description>myResourceDescription</description>
</attachment>
```

Response:

```
<success>
```
<message>The resource of type "Attachment" has been successfully created.</message>

<id>9994</id>
</success>

Related Information

Getting the List of Documents [page 195]

8.1.21 Adding a Cache Entry to a Document

Usage

Add a cache entry to a document.

Request

POST /documents/<documentID>/cache

Request body:

```xml
<cache>
  <output mimeType="application/pdf">
    <locale value="fr_FR"/>
    <locale value="de_DE"/>
  </output>
  <output mimeType="application/pdf">
    <locale value="en"/>
  </output>
</cache>
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/3422/cache

Request body:

```xml
<cache>
</cache>
```
8.2 Managing Reports

Below are the main operations available on reports.

⚠️ Restriction

The following workflows are not supported:

- Retrieving and setting fold/unfold state of a document report
- Retrieving and setting collapse/expand state of a hierarchy displayed in a document report
- Text search in a document or report

Creating a Report [page 289]
Getting the List of Reports [page 290]
Getting the Details of a Report [page 291]
Moving a Report [page 293]
Copying a Report [page 294]
Updating the Properties of a Report [page 295]
Deleting a Report [page 296]
Getting the Map of a Report [page 297]
Exporting Reports [page 298]
Working with Input Controls of a Report [page 306]
Getting the Structure of a Report [page 319]
Updating the Structure of a Report [page 320]
Managing Data Filters in Reports [page 325]
8.2.1 Creating a Report

Usage

Creates a report in the given Web Intelligence document.

Request

POST /documents/<documentID>/reports

Request body (optional):

```xml
<report>
  <name/>
</report>
```

Where:

- `<name>` is a string that specifies the name of the report to be created. If not specified, the service assigns a name to the report automatically.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

**Example**

**XML**

POST /documents/12782/reports

Response:

```xml
<success>
  <message>The resource of type "Report" with identifier "2" has been successfully created.</message>
  <id>2</id>
</success>
```

The name "Report 2" and the ID "2" have been assigned automatically.

**Example**

**JSON**

POST /documents/5152/reports
Request:

```
{"report":
  {"name":"Chart Report"}
}
```

Response:

```
{"success":
  {
    "message":"The resource of type \"Report\" with identifier \"6\" has been successfully created.\",
    "id":6
  }
}
```

The ID "6" has been assigned automatically.

---

**Related Information**

*Getting the List of Documents [page 195]*

---

### 8.2.2 Getting the List of Reports

**Usage**

Gets the list of reports for a Web Intelligence document.

The following items identify a report:

- Id
- Reference

**Request**

```
GET /documents/<documentID>/reports
```

**Response**

*Response type:* application/xml or application/json

*Response body:*

- The `<id>`, `<name>`, and `<reference>` of the reports
• The `<showDataChanges>` Boolean value that expresses whether data changes are displayed

**Example**

**XML**

GET /documents/12782/reports

Response:

```xml
<reports>
  <report>
    <id>1</id>
    <name>Cross Tab View</name>
    <reference>1.RS</reference>
    <showDataChanges>false</showDataChanges>
  </report>
  <report>
    <id>2</id>
    <name>Simple charting view</name>
    <reference>2.RS</reference>
    <showDataChanges>false</showDataChanges>
  </report>
</reports>
```

**Example**

**JSON**

GET /documents/5152/reports

Response:

```json
{
  "reports": {
    "report": [ {
      "id":1,"name":"Basic Charts","reference":"1.RS","showDataChanges":false}, {
      "id":2,"name":"Combination Chart","reference":"2.RS","showDataChanges":false}, {
      "id":3,"name":"Custom Color Palette","reference":"3.RS","showDataChanges":false}, {
      "id":4,"name":"Chart Property Support","reference":"4.RS","showDataChanges":false}, {
      "id":5,"name":"Element Linking","reference":"5.RS","showDataChanges":false} ]
  }
}
```

### 8.2.3 Getting the Details of a Report

**Usage**

Gets the details of the report specified by the given identifier.
Request
GET /documents/<documentID>/reports/<reportID>

Response

Response type: application/xml or application/json

Response body:
- The <id>, <name>, and <reference> of the reports
- The <showDataChanges> Boolean value that expresses whether data changes are displayed
- The report style and page settings if defined

Example

XML
GET /documents/7858/reports/1

Response:

```xml
<report>
    <id>1</id>
    <name>Revenue by City</name>
    <reference>1.RS</reference>
    <showDataChanges>false</showDataChanges>
    <style>
        <hyperLinkColors visited="#000000" link="#0000ff" hover="#000000"
active="#000000"/>
    </style>
    <pageSettings>
        <margins left="2835" right="2835" top="2835" bottom="2835"/>
        <format orientation="Portrait" height="42094" width="29764"/>
        <records vertical="100" horizontal="20"/>
        <scaling factor="100"/>
    </pageSettings>
</report>
```

Example

JSON
GET /documents/5152/reports/5

Response:

```json
{"report":
    {"id":5,
     "name":"Element Linking",
     "reference":"5.RS",
     "showDataChanges":false,
     "style":
     {"hyperLinkColors":
     {"@visited":"#000000","@link":"#0000ff","@hover":"#000000","@active":"#000000"}
     },
     },
```
8.2.4 Moving a Report

Usage

Moves a report within a Web Intelligence document.

Request

PUT /documents/<documentID>/reports?fromId=<fromID>&toId=<toID>

Where:

- `<fromID>` is a valid report identifier (before the move)
- `<toID>` is a valid report identifier (after the move)

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/12782/reports?fromId=1&toId=3

Response:

<success>
8.2.5 Copying a Report

Usage

Copies a report in a given Web Intelligence document.

Request

POST /documents/<documentID>/reports?fromId=<fromID>&keepDriller=<keepDriller>

Where:
- `<fromID>` is a valid identifier of a report
- `<keepDriller>` preserves the driller mode in the copied report if true (default). It is only applicable if the source report is in driller mode.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/14308/reports?fromId=2&keepDriller=false

Response:

<success>
  <message>The resource of type "Report" with identifier "2" has been successfully created.</message>
  <id>2</id>
</success>
8.2.6 Updating the Properties of a Report

Usage

Updates the properties of a report in the given Web Intelligence document.

Request

PUT /documents/<documentID>/reports/<reportID>

Request body:

```xml
<report>
  <name/>
  <showDataChanges/>
</report>
```

You can modify the report name and the `<showDataChanges>` value.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/9512/reports/67

Request body (XML file):

```xml
<report>
  <name>New report name</name>
  <showDataChanges>true</showDataChanges>
</report>
```
Response:

```xml
<success>
  <message>The resource of type "Report" with identifier "67" has been successfully updated.</message>
  <id="67"></id>
</success>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]

8.2.7 Deleting a Report

Usage

Deletes a report from the given Web Intelligence document.

➤ Remember

You cannot undo this operation.

Request

DELETE /documents/<documentID>/reports/<reportID>

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/7858/reports/9372

```xml
<success>
  <message>The resource of type "report" with identifier "9372" has been successfully removed.</message>
  <id>9372</id>
</success>
```
Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]

8.2.8 Getting the Map of a Report

Usage

Gets the map of a report in a Web Intelligence document.
The map of a report is the list of all the instances of a report element in the document.

Request

GET /documents/<documentID>/reports/<reportID>/map?reference=<nodeReference>

Where:

- <nodeReference> is a valid reference of a map node (optional)

Response

Response type: application/xml or application/json

Response body: the node references of the report map.

Example

Getting the Full Map References of a Report

GET /documents/18809/reports/1/map

Response:

```xml
<map>
  <node reference="1.G.0" name="2004"/>
  <node reference="1.G.1" name="2005"/>
  <node reference="1.G.2" name="2006"/>
</map>
```
Example

Getting the Map Reference of a Node of a Report

Once you have obtained the list of node references for a report, you can use this example to get a specific map reference for a node.

GET /documents/18809/reports/1/map?reference=1.0.1

Response:

```xml
<map>
  <node reference="1.5.1.0" name="California"/>
  <node reference="1.5.1.1" name="Colorado"/>
  <node reference="1.5.1.2" name="DC"/>
  <node reference="1.5.1.3" name="Florida"/>
  <node reference="1.5.1.4" name="Illinois"/>
  <node reference="1.5.1.5" name="Massachusetts"/>
  <node reference="1.5.1.6" name="New York"/>
  <node reference="1.5.1.7" name="Texas"/>
</map>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]

8.2.9 Exporting Reports

You can export:

- An entire report as a single document
- A report in paginated mode (one file per report page)
- One page of a report

Exporting a Report in Listing Mode [page 299]
Exporting a Report as a Series of Pages [page 301]
Exporting a Page of a Report [page 303]
8.2.9.1 Exporting a Report in Listing Mode

Usage

Exports the report in one of the following formats:

- HTML
- Zipped HTML
- MHTML (multipart HTML)
- XML
- PDF
- Microsoft Excel 2003
- Microsoft Excel 2007
- CSV

Note

- If you choose the HTML format, the web service generates the image links. Therefore, the logon token must still be valid when the HTML output is displayed.
- Exports to HTML will be optimized for your browser if you use the User-Agent HTTP header in the REST API call.

Request

GET /documents/<documentID>/reports/<reportID>?<optional_parameters>

Table 173: Optional Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Supported Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpi</td>
<td>Resolution in dots per inch (dpi) for generated charts. Value between 75 and 9600. Default is 300 for PDF format, 96 for all other formats.</td>
<td>All, except XML and CSV</td>
</tr>
<tr>
<td>chartOutputFormat</td>
<td>Output format for generated chart. Values are: jpeg, bmp, gif, and png (default).</td>
<td>HTML, ZIP, and MHTML</td>
</tr>
<tr>
<td>imageUrl</td>
<td>String used to customize image links when they cannot be reached from the information system.</td>
<td>HTML</td>
</tr>
<tr>
<td>fileName</td>
<td>A string that defines the name of the ZIP file. If set, the HTML output is zipped to a fileName.zip file. Maximum string length is 96.</td>
<td>ZIP</td>
</tr>
<tr>
<td>unit</td>
<td>A string that defines the unit that sizes will be reported in. Values are &quot;metric&quot;, &quot;millimeter&quot;, &quot;point&quot;, and &quot;pixel&quot; (default).</td>
<td>XML</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Supported Formats</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>rawValues</td>
<td>Boolean. Default is false. If true, the raw values and their types are exported with the formatted values.</td>
<td>XML</td>
</tr>
<tr>
<td>optimized</td>
<td>Boolean. Default is false. If true, the generated output is optimized for calculations inside Microsoft Excel.</td>
<td>Microsoft Excel 2003 and Microsoft Excel 2007</td>
</tr>
<tr>
<td>textQualifier</td>
<td>Character used to surround each column value. Values are `` or `.</td>
<td>CSV</td>
</tr>
<tr>
<td>columnDelimiter</td>
<td>String that defines a character put between columns. Values are comma (,), semi-colon (;) or the special string Tab.</td>
<td>CSV</td>
</tr>
<tr>
<td>charset</td>
<td>String that defines a valid server charset retrieved from the list of charsets.</td>
<td>CSV</td>
</tr>
</tbody>
</table>

### Customization of Image Links

You may need to customize the source of an image in a report exported as HTML if the image is unreachable by your system:

```html
<img border="0" style="position:absolute;top:0px;left:0px;width:400px;height:300px;"
    X-SAP-LogonToken={X-SAP-LogonToken}">
</img>
```

To do this, use the `imageUrl` parameter with an appropriate value in the GET call:


The resulting image link looks like:

```html
<img border="0" style="position:absolute;top:0px;left:0px;width:400px;height:300px;"
    alt="3D Column Chart" src="http://myServer/myPage.jsp?documentId=6406&reportId=1&imageId=dxXMLDraft.drilloff_1*2*6&X-SAP-LogonToken={X-SAP-LogonToken}">
</img>
```

### Response

Response type:
- text/html for HTML
- application/zip for zipped HTML
- multipart/related for MHTML
- text/xml
- application/pdf
8.2.9.2 Exporting a Report as a Series of Pages

Usage

Exports a report of a Web Intelligence document as a series of pages in one of the following formats:

- XML
- PDF
- Microsoft Excel 2003
- Microsoft Excel 2007

Request

GET /documents/<documentID>/reports/<reportID>/pages?<optional_parameters>
### Table 174: Optional Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Supported Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpi</td>
<td>Resolution in dots per inch (dpi) for generated charts. Value between 75 and 9600. Default is 300 for PDF format, 96 for all other formats.</td>
<td>All</td>
</tr>
<tr>
<td>mode</td>
<td>normal (default) or quickDisplay.</td>
<td>All</td>
</tr>
<tr>
<td>unit</td>
<td>A string that defines the unit that sizes will be reported in. Values are &quot;metric&quot;, &quot;millimeter&quot;, &quot;point&quot;, and &quot;pixel&quot; (default).</td>
<td>XML</td>
</tr>
<tr>
<td>rawValues</td>
<td>Boolean. Default is false. If true, the raw values and their types are exported with the formatted values.</td>
<td>XML</td>
</tr>
<tr>
<td>optimized</td>
<td>Boolean. Default is false. If true, the generated output is optimized for calculations inside Microsoft Excel.</td>
<td>Microsoft Excel 2003 and Microsoft Excel 2007</td>
</tr>
<tr>
<td>orientation</td>
<td>Page orientation. Use to force a specific page orientation. Values are portrait and landscape.</td>
<td>If mode=normal</td>
</tr>
<tr>
<td>widthScaling</td>
<td>Number of pages per report displaying in width. The document setting applies by default.</td>
<td>If mode=normal</td>
</tr>
<tr>
<td>heightScaling</td>
<td>Number of pages per report displaying in height. The document setting applies by default.</td>
<td>If mode=normal</td>
</tr>
</tbody>
</table>

### Response

**Response type:**
- text/xml
- application/pdf
- application/vnd.ms-excel for Microsoft Excel 2003
- application/vnd.openxmlformats-officedocument.spreadsheetml.sheet for Microsoft Excel 2007

**Response body:** the report in the expected format.

#### Example

**XML**

```bash
```
8.2.9.3 Exporting a Page of a Report

Usage

Exports one given page of a report in one of the following formats:

- HTML
- Zipped HTML
- MHTML (multipart HTML)
- XML
- PDF
- Microsoft Excel 2003
- Microsoft Excel 2007
Note

- If you choose the HTML format, the web service generates the image links. Therefore, the logon token must still be valid when the HTML output is displayed.
- Exports to HTML will be optimized for your browser if you use the User-Agent HTTP header in the REST API call.

Request

GET /documents/<documentID>/reports/<reportID>/pages/<pageIndex>?<optional_parameters>

Where:

- `<pageIndex>` is the number of the page to export. It begins with 1 and while the number of report pages is not known in advance, you have to use this URL in a loop, which returns the 404 error message "Not Found" after the last found page.

Remember

To export the last page of the report, use GET /documents/<documentID>/reports/<reportID>/pages/last.

Table 175: Optional Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Supported Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpi</td>
<td>Resolution in dots per inch (dpi) for generated charts. Value between 75 and 9600. Default is 300 for PDF format, 96 for all other formats.</td>
<td>All</td>
</tr>
<tr>
<td>mode</td>
<td>normal (default) or quickDisplay.</td>
<td>All</td>
</tr>
<tr>
<td>orientation</td>
<td>Page orientation. Used to force a specific page orientation. Values are portrait and landscape.</td>
<td>If mode=normal</td>
</tr>
<tr>
<td>widthScaling</td>
<td>Number of pages per report displaying in width. The document setting applies by default.</td>
<td>If mode=normal</td>
</tr>
<tr>
<td>heightScaling</td>
<td>Number of pages per report displaying in height. The document setting applies by default.</td>
<td>If mode=normal</td>
</tr>
<tr>
<td>chartOutputFormat</td>
<td>Output format for generated chart. Values are: jpeg, bmp, gif, and png (default).</td>
<td>HTML, ZIP and MHTML</td>
</tr>
<tr>
<td>imageUrl</td>
<td>String used to customize image links when they cannot be reached from the information system.</td>
<td>HTML</td>
</tr>
<tr>
<td>fileName</td>
<td>A string that defines the name of the ZIP file. If set, the HTML output is zipped to a fileName.zip file. Maximum string length is 96.</td>
<td>ZIP</td>
</tr>
</tbody>
</table>
### Customization of Image Links

You may need to customize the source of an image in a page exported as HTML if the image is unreachable by your system:

```html
<img border="0" style="position:absolute;top:0px;left:0px;width:400px;height:300px;"
    X-SAP-LogonToken={X-SAP-LogonToken}">
</img>
```

To do this, use the `imageUrl` parameter with an appropriate value in the GET call:

```
%2FmyPage.jsp
```

The resulting image link looks like:

```html
<img border="0" style="position:absolute;top:0px;left:0px;width:400px;height:300px;"
    alt="3D Column Chart" src="http://myServer/myPage.jsp?documentId=6406&reportId=
    1&pageId=1&imageId=dxXMLDraft.drilloff_1*2*6&X-SAP-LogonToken={X-SAP-
    LogonToken}">
</img>
```

### Response

**Response type:**
- text/html
- application/zip for zipped HTML
- multi-part/related for MHTML
- text/xml
- application/pdf
- application/vnd.ms-excel for Microsoft Excel 2003
- application/vnd.openxmlformats-officedocument.spreadsheetml.sheet for Microsoft Excel 2007

**Response body:** the report in the expected format.
8.2.10 Working with Input Controls of a Report

You can apply input controls to one report, a series of reports, or a whole document. You can also move an input control from a report to a document, and vice versa.

The following input controls are supported:

- TextField
- ComboBox
- RadioButtons
- CheckBox
- ListBox
- Calendar
- Spinner
- Slider
- TreeList

Getting the Input Controls of a Report [page 307]
8.2.10.1 Getting the Input Controls of a Report

Usage

Lists all the input controls of a report.

Request

GET /documents/<documentID>/reports/<reportID>/inputcontrols

Response

Response type: application/xml or application/json

Response body: the list of input control identifiers and names.

Example

GET /documents/6452/reports/1/inputcontrols

Response:

```xml
<inputcontrols>
  <inputcontrol>
    <id>R1.IF0</id>
    <name>Country</name>
  </inputcontrol>
  <inputcontrol>
    <id>R1.IF1</id>
    <name>Service Line</name>
  </inputcontrol>
  <inputcontrol>
    <id>R1.IF2</id>
    <name>Year</name>
  </inputcontrol>
</inputcontrols>
```
8.2.10.2 Getting the Definition of an Input Control

Usage

Returns the details of an input control of a report.

Request

GET /documents/<documentID>/reports/<reportID>/inputcontrols/<inputControlID>

Response

Response type: application/xml or application/json

Example

Calendar

```xml
<inputcontrol>
  <id>R2.IF3</id>
  <name>Invoice Date (type:DateTime)</name>
  <calendar useCustom="false" operator="Equal"/>
  <assignedDataObject refId="DP2.DO17" kind="Expression"/>
  <assignedReportElements>
    <assignedReportElement refId="1" reference="R2.IF3.T0"/>
  </assignedReportElements>
</inputcontrol>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
Example

Check Box

```
<inputcontrol>
  <id>R1.IF0</id>
  <name>Country</name>
  <checkBox allowAllValuesSelection="true" useCustom="false" numberOfLines="5"
    operator="InList">
    <default>
      <value>France</value>
      <value>US</value>
    </default>
  </checkBox>
  <assignedDataObject refId="DP0.DO6" kind="Expression"/>
  <assignedReportElements>
    <assignedReportElement refId="15" reference="R1.IF0.T0"/>
  </assignedReportElements>
</inputcontrol>
```

Example

Combo Box

```
<inputcontrol>
  <id>R2.IF2</id>
  <name>var1_serviceline</name>
  <comboBox allowAllValuesSelection="true" useCustom="false" operator="Equal"/>
  <assignedDataObject refId="L1" kind="Variable"/>
  <assignedReportElements>
    <assignedReportElement refId="1" reference="R2.IF2.T0"/>
  </assignedReportElements>
</inputcontrol>
```

Example

List Box

```
<inputcontrol>
  <id>R1.IF1</id>
  <name>Service Line</name>
  <listBox allowAllValuesSelection="true" useCustom="false" numberOfLines="5"
    cardinality="Multiple" operator="InList">
    <default>
      <value>Accommodation</value>
      <value>Food &amp; Drinks</value>
    </default>
  </listBox>
  <assignedDataObject refId="DP0.DO4" kind="Expression"/>
  <assignedReportElements>
    <assignedReportElement refId="15" reference="R1.IF1.T0"/>
  </assignedReportElements>
</inputcontrol>
```
Example
Radio Button

```xml
<inputcontrol>
  <id>R1.IF2</id>
  <name>Year</name>
  <radioButtons allowAllValuesSelection="true" useCustom="false"
    numberOfLines="5" operator="Equal"/>
  <assignedDataObject refId="DP0.DO18" kind="Expression"/>
  <assignedReportElements>
    <assignedReportElement refId="15" reference="R1.IF2.T0"/>
  </assignedReportElements>
</inputcontrol>
```

Example
Slider

```xml
<inputcontrol>
  <id>R1.IF3</id>
  <name>Revenue</name>
  <slider minValue="30000" maxValue="600000" increment="1000"
    cardinality="Interval" operator="Between">
    <default>
      <value>30000</value>
      <value>600000</value>
    </default>
  </slider>
  <assignedDataObject refId="DP0.DO7" kind="Expression"/>
  <assignedReportElements>
    <assignedReportElement refId="15" reference="R1.IF3.T0"/>
  </assignedReportElements>
</inputcontrol>
```

Example
Spinner

```xml
<inputcontrol>
  <id>R1.IF5</id>
  <name>var2_revenue</name>
  <spinner minValue="10" maxValue="15" increment="1" operator="Equal">
    <default>
      <value>11</value>
    </default>
  </spinner>
  <assignedDataObject refId="L2" kind="Variable"/>
  <assignedReportElements>
    <assignedReportElement refId="20" reference="R1.IF5.T0"/>
  </assignedReportElements>
</inputcontrol>
```

Example
Text Field

```xml
<inputcontrol>
  <id>R2.IF1</id>
  <name>Year</name>
</inputcontrol>
```
Example

Tree List

```xml
<inputcontrol>
  <id>R2.IF4</id>
  <name>Customer Geography</name>
  <treeList numberOfLines="5" cardinality="Multiple"
            allowComplexSelection="true" operator="InList"/>
  <assignedDataObject refId="DP1.DO9d" kind="Expression"/>
  <assignedReportElements>
    <assignedReportElement refId="1" reference="R2.IF4.T0"/>
  </assignedReportElements>
</inputcontrol>
```

Related Information

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]
- Getting the Input Controls of a Report [page 307]

8.2.10.3 Updating the Definition of an Input Control

Usage

Updates the details of an input control of a report.

Request

```
PUT /documents/<documentID>/reports/<reportID>/inputcontrols/<inputControlID>
```

The update of an input control must adhere to the following rules:

- You cannot change the widget of an input control.
- The input control must have at least one assigned report element.
• You can change the assigned data object, but it must be compatible with the widget.
• You can change the operator, but it must be compatible with the widget.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/7214/reports/1/inputcontrols/R1.IF2

Request body:

```xml
<inputcontrol>
  <name>IC Label</name>
  <radioButtons allowAllValuesSelection="true" useCustom="false"
       numberOfLines="5" operator="NotEqual"/>
  <assignedDataObject refId="DP0.DO4"/>
  <assignedReportElements>
    <assignedReportElement refId="2"/>
  </assignedReportElements>
</inputcontrol>
```

Response:

```xml
<success>
  <message>The resource of type "InputControl" with identifier "R1.IF2" has been successfully updated.</message>
  <id>R1.IF2</id>
</success>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
Getting the Input Controls of a Report [page 307]

8.2.10.4 Adding an Input Control

Usage

Adds a new input control to a report.
Request

POST /documents/<documentID>/reports/<reportID>/inputcontrols

The request body must adhere to the following rules:
- The widget must be compatible with the assigned data object, for example: slide on a measure, radio buttons on a dimension.
- The input control must have at least one assigned report element.
- The widget must be compatible with the operator.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/7216/reports/3/inputcontrols

Request body:

```xml
<inputcontrol>
  <name>Slider on Revenue</name>
  <slider minValue="30000" maxValue="600000" increment="1000"
            cardinality="Single" operator="Greater"/>
  <assignedDataObject refId="DP0.DO7"/>
  <assignedReportElements>
    <assignedReportElement refId="2"/>
  </assignedReportElements>
</inputcontrol>
```

Response:

```xml
<success>
  <message>The resource of type "InputControl" with identifier "R3.IF6" has been successfully created.</message>
  <id>R3.IF6</id>
</success>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
8.2.10.5 Deleting an Input Control

Usage

Removes an input control from a report.

Request

DELETE /documents/<documentID>/reports/<reportID>/inputcontrols/<inputControlID>

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/7232/reports/2/inputcontrols/R2.IF5

Response:

<success>
  <message>The resource of type "InputControl" with identifier "R2.IF5" has been successfully removed.</message>
  <id>R2.IF5</id>
</success>

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
Getting the Input Controls of a Report [page 307]

8.2.10.6 Getting the Selection of an Input Control

Usage

Gets the values selected for an input control of a report.
Request

GET /documents/<documentID>/reports/<reportID>/inputcontrols/<inputControlID>/selection

Response

Response type: application/xml or application/json

The response is the list of values selected for the input control.

Example

GET /documents/7246/reports/1/inputcontrols/R1.IF4/selection

Response:

<selection>
  <value>Japan</value>
  <value>UK</value>
  <value>Australia</value>
</selection>

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
Getting the Input Controls of a Report [page 307]

8.2.10.7 Setting the Selection of an Input Control

Usage

Sets the values selected for an input control of a report.

Request

PUT /documents/<documentID>/reports/<reportID>/inputcontrols/<inputControlID>/selection

Request body: the values can be obtained by getting the list of values from the assigned data object.
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/7246/reports/1/inputcontrols/R1.IF4/selection

Request body:

```xml
<selection>
    <value>Japan</value>
    <value>UK</value>
    <value>Australia</value>
</selection>
```

Response:

```xml
<success>
    <message>The resource of type "Input Control Selection" with identifier "R1.IF4" has been successfully updated.</message>
    <id>R1.IF4</id>
</success>
```

Related Information

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]
- Getting the Input Controls of a Report [page 307]
- Getting the List of Values of a Data Object [page 205]

8.2.10.8 Deleting the Selection of an Input Control

Usage

Removes the values selected of an input control of a report.

Request

DELETE /documents/<documentID>/reports/<reportID>/inputcontrols/<inputControlID>/selection
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/7255/reports/1/inputcontrols/R1.IF4/selection

Response:

<success>
  <message>The resource of type "InputControlSelection" has been successfully removed.</message>
</success>

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
Getting the Input Controls of a Report [page 307]

8.2.10.9 Moving an Input Control from a Document to a Report

Usage

Moves an input control from a document to a report. The input control obtains a new identifier after the move, but keeps the same data objects.

You can also specify the report elements to associate with the input control in the request body.

Request

PUT /documents/<documentID>/reports/<reportID>/inputcontrols?fromId=<fromID>

Where:

- <fromID> is the input control ID in the document

Request body:

<inputcontrol>
  <assignedReportElements>
    <assignedReportElement refId="string"/>
  </assignedReportElements>
</inputcontrol>
The request body is optional.
You can retrieve the report element identifiers (refId) from the list of report elements (<id>):

```
GET documents/<documentID>/reports/<reportID>/elements
```

Response:

```
<elements>
  <element>
    <id>
```

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

Moving an Input Control without Assigned Report Element

```
PUT /documents/6671/reports/2/inputcontrols?fromId=D.IF0
```

Response:

```
<success>
  <message>The resource of type "Input Control" with identifier "D.IF0" has been successfully moved.</message>
  <id>R2.IF1</id>
</success>
```

The <id> is the new identifier of the input control in the report.

Example

Moving an Input Control with Assigned Report Elements

```
PUT /documents/6671/reports/2/inputcontrols?fromId=D.IF0
```

Request body:

```
<inputcontrol>
  <assignedReportElements>
    <assignedReportElement refId="10"/>
    <assignedReportElement refId="20"/>
    <assignedReportElement refId="19"/>
  </assignedReportElements>
</inputcontrol>
```

Response:

```
<success>
  <message>The resource of type "Input Control" with identifier "D.IF0" has been successfully moved.</message>
  <id>R2.IF1</id>
</success>
```

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer
Web Intelligence REST API Reference
Another call to retrieve the definition of the report input control shows the assigned report elements:

GET /documents/6671/reports/2/inputcontrols/R2.IF1

Response:

```
<inputcontrol>
  <id>R2.IF1</id>
  <name>Country</name>
  <comboBox allowAllValuesSelection="true" useCustom="false" operator="Equal"/>
  <assignedDataObject refId="DP0.DO39" kind="Expression"/>
    <assignedReportElements>
      <assignedReportElement refId="10" reference="R2.IF1.T0"/>
      <assignedReportElement refId="20" reference="R2.IF1.T1"/>
      <assignedReportElement refId="19" reference="R2.IF1.T2"/>
    </assignedReportElements>
</inputcontrol>
```

### 8.2.11 Getting the Structure of a Report

#### Note

Prefer using the report element APIs to get Web Intelligence report content. See Managing Report Elements [page 345].

#### Usage

Gets the structure of a report.

#### Request

`GET /documents/<documentID>/reports/<reportID>/specification`

#### Response

Response type: `text/xml`

Response body: the description of the report structure.

#### Example

`GET /documents/2334/reports/23/specification`
8.2.12 Updating the Structure of a Report

Note
Prefer using the report element APIs to get Web Intelligence report content. See Managing Report Elements [page 345].

Usage

Updates the report structure of the specified report.

Request

PUT /documents/<documentID>/reports/<reportID>/specification

Request type: text/xml
Request body: the report structure

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

To Add a Cell to a Report Structure

PUT /documents/2334/reports/23/specification

Request body:

```
<REPORT rId="12" name="My First Cell">
    <PAGE_BODY>
        <CELL x="1000" y="1500">
            <CONTENT>="My First CELL"</CONTENT>
        </CELL>
    </PAGE_BODY>
</REPORT>
```

Response:

```
<success>
  <message>The resource of type "Report" with identifier "23" has been successfully updated.</message>
  <id>23</id>
</success>
```

Example

To Add a Vertical Table to a Report Structure

Request body:

```
<REPORT name="Vertical Table">
    <PAGE_HEADER/>
    <PAGE_BODY>
        <VTABLE name="My Vertical Table" x="3037" y="4455">
            <ROWGROUP type="HEADER">
                <TR>
                    <TDCELL>
                        <CONTENT>="NameOf([Country])"</CONTENT>
                    </TDCELL>
                    <TDCELL>
                        <CONTENT>="NameOf([Revenue])"</CONTENT>
                    </TDCELL>
                </TR>
            </ROWGROUP>
            <ROWGROUP type="BODY">
                <TR>
                    <TDCELL>
                        <CONTENT>=[Country]</CONTENT>
                    </TDCELL>
                    <TDCELL>
                        <CONTENT>=[Revenue]</CONTENT>
                    </TDCELL>
                </TR>
            </ROWGROUP>
        </VTABLE>
    </PAGE_BODY>
</REPORT>
```
Example

To Add a Vertical Table with Break

Request body:

```xml
<REPORT name="Vertical Table And Break">
  <PAGE_HEADER/>
  <PAGE_BODY>
    <VTABLE name="Table And Break " x="3037" y="4455">
      <AXIS>
        <EXPRS>
          <AXIS_EXPR>=[Country]</AXIS_EXPR>
          <AXIS_EXPR>=[Resort]</AXIS_EXPR>
        </EXPRS>
        <BREAK bId="1" expr="=[Country]" addSort="yes" onePage="yes" newPage="yes" duplicate="center"/>
      </AXIS>
      <ROWGROUP breakId="1" type="HEADER">
        <TR>
          <TDCELL>
            <CONTENT>=NameOf([Country])</CONTENT>
          </TDCELL>
          <TDCELL>
            <CONTENT>=NameOf([Resort])</CONTENT>
          </TDCELL>
          <TDCELL>
            <CONTENT>=NameOf([Revenue])</CONTENT>
          </TDCELL>
        </TR>
      </ROWGROUP>
      <ROWGROUP type="BODY">
        <TR>
          <TDCELL>
            <STYLE>
              <TEXTVALIGN value="center"/>
            </STYLE>
            <CONTENT>=[Country]</CONTENT>
          </TDCELL>
          <TDCELL>
            <CONTENT>=[Resort]</CONTENT>
          </TDCELL>
          <TDCELL>
            <CONTENT>=[Revenue]</CONTENT>
          </TDCELL>
        </TR>
        <TR>
          <TDCELL>
            <CONTENT>=[Country]</CONTENT>
          </TDCELL>
          <TDCELL>
            <CONTENT>=[Resort]</CONTENT>
          </TDCELL>
          <TDCELL>
            <CONTENT>=[Revenue]</CONTENT>
          </TDCELL>
        </TR>
        <TR>
          <TDCELL>
            <CONTENT>=[Country]</CONTENT>
          </TDCELL>
          <TDCELL>
            <CONTENT>=[Resort]</CONTENT>
          </TDCELL>
          <TDCELL>
            <CONTENT>=[Revenue]</CONTENT>
          </TDCELL>
        </TR>
      </ROWGROUP>
  </PAGE_BODY>
</REPORT>
```
Example

To Add a Vertical Table with one Filter on the Report and one Filter on the Table

Request body:

```xml
<REPORT name="Filter">  
  <DATA>
    <DATA_FILTER>
      <WHERE>
        <FILTER key="[Country]">
          <CONDITION operatorCondition="InList">
            <MEMBER>France</MEMBER>
          </CONDITION>
        </FILTER>
      </WHERE>
    </DATA_FILTER>
  </DATA>
  <PAGE_BODY>
    <VTABLE bId="17" name="Block 1" x="3749" y="3443">
      <AXIS>
        <EXPR>
          <AXIS_EXPR>=[Country]</AXIS_EXPR>
        </EXPR>
        <EXPR>
          <AXIS_EXPR>=[Resort]</AXIS_EXPR>
        </EXPR>
      </AXIS>
      <DATA>
        <DATA_FILTER>
          <WHERE>
            <FILTER key="[Resort]">
              <CONDITION operatorCondition="InList">
                <MEMBER>French Riviera</MEMBER>
              </CONDITION>
            </FILTER>
          </WHERE>
        </DATA_FILTER>
      </DATA>
      <ROWGROUP type="HEADER">
        <TR height="567">
          <TDCELL>  
            <CONTENT>=NameOf([Country])</CONTENT>
          </TDCELL>
          <TDCELL>  
            <CONTENT>=NameOf([Resort])</CONTENT>
          </TDCELL>
          <TDCELL>  
            <CONTENT>=NameOf([Revenue])</CONTENT>
          </TDCELL>
        </TR>
      </ROWGROUP>
      <ROWGROUP type="BODY">
        <TR height="567">
          <TDCELL>  
          </TDCELL>
          <TDCELL>  
          </TDCELL>
          <TDCELL>  
          </TDCELL>
        </TR>
      </ROWGROUP>
    </VTABLE>
  </PAGE_BODY>
</REPORT>
```
### Example

**To Add a Section with a Table and a Cell**

Request body:

```xml
<REPORT name="Section & Filter">
  <DATA/>
  <PAGE_BODY>
    <ROW height="6255"/>
    <SECTION>
      <DATA>
        <DATA_FILTER>
          <WHERE>
            <FILTER key="[Country]">
              <CONDITION operatorCondition="InList">
                <MEMBER>France</MEMBER>
              </CONDITION>
            </FILTER>
          </WHERE>
        </DATA_FILTER>
      </DATA>
      <AXIS>
        <SORTS>
          <SORT sign="desc" expr="=[Country]"/>
        </SORTS>
      </AXIS>
      <SBODY bottomPadding="3638" bookmark="yes">
        <CELL class="ia-section-cell" x="600" y="500">
          <CONTENT>=[Country]</CONTENT>
        </CELL>
        <VTABLE name="Block 1" x="3749" y="500">
          <DATA/>
          <ROWGROUP type="HEADER">
            <TR>
              <TDCELL>
                <CONTENT>=NameOf([Resort])</CONTENT>
              </TDCELL>
            </TR>
            <TR>
              <TDCELL>
                <CONTENT>=NameOf([Revenue])</CONTENT>
              </TDCELL>
            </TR>
          </ROWGROUP>
          <ROWGROUP type="BODY">
            <TR>
              <TDCELL>
                <CONTENT>=[Resort]</CONTENT>
              </TDCELL>
              <TDCELL>
                <CONTENT>=[Revenue]</CONTENT>
              </TDCELL>
            </TR>
          </ROWGROUP>
        </VTABLE>
      </SBODY>
    </SECTION>
  </PAGE_BODY>
</REPORT>
```

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
Example

To Add Cells with hideAlways Attribute

Request body:

```xml
<REPORT name="Cell with hide always and Cell with hide conditional">
  <DATA/>
  <PAGE_BODY>
    <ROW height="2992" bottomPadding="0">
      <CELL x="7416" y="432" hideAlways="yes" width="15048" height="1547">
        <CONTENT>ALWAYS HIDE</CONTENT>
      </CELL>
      <CELL x="7416" y="2000" hideAlways="no" hideExpr="=Count([Country])>3"  width="15048" height="1547">
        <CONTENT>=Count([Country])>3</CONTENT>
      </CELL>
    </ROW>
  </PAGE_BODY>
</REPORT>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]

8.2.13 Managing Data Filters in Reports

Below are the tasks you can perform on the data filters of a report.

- Creating a Data Filter [page 326]
- Getting the Details of a Data Filter [page 327]
- Updating a Data Filter [page 328]
- Deleting a Data Filter [page 330]
8.2.13.1 Creating a Data Filter

Usage

Creates a data filter for a specific report, referenced by its ID.

Request

POST /documents/<documentID>/reports/<reportID>/datafilter

Request body: the description of the data filter. The key attribute should be one of the formulaLanguageId elements of the report.

The operator is one of the available report filters:

- Equal
- NotEqual
- Greater
- GreaterOrEqual
- Less
- LessOrEqual
- Between
- NotBetween
- InList
- NotInList
- IsNull
- IsNotNull
- IsAny
- Like
- NotLike
- Both
- Except

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/127/reports/1/datafilter
Request body:

```xml
<datafilter>
  <and>
    <condition key="[Lines]" operator="InList">
      <value>City Trousers</value>
      <value>Shirt Waist</value>
      <value>Jackets</value>
    </condition>
    <condition key="[State]" operator="NotEqual">
      <value>Colorado</value>
    </condition>
  </and>
  <or>
    <condition key="[Sales revenue]" operator="Greater">
      <value>5000</value>
    </condition>
    <condition key="[Sales revenue]" operator="Less">
      <value>10</value>
    </condition>
  </or>
</datafilter>
```

Response:

```xml
<success>
  <message>The resource of type "Report" with identifier "1" has been successfully updated.</message>
  <id>1</id>
</success>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]

8.2.13.2 Getting the Details of a Data Filter

Usage

Gets the description of the data filter of a specific report, referenced by its ID.

Request

```plaintext
GET /documents/<documentID>/reports/<reportID>/datafilter
```
Response

Response type: application/xml or application/json

Response body: the description of the data filter. The key attribute should be one of the formulaLanguageId elements of the report.

Example

GET /documents/4567/reports/1/datafilter

Response:

```xml
<datafilter>
  <and>
    <condition key="[Lines]" operator="InList">
      <value>City Trousers</value>
      <value>Shirt Waist</value>
      <value>Jackets</value>
    </condition>
    <condition key="[State]" operator="NotEqual">
      <value>Colorado</value>
    </condition>
  </and>
  <or>
    <condition key="[Sales revenue]" operator="Greater">
      <value>5000</value>
    </condition>
    <condition key="[Sales revenue]" operator="Less">
      <value>10</value>
    </condition>
  </or>
</datafilter>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]

8.2.13.3 Updating a Data Filter

Usage

Updates the data filter applied to a given report specified in the URL parameter.
Request

PUT /documents/<documentID>/reports/<reportID>/datafilter

Request body: the description of the data filter to update. The operator is one of the available report filter conditions:

- Equal
- NotEqual
- Greater
- GreaterOrEqual
- Less
- LessOrEqual
- Between
- NotBetween
- InList
- NotInList
- IsNull
- IsNotNull
- IsAny
- Like
- NotLike
- Both
- Except

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/223/reports/1/datafilter

Request body:

```
<datafilter>
  <and>
    <condition key="[Lines]" operator="InList">
      <value>City Trousers</value>
      <value>Shirt Waist</value>
      <value>Jackets</value>
    </condition>
    <condition key="[State]" operator="NotEqual">
      <value>Colorado</value>
    </condition>
  </and>
  <or>
    <condition key="[Sales revenue]" operator="Greater">
      <value>5000</value>
    </condition>
  </or>
</datafilter>
```
Response:

```xml
<success>
    <message>The resource of type "Report" with identifier "1" has been successfully updated.</message>
    <id>1</id>
</success>
```

### Related Information

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]

### 8.2.13.4 Deleting a Data Filter

#### Usage

Deletes a data filter from a specific report, referenced by its ID.

#### Request

```
DELETE /documents/<documentID>/reports/<reportID>/datafilter
```

#### Response

**Response type:** application/xml or application/json

The response is a message stating the success or failure of the request.

**Example**

```
DELETE /documents/9512/reports/67/datafilter
```
8.3  Drilling on Report Data

You use the drill function to analyze the results displayed in reports. Drilling on reports lets you look deeper into your data to discover the details behind a summary result displayed in tables, charts, or sections.

Below are the main drill tasks you can perform on reports.

- Getting the Drill Mode [page 332]
- Updating the Drill Mode [page 332]
- Enabling the Query Drill [page 333]
- Disabling the Query Drill [page 334]
- Getting Information on the Drill Hierarchies [page 335]
- Getting the Free Drill Elements [page 337]
- Creating a Drill Filter [page 338]
- Getting the Drill Filters of a Report [page 339]
- Getting the Details of a Drill Filter [page 340]
- Updating a Drill Filter [page 341]
- Removing a Drill Filter [page 342]
- Performing a Drill [page 343]
- Making a Snapshot of a Report in Drill Mode [page 344]
8.3.1 Getting the Drill Mode

Usage

Obtains the current drill information.
Returns an error if the drill is not activated on the report.

Request

GET /documents/<documentID>/reports/<reportID>/driller

Response

Response type: application/xml or application/json
Response body: the drill settings, where <output> specifies the drill mode (true|false).

Example

GET /documents/4567/reports/1/driller
Response:

<driller>
  <output>true</output>
</driller>

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]

8.3.2 Updating the Drill Mode

Usage

Updates the drill mode of a report.
Request

PUT /documents/<documentID>/reports/<reportID>/driller

Request body:

```
<driller>
  <output/>
```

<output> specifies the drill mode (true|false).

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

```
Example

PUT /documents/3422/reports/2/driller

Request body:

```
<driller>
  <output>false</output>
</driller>
```

Response:

```
<success>
  <message>The resource of type "Driller" has been successfully updated."/message>
  <id>1</id>
</success>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]

8.3.3 Enabling the Query Drill

Usage

Activates the drill on a report.
Request

POST /documents/<documentID>/reports/<reportID>/driller

Request body (optional):

```xml
<driller>
  <output>
  </output>
</driller>
```

<output> specifies the drill mode \((true|false)\). If not provided, the drill is activated with the default settings \((true)\).

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

```xml
<success>
  <message>The resource of type "Driller" has been successfully created.</message>
  <id>2</id>
</success>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]

8.3.4 Disabling the Query Drill

Usage

Disables the drill on the specified report.
Request

DELETE /documents/<documentID>/reports/<reportID>/driller

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /8022/reports/1/driller

Response:

  <success>
    <message>The resource of type "Driller" has been successfully removed.</message>
    <id>2</id>
  </success>

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]

8.3.5 Getting Information on the Drill Hierarchies

Usage

Obtains the drill hierarchies.

It returns an error if the drill is not activated on this report.

Request

GET /documents/<documentID>/reports/<reportID>/driller/hierarchies
Response

Response type: application/xml or application/json

Response body: the details of the drill hierarchies.

Example

GET /documents/4567/reports/1/driller/hierarchies

Response:

```xml
<hierarchies>
  <hierarchy>
    <id>DP0.DH1</id>
    <name>Resort Hierarchy</name>
    <dataProviderId>DP0</dataProviderId>
    <elements>
      <element>
        <id>DP0.DO39</id>
        <name>Country</name>
        <description></description>
        <filterValue></filterValue>
        <qualification>dimension</qualification>
        <inQuery>false</inQuery>
        <ambiguous>false</ambiguous>
        <inScope>true</inScope>
      </element>
      <element>
        <id>DP0.DO2</id>
        <name>Resort</name>
        <description></description>
        <filterValue></filterValue>
        <qualification>dimension</qualification>
        <inQuery>false</inQuery>
        <ambiguous>false</ambiguous>
        <inScope>true</inScope>
      </element>
      <element>
        <id>DP0.DO4</id>
        <name>Service Line</name>
        <description></description>
        <filterValue></filterValue>
        <qualification>dimension</qualification>
        <inQuery>false</inQuery>
        <ambiguous>false</ambiguous>
        <inScope>true</inScope>
      </element>
      <element>
        <id>DP0.DO5</id>
        <name>Service</name>
        <description></description>
        <filterValue></filterValue>
        <qualification>dimension</qualification>
        <inQuery>false</inQuery>
        <ambiguous>false</ambiguous>
        <inScope>true</inScope>
      </element>
    </elements>
  </hierarchy>
</hierarchies>
```
8.3.6 Getting the Free Drill Elements

Usage

Obtains the free drill elements.
Returns an error if the drill is not activated on the report.

Request

GET /documents/<documentID>/reports/<reportID>/driller/drillelements

Response

Response type: application/xml or application/json

Response body: the details of the available drill elements of the report:

- qualification
- <id>
- <name>

Example

GET /documents/4567/reports/1/driller/drillelements

Response:

<drillelements>
  <drillelement qualification="Dimension" inQuery="false" ambiguous="false"
inScope="true">
    <id>DP0.DO39</id>
    <name>Country</name>
  </drillelement>
  <drillelement qualification="Dimension" inQuery="false" ambiguous="false"
inScope="true">
    <id>DP0.DO2</id>
    <name>Resort</name>
  </drillelement>
  <drillelement qualification="Dimension" inQuery="false" ambiguous="false"
inScope="true">
    <id>DP0.DO4</id>
    <name></name>
  </drillelement>
</drillelements>
8.3.7 Creating a Drill Filter

Usage

Creates a drill filter for a report.

Returns an error if the drill is not activated on the report.

Request

POST /documents/<documentID>/reports/<reportID>/driller/filters

Request body: the identifier and value of the report drill filter.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/127/reports/1/driller/filters

Request body:

```xml
<filter>
  <id>DP1.DO22</id>
  <value>France</value>
</filter>
```
Response:

```xml
<success>
  <message>The resource of type "DrillFilter" with identifier "DP1.DO22" has been successfully created.</message>
  <id>DP1.DO22</id>
</success>
```

Related Information

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]

8.3.8 Getting the Drill Filters of a Report

Usage

Get the list of drill filters for a report.

Returns an error if the drill is not activated on the report.

Request

GET /documents/<documentID>/reports/<reportID>/driller/filters

Response

Response type: application/xml or application/json

Response body: the description of the available filters for the report.

Example

```xml
GET /documents/4567/reports/1/driller/filters
```

Response:

```xml
<filters>
  <filter qualification="Dimension" inQuery="false" ambiguous="false" inScope="true">
    <id>DP0.DO39</id>
  </filter>
</filters>
```
8.3.9 Getting the Details of a Drill Filter

Usage

Gets the details of a drill filter for a report.

Returns an error if the drill is not activated on the report.

Request

GET /documents/<documentID>/reports/<reportID>/driller/filters/<filterID>

Response

Response type: application/xml or application/json

Response body: the description of the drill filter.

Example

GET /documents/13343/reports/1/driller/filters/DP0.DO13

Response:

<filter qualification="Dimension" inQuery="false" ambiguous="false" inScope="true">
  <id>DP0.DO13</id>
  <name>City</name>
  <lov hierarchical="false" partial="false" refreshable="false">
    <values>
      <value>Albertville</value>
      <value>Augsburg</value>
      <value>Belfast</value>
      ...  
      <value>Washington D.C.</value>
    </values>
  </lov>
</filter>
8.3.10 Updating a Drill Filter

Usage

Updates the definition of a drill filter for a report.
Returns an error if the drill is not activated on the report.

Request

PUT /documents/<documentID>/reports/<reportID>/driller/filters/<filterID>

Request body: the drill filter to update.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/223/reports/2/driller/filters/D02.011

Request body:

<filter>
  <value>Germany</value>
</filter>
Response:

```xml
<success>
  <message>The resource of type "DrillFilter" with identifier "DP2.O11" has been successfully updated.</message>
  <id>DP2.O11</id>
</success>
```

**Related Information**

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]
- Getting the Drill Filters of a Report [page 339]

### 8.3.11 Removing a Drill Filter

**Usage**

Removes a drill filter for a report.

Returns an error if the drill is not activated on the report.

**Request**

DELETE /documents/<documentID>/reports/<reportID>/driller/filters/<filterID>

**Response**

Response type: `application/xml` or `application/json`

The response is a message stating the success or failure of the request.

**Example**

DELETE /documents/9512/reports/67/driller/filters/DP5.O12

Response:

```xml
<success>
  <message>The resource of type "DrillFilter" with identifier "DP5.O12" has been successfully deleted.</message>
  <id>DP5.O12</id>
</success>
```
Related Information

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]
- Getting the Drill Filters of a Report [page 339]

8.3.12 Performing a Drill

Usage

Performs a drill on a report.

This throws an error if the resource is not created or if the parameters are not valid.

Request

POST /documents/<documentID>/reports/<reportID>/driller/instructions

Request body: the drill instructions. The drill instructions are made of the following:

- The instruction type (either "Down", "Up", or "By")
- The identifier of the report element on which you perform the drill
- The drill elements on which you perform the drill

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/9512/reports/67/driller/instructions

Request body:

```xml
<instruction type="Down">
   <elementId>11</elementId>
   <from>
      <drillElement>
```
Response:

```xml
<success>
  <message>The resource of type "DrillerInstruction" has been successfully created.</message>
  <id>1</id>
</success>
```

Related Information

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]

8.3.13 Making a Snapshot of a Report in Drill Mode

Usage

Makes a snapshot of a report in drill mode.
This throws an error if the resource is not created or if the parameters are not valid.

Request

POST /documents/<documentID>/reports/<reportID>/driller/snapshot

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.
### Example

**POST /documents/7738/reports/12/driller/snapshot**

**Response:**

```xml
<success>
  <message>Resource of type "Report" with identifier "12" has been successfully created.</message>
  <id>12</id>
</success>
```

### Related Information

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]

### 8.4 Managing Report Elements

Below are the main tasks you can perform on the elements of a report.

- Creating a Report Element [page 346]
- Getting the List of Report Elements [page 349]
- Getting the Details of a Report Element [page 352]
- Getting the Background of a Report Element [page 360]
- Getting the Datapaths of a Report Element [page 361]
- Getting the Dataset of a Report Element [page 364]
- Updating a Report Element [page 370]
- Inserting a Row or Column into a Table [page 375]
- Merging or Splitting Cells of a Table [page 376]
- Exporting a Report Element [page 377]
- Deleting a Report Element [page 381]
- Working with Axes [page 383]
- Working with Rankings [page 390]
- Working with Custom Properties [page 394]
- Managing Data Filters in Report Elements [page 398]
8.4.1 Creating a Report Element

Usage

Creates one of the following report elements:

- Cell (only free cells, not cells in a table)
- Section
- Table (VTable, HTable, XTable, and Form)
- Visualization (charts)

Remember

You cannot create PageZone elements and table elements of type Cell.

Request

POST /documents/<documentID>/reports/<reportID>/elements?unit=<unit>

Where:

- <unit> is an optional parameter of type string that defines the unit of measurement used for all dimensional values such as size, padding, and position. Values are metric (default), inch and centimeter.

Request type: application/xml or application/json

Request body: the valid definition of an element. See the Chart Response Body Schema [page 59] in the case of a chart.

Some settings are common to all types of report elements, while some are only specific. These specific settings are located into content tags. The element is added and an ID attributed to the element.

Remember

- You can create a report element with its expressions or create an empty report element, and add expressions in a second call using PUT /documents/<documentID>/reports/<reportID>/elements/<elementID>/axes/<axisID>/expressions.
- You can change the type of report element after you create it by calling PUT /documents/<documentID>/reports/<reportID>/elements/<elementID>.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.
**Example**

**Cell**

POST /documents/127/reports/1/elements?unit=inch

In the request body, you define the cell dimensions, padding and contents.

```xml
<element type="Cell">
  <parentId>2</parentId>
  <size minimalWidth="2.23" minimalHeight="3.56" autofitWidth="false" autofitHeight="true"/>  
  <padding left="0.069" right="0.069" top="0.069" bottom="0.069"/>
  <content>
    <expression>
      <formula type="Text" dataType="Numeric">=Sum([Sales revenue])</formula>
    </expression>
  </content>
</element>
```

**Example**

**Section**

POST /documents/127/reports/1/elements

In the request body, you define how duplicate row aggregation is managed and a formula.

```xml
<element type="Section">
  <parentId>2</parentId>
  <size minimalHeight="1000"/>
  <padding bottom="1000"/>
  <position repeatOnEveryVerticalPage="true"/>
  <style>
    <background>
      <color rgb="#0000ff"/>
    </background>
  </style>
  <content>
    <axes duplicateRowAggregation="true">
      <axis role="Row">
        <expressions>
          <formula dataType="String">=[Year]</formula>
        </expressions>
      </axis>
    </axes>
  </content>
</element>
```

**Example**

**Table**

POST /documents/14695/reports/1/elements

Request body:

```xml
<element type="XTable">
  <parentId>2</parentId>
  <content>
    <axes>
      
    </axes>
  </content>
</element>
```
Example

Chart

POST /documents/13069/reports/6/elements?unit=centimeter

Request body:

```xml
<element type="Visualization">
  <parentId>2</parentId>
  <content>
    <chart type="HorizontalBar">
      <axes>
        <axis role="Color">
          <expressions>
            <formula dataType="String">=[Resort]</formula>
            <formula dataType="String">=[Country]</formula>
          </expressions>
        </axis>
        <axis role="Category">
          <expressions>
            <formula dataType="String">=[Year]</formula>
          </expressions>
        </axis>
        <axis role="Value">
          <expressions>
            <formula dataType="Numeric">=[Revenue]</formula>
          </expressions>
        </axis>
      </axes>
    </chart>
  </content>
</element>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
8.4.2 Getting the List of Report Elements

Usage

Gets the elements of a specific report.

The following items identify a report element:

- Id
- Reference

Request

GET /documents/<documentID>/reports/<reportID>/elements?unit=<unit>

Where:

- <unit> is an optional parameter of type string that defines the unit of measurement used for all dimensional values such as size, padding, and position. Values are metric (default), inch and centimeter.

Response

Response type: application/xml or application/json

Response body: the list of elements of the report, with the following information:

- <id>
- <name>
- <reference>
- <size>
- <position>
- <padding>
- type attribute, with the possible values PageZone, Cell, VTable, HTable, XTable, Form, and Visualization.

Example

XML

GET /documents/6528/reports/1/elements?unit=centimeter

Response:

<elements>
Example

JSON

GET /documents/5157/reports/1/elements

Response:

```json
{"elements":{
  "element":[
    {
      "@type":"PageZone",
      "id":1,
      "name":"Header",
      "size":{"@minimalHeight":"1984.0"}
    },
    {
      "@type":"Cell",
      "id":19,
      "reference":"1.J",
      "parentId":29,
      "size":{
        "@autofitHeight":"true",
        "@autofitWidth":"false",
        "@minimalHeight":"567.0",
        "@minimalWidth":"3005.0"},
      "padding":{
        "@bottom":"230.0",
        "@top":"220.0",
        "@right":"250.0",
        "@left":"250.0"}
    },
    ...
    {
      "@type":"HTable",
      "id":29,
      "reference":"1.T",
      "name":"Block 1 (1)",
      "parentId":2,
      "position":{
        "@newVerticalPage":"false",
        "@oneVerticalPage":"false",
        "@repeatOnEveryVerticalPage":"false",
        "@newHorizontalPage":"false",
        "@oneHorizontalPage":"false",
        "@verticalAnchorType":"None",
        "@horizontalAnchorType":"None",
        "@y":"1950.0",
        "@x":"2662.0"}
    },
    ...
    {
      "@type":"VTable",
      "id":16,
      "reference":"1.G",
      "name":"Block 1",
      "parentId":2,
      "position":{
        "@newVerticalPage":"false",
        "@oneVerticalPage":"false",
        "@repeatOnEveryVerticalPage":"false",
        "@newHorizontalPage":"false",
        "@oneHorizontalPage":"false",
        "@verticalAnchorType":"None",
        "@horizontalAnchorType":"None",
        "@y":"18450.0",
        "@x":"2700.0"}
    },
    ...
    {
      "@type":"XTable",
      "id":42,
      ...}
  ]
}}
```
Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]

8.4.3 Getting the Details of a Report Element

Usage

Gets the details of a specific report element, referenced by its ID.

Request

```
GET /documents/<documentID>/reports/<reportID>/elements/<elementID>?
<optional_parameters>
```

Table 176: Optional Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| reference | Specifies an element reference, for example: 
{elementId}?reference=1.D.7. This results in an additional piece of information called datapath. |
| unit      | Specifies the unit of measurement used for all dimensional values such as size, padding, and position. Values are metric (default), inch and centimeter. |
Response:

Response type: application/xml or application/json

Response body: the definition of a report element, with the following information:

- <id>
- <reference>
- <parentId>
- <size>
- <position>
- <hide>
- <padding>
- <style>
- <content>

- type attribute, with the possible values PageZone, Cell, VTable, HTable, XTable, Form, and Visualization.

Example

Cell

GET /documents/6580/reports/1/elements/4?unit=inch

Response:

```
<element type="Cell">
  <id>4</id>
  <reference>1.4</reference>
  <parentId>2</parentId>
  <size minimalWidth="4.18" minimalHeight="0.43" autofitWidth="false" autofitHeight="true" />
  <position x="2.06" y="0.12" horizontalAnchorType="None" verticalAnchorType="None" oneHorizontalPage="false" newHorizontalPage="false" repeatOnEveryVerticalPage="false" oneVerticalPage="false" newVerticalPage="false" />
  <hide always="false" />
  <padding left="0.059" right="0.059" top="0.043" bottom="0.043" />
  <style>
    <border>
      <top thickness="None" rgb="#000000" style="None" />
      <bottom thickness="None" rgb="#000000" style="None" />
      <left thickness="None" rgb="#000000" style="None" />
      <right thickness="None" rgb="#000000" style="None" />
    </border>
    <background>
      <color rgb="#000000" />
    </background>
    <font size="16" face="Arial" italic="false" bold="true" strikethrough="false" underline="false" rgb="#333333" />
    <alignment horizontal="Center" vertical="Center" wrapText="false" />
  </style>
  <content>
    <expression>
      <formula type="Text" dataType="Numeric">=[帳票_累計(内)_発生]+[帳票_累計(外)_発生]</formula>
    </expression>
    <format type="Custom">
      <template positive="#,##0" negative="-#,##0" zero="0" undefined="0" />
    </format>
  </content>
</element>
```
Example

PageZone

GET /documents/6580/reports/1/elements/1?unit=centimeter

Response:

```xml
<element type="PageZone">
  <id>1</id>
  <size minimalHeight="1.4"/>
  <hide always="false"/>
  <style>
    <border>
      <top thickness="None" rgb="#000000" style="None"/>
      <bottom thickness="None" rgb="#000000" style="None"/>
      <left thickness="None" rgb="#000000" style="None"/>
      <right thickness="None" rgb="#000000" style="None"/>
    </border>
    <background>
      <color rgb="#ffffff"/>
    </background>
  </style>
</element>
```

---

Example

Section

GET /documents/6580/reports/1/elements/13?unit=inch

Response:

```xml
<element type="Section">
  <id>13</id>
  <reference>1.D</reference>
  <parentId>2</parentId>
  <size minimalHeight="0.2"/>
  <position repeatOnEveryVerticalPage="true" oneVerticalPage="true" newVerticalPage="true"/>
  <hide always="false">
    <expression dataType="Boolean">Min([Number of guests])=0</expression>
  </hide>
  <padding bottom="0.1"/>
  <style>
    <background>
      <color rgb="#ffffce"/>
    </background>
  </style>
  <content>
    <axes duplicateRowAggregation="true">
      <axis role="Row">
        <id>0</id>
        <expressions>
          <formula dataType="String" dataObjectId="DP0.DO39">=[Country]</formula>
        </expressions>
      </axis>
    </axes>
  </content>
</element>
```
Example

**XTable**

GET /documents/19035/reports/2/elements/12?unit=centimeter

Response:

```xml
<element type="XTable">
  <id>12</id>
  <reference>2.C</reference>
  <name>Block 1</name>
  <parentId>1</parentId>
  <position x="4.894" y="1.376" horizontalAnchorType="None" verticalAnchorType="None" oneHorizontalPage="false" newHorizontalPage="false" repeatOnEveryVerticalPage="false" oneVerticalPage="false" newVerticalPage="false"/>
  <hide always="false"/>
  <style>
    <border>
      <top thickness="None" rgb="#000000" style="None"/>
      <bottom thickness="None" rgb="#000000" style="None"/>
      <left thickness="None" rgb="#000000" style="None"/>
      <right thickness="None" rgb="#000000" style="None"/>
    </border>
    <background width="0.0" height="0.0"/>
    <alternateColor frequency="2" rgb="#fcfdfd"/>
  </style>
  <content>
    <axes duplicateRowAggregation="true">
      <axis role="Row">
        <id>0</id>
        <breaks>
          <break onePage="false" newPage="false" sort="true" duplicate="Remove" repeatHeader="false" showHeader="true" showFooter="true"/>
        </breaks>
        <formula dataType="String" dataObjectId="DP0.DO31">=[Year]</formula>
      </axis>
      <axis role="Column">
        <formula dataType="String" dataObjectId="DP0.DO31">=[Country]</formula>
      </axis>
    </axes>
    <expressions>
      <formula dataType="String" dataObjectId="DP0.DO31">=[Year]</formula>
    </expressions>
  </content>
</element>
```
Example

Cell with Reference

GET /documents/8362/reports/1/elements/6?reference=1.6.Ga

[element type="Cell">
  <id>6</id>
  <reference>1.6</reference>
  <parentId>11</parentId>
  <datapath>DP0.DO1:"Robert King"|"0000000007",DP0.DO10:" 6561 - Centre Sport"|"0000006561"</datapath>
  ...
</element>
Example

Chart

GET /documents/6503/reports/1/elements/39?unit=inch

```xml
<element type="Visualization">
  <id>39</id>
  <reference>1.d</reference>
  <name>Block 2 (2)</name>
  <parentId>2</parentId>
  <size minimalWidth="4.165" minimalHeight="3.125"/>
  <position x="0.334" y="0.018" oneHorizontalPage="false"
    newHorizontalPage="false"
    repeatOnEveryVerticalPage="false" oneVerticalPage="false"
  />
  <hide always="false"/>
  <style>
    <border>
      <top thickness="None" rgb="#000000" style="None"/>
      <bottom thickness="None" rgb="#000000" style="None"/>
      <left thickness="None" rgb="#000000" style="None"/>
      <right thickness="Thin" rgb="#ff0000" style="Dashed"/>
    </border>
    <background>
      <gradient orientation="Horizontal">
        <start alpha="255" rgb="#ffff00"/>
        <end alpha="255" rgb="#ff00ff"/>
      </gradient>
    </background>
  </style>
  <content>
    <chart type="HorizontalBar">
      <layout showDimensionsWithEmptyMeasureValues="true"
        showDimensionsWithMeasuresEqualToZero="false"
        showDimensionsWithSumOfMeasuresEqualToZero="true"
        showMeasuresWithEmptyDimensionValues="true"
        showParentNodes="true" duplicateRowAggregation="true"
        horizontal="true"/>
      <title visible="true">
        <style>
          <border thickness="None">
        </border>
        <background>
          <color alpha="255" rgb="#000000"/>
        </background>
        <font size="9" face="Arial" italic="false" bold="true"
          strikethrough="false" underline="false" rgb="#000000"/>
        <alignment horizontal="Left" vertical="Center" textPolicy="Truncate"/>
      </style>
      </title>
      <legend visible="true">
        <style>
          <border thickness="None">
        </border>
        <background>
          <color alpha="0" rgb="#000000"/>
        </background>
        <font size="8" face="Arial" italic="false" bold="true"
          strikethrough="false" underline="false" rgb="#707070"/>
      </style>
      </legend>
    </chart>
  </content>
</element>
```
8.4.4 Getting the Background of a Report Element

Usage

Returns the background image or skin of a report element as a binary stream.

Request

Response

The response type depends on the background image or skin format, for example:

- `image/bmp`
- `image/gif`
- `image/png`
- `image/jpg`
- `image/jpeg`

Use `image/*` if you do not know the background format.

An error is thrown when:

- There is no background.
- Response type and background format are not compatible.

Example

GET /documents/9199/reports/154769/elements/289477/background

Response:

```
Content-Type: image/png
HTTP Response Code: 200
```

Related Information

- [Getting the List of Documents](page 195)
- [Getting the List of Reports](page 290)
- [Getting the List of Report Elements](page 349)

8.4.5 Getting the Datapaths of a Report Element

Usage

Gets the datapaths available for a report element.

A datapath allows you to find a specific instance of a report element that can be used for example in several sections, or to find a specific report element of type cell in a table. The datapath may also be used to export a report element or to get its associated data.
Request


Table 177: Optional Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datapath</td>
<td>A string that specifies an initial datapath needed to get to the next level, for example datapaths?datapath=DP0.DO18:&quot;FY93&quot;.</td>
</tr>
</tbody>
</table>

⚠️ Caution

You cannot use datapath and reference parameters in the same URL. This generates a 400 HTTP error.

Response

Response type: application/xml or application/json

Example

No Datapath

GET /documents/8413/reports/1/elements/20/datapaths

Response:

```xml
<datapaths elementId="17">
  <datapath>DP0.DObc:"2004"</datapath>
  <datapath>DP0.DObc:"2005"</datapath>
  <datapath>DP0.DObc:"2006"</datapath>
</datapaths>
```

ℹ️ Note

The returned elementId attribute specifies the report element identifier on which to go on iterating.

Example

With a Datapath

GET /documents/8413/reports/1/elements/17/datapaths?datapath=DP0.DObc:"2006"

Response:

```xml
<datapaths elementId="10">
  <datapath>DP0.DObc:"2006",DP0.DOda:"California"</datapath>
  <datapath>DP0.DObc:"2006",DP0.DOda:"Colorado"</datapath>
  <datapath>DP0.DObc:"2006",DP0.DOda:"DC"</datapath>
  <datapath>DP0.DObc:"2006",DP0.DOda:"Florida"</datapath>
</datapaths>
```
<datapath>DP0.DOabc:"2006",DP0.DOda:"Illinois"</datapath>
<datapath>DP0.DOabc:"2006",DP0.DOda:"Massachusetts"</datapath>
<datapath>DP0.DOabc:"2006",DP0.DOda:"New York"</datapath>
<datapath>DP0.DOabc:"2006",DP0.DOda:"Texas"</datapath>
</datapaths>

**Example**

**With a Reference**

GET /documents/8413/reports/1/elements/10/datapaths?reference=1.5.2.7

Response:

```xml
<datapaths elementId="11">
  <datapath>DP0.DOa5:"Accessories"</datapath>
  <datapath>DP0.DOa5:"City Skirts"</datapath>
  <datapath>DP0.DOa5:"City Trousers"</datapath>
  <datapath>DP0.DOa5:"Dresses"</datapath>
  <datapath>DP0.DOa5:"Jackets"</datapath>
  <datapath>DP0.DOa5:"Leather"</datapath>
  <datapath>DP0.DOa5:"Outerwear"</datapath>
  <datapath>DP0.DOa5:"Overcoats"</datapath>
  <datapath>DP0.DOa5:"Shirt Waist"</datapath>
  <datapath>DP0.DOa5:"Sweaters"</datapath>
  <datapath>DP0.DOa5:"Sweat-T-Shirts"</datapath>
  <datapath>DP0.DOa5:"Trousers"</datapath>
</datapaths>
```

**Example**

**Workflow**

1. GET /documents/8275/reports/1/elements/16/dataset to get chart data (ID=16).
   This returns an error:
   ```xml
   <error>
     <error_code>101</error_code>
     <message>The resource of type "Report element" with identifier "16" contains datapaths: one must be specified.</message>
   </error>
   ```

2. GET /documents/8275/reports/1/elements/16/datapaths to get a datapath to access values.
   This returns:
   ```xml
   <datapaths elementId="5">
     <datapath>DP0.DO18:"FY92"</datapath>
     <datapath>DP0.DO18:"FY93"</datapath>
     <datapath>DP0.DO18:"FY94"</datapath>
     <datapath>DP0.DO18:"FY95"</datapath>
   </datapaths>
   ```

3. GET /documents/8275/reports/1/elements/5/datapaths?datapath=DP0.DO18:"FY92" to get the needed datapath, given the elementId and the chosen datapath:
   ```xml
   <datapaths elementId="8">
     <datapath>DP0.DO18:"FY92",DP0.DOla:"Q1"</datapath>
     <datapath>DP0.DO18:"FY92",DP0.DOla:"Q2"</datapath>
     <datapath>DP0.DO18:"FY92",DP0.DOla:"Q3"</datapath>
     <datapath>DP0.DO18:"FY92",DP0.DOla:"Q4"</datapath>
   </datapaths>
   ```
<dataset>
  <data id="Category" type="String">
    <label>Month</label>
    <value>Feb</value>
  </data>
  <data id="Value" type="Numeric">
    <label>Revenue</label>
    <value>291190</value>
  </data>
</dataset>

8.4.6 Getting the Dataset of a Report Element

Usage

Returns the data of a report element.

You may use a specific datapath or reference of a report element to retrieve its data.

Request

### Table 178: Optional Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datapath</td>
<td>A string that specifies a datapath, which allows you to find a specific instance of a report element, for example <code>dataset?datapath=DP1.DOabc:&quot;2004&quot;,DP1.DOa6:&quot;Austin&quot;</code>.</td>
</tr>
</tbody>
</table>

⚠️ **Caution**

You cannot use `datapath` and `reference` parameters in the same URL. This generates a 400 HTTP error.

### Response

**Response type:** `application/xml` or `application/json`

**Response body:** the details of the dataset, which are:

- The column metadata (name, type, and data object identifier)
- A list of rows containing the values

#### Example

**Cell**


Response:

```xml
<dataset>
  <metadata>
    <value dataObjectId="DP1.DO93" type="Numeric">Sales revenue</value>
  </metadata>
  <row>
    <value>910451.2</value>
  </row>
</dataset>
```

#### Example

**VTable**

The vertical table contains six rows which can display up to five data object values.

GET `/documents/6395/reports/1/elements/78/dataset`

Response:

```xml
<dataset>
  <metadata>
    <value dataObjectId="DP1.DO84" type="String">Country</value>
    <value dataObjectId="DP1.DO2" type="String">Resort</value>
    <value dataObjectId="DP1.DO5" type="String">Service</value>
    <value dataObjectId="DP1.DO14" type="Numeric">Number of guests</value>
    <value dataObjectId="DP1.DO7" type="Numeric">Revenue</value>
  </metadata>
</dataset>
```
Example

HTable

The horizontal table contains six columns that can display up to three data object values.

GET /documents/6361/reports/3/elements/101/dataset

Response (JSON):

{"dataset": "{"metadata": 
{"value": 
[["@dataObjectId":"DP0.DOa5", "@type":"String", "$":"Region"], 
["@dataObjectId":"DP0.DOa7", "@type":"String", "$":"City"], 
["@dataObjectId":"DP0.DO14", "@type":"Numeric", "$":"Number of guests"]}]}
Example

XTable

The cross table displays the values of six data objects.

GET /documents/6395/reports/3/elements/19/dataset

Response:

```xml
<dataset>
  <metadata>
    <value dataObjectId="DP1.DO84" type="String">Country</value>
    <value dataObjectId="DP1.DOa7" type="String">City</value>
    <value dataObjectId="DP1.DO2" type="String">Resort</value>
    <value dataObjectId="DP1.DO5" type="String">Service</value>
    <value dataObjectId="DP1.DO7" type="Numeric">Revenue</value>
    <value dataObjectId="DP1.DO14" type="Numeric">Number of guests</value>
  </metadata>
  <row>
    <value>Bahamas Islands</value>
    <value>Coral Bay</value>
    <value>Bahamas Beach</value>
    <value>Activities</value>
    <value>5600</value>
  </row>
  <row>
    <value>Bahamas Islands</value>
    <value>Long Island</value>
    <value>Bahamas Beach</value>
    <value>Bungalow</value>
    <value>3200</value>
    <value>4</value>
  </row>
  <row>
    <value>Bahamas Islands</value>
    <value>Nassau</value>
    <value>Bahamas Beach</value>
    <value>Bungalow</value>
    <value>43840</value>
    <value>44</value>
  </row>
  <row>
    <value>Bahamas Islands</value>
    <value>Nassau</value>
    <value>Sun Resort</value>
    <value>Bungalow</value>
    <value>40800</value>
    <value>44</value>
  </row>
  <row>
    <value>Bahamas Islands</value>
    <value>Nassau</value>
    <value>Hawaiian Club</value>
  </row>
</dataset>
```
Example

Chart

The chart displays three data object values.

GET /documents/6405/reports/1/elements/13/dataset?datapath=DP1.DOa6:"New York"

Response (XML):

```xml
<dataset>
  <metadata>
    <value dataObjectId="DP1.DO178" type="String">Store name</value>
    <value dataObjectId="DP1.DObc" type="String">Year</value>
    <value dataObjectId="DP1.DO93" type="Numeric">Sales revenue</value>
  </metadata>
  <row>
    <value>e-Fashion New York 5th</value>
    <value>2004</value>
    <value>644635.1</value>
  </row>
  <row>
    <value>e-Fashion New York 5th</value>
    <value>2005</value>
    <value>1.076144e6</value>
  </row>
  <row>
    <value>e-Fashion New York 5th</value>
    <value>2006</value>
    <value>1.2395874e6</value>
  </row>
  <row>
    <value>e-Fashion New York Magnolia</value>
    <value>2004</value>
    <value>1.0230607e6</value>
  </row>
</dataset>
```
Example

Chart

The chart displays five data object values.

GET /documents/6404/reports/4/elements/16/dataset

Response (JSON):

```json
{"dataset":
{"metadata":
{"@dataObjectId":"DP1.DOda","@type":"String","$":"State"},
{"@dataObjectId":"DP1.DOa6","@type":"String","$":"City"},
{"@dataObjectId":"DP1.DO178","@type":"String","$":"Store name"},
{"@dataObjectId":"DP1.DO93","@type":"Numeric","$":"Sales revenue"},
{"@dataObjectId":"DP1.DO94","@type":"Numeric","$":"Quantity sold"}
},
"row":
["California","Los Angeles","e-Fashion Los Angeles","1.6566757e6",9869],
["California","San Francisco","e-Fashion San Francisco","1.3360033e6",7900],
["Colorado","Colorado Springs","e-Fashion Colorado Springs",843584.2,5116],
["DC","Washington","e-Fashion Washington Tolbooth","1.0535814e6",6491],
["Florida","Miami","e-Fashion Miami Sundance",811923.6,4830],
["Illinois","Chicago","e-Fashion Chicago 33rd","1.1340854e6",6519],
["Massachusetts","Boston","e-Fashion Boston Newbury",887169.2,5269],
["New York","New York","e-Fashion New York 5th","1.2395874e6",7458],
["New York","New York","e-Fashion New York Magnolia","1.9114343e6",11651],
["Texas ","Austin","e-Fashion Austin","1.1354791e6",6919],
["Texas ","Dallas","e-Fashion Dallas",803420.8,4932],
["Texas ","Houston","e-Fashion Houston",910451.2,5419],
["Texas ","Houston","e-Fashion Houston Leighton","1.3357472e6",7923]]
}
```

Related Information

Getting the List of Documents [page 195]
8.4.7 Updating a Report Element

Usage

- You can update the following report elements:
  - Cell
  - Section
  - Table (VTable, HTable, XTable, and Form)
  - PageZone
  - Visualization (charts)
- You can change the type of the following report elements:
  - From a type of Table to another type of Table
  - From a Table to a Visualization
  - From a Visualization to a Table
  - From a type of Visualization to another type of Visualization

Request


Where:

- `<unit>` is an optional parameter of type string that defines the unit of measurement used for all dimensional values such as size, padding, and position. Values are `metric` (default), `inch` and `centimeter`.

Request body: the relevant parameters for the element you want to update. See Charts [page 58] for a description of the Visualizations.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Updating the Axis Expressions

You can also update the axis expressions of a report element by using the following API:

See Updating the Expressions of an Axis [page 383].

Example

Section

PUT /documents/127/reports/1/elements/43

Request body:

```xml
<element type="Section">
  <parentId>2</parentId>
  <content>
    <axes duplicateRowAggregation="true">
      <axis role="Row">
        <expressions>
          <formula dataType="String">=[Year]</formula>
        </expressions>
      </axis>
    </axes>
  </content>
</element>
```

Response:

```xml
<success>
  <message>The resource of type "Report element" with identifier "43" has been successfully updated.</message>
  <id>43</id>
</success>
```

Example

Cell

Request body:

```xml
<element type="Cell">
  <parentId>2</parentId>
  <size minimalWidth="4500" minimalHeight="675" autofitWidth="false" autofitHeight="true" />
  <padding left="75" right="75" top="75" bottom="75"/>
  <content>
    <expression>
      <formula type="Text" dataType="Numeric">=Sum([Sales revenue])</formula>
    </expression>
  </content>
</element>
```

Example

VTable

Request body:

```xml
<element type="VTable">
  <parentId>2</parentId>
```

SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer
Web Intelligence REST API Reference

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
Example

PageZone

Request body:

i Note

Only the following tags are available for PageZone report elements:

- `<id>`
- `<size>` with the `minimalHeight` attribute
- `<hide>` without the associated expression
- `<style>` (<font> and <alignment> tags are not available in a PageZone style.)

```xml
<element type="PageZone">
  <id>1</id>
  <size minimalHeight="8888"/>
  <hide always="true"/>
  <style>
    <border>
      <top thickness="Thin" rgb="#ff0000" style="Dotted"/>
      <left thickness="Thin" rgb="#ff0000" style="Dotted"/>
    </border>
    <background>
      <skin>Curve</skin>
    </background>
  </style>
</element>
```

Example

Chart

Request body:

```xml
<element type="Visualization">
  <parent1Id>2</parent1Id>
  <size minimalWidth="10.84" minimalHeight="6.26"/>
  <position x="0.2" y="0.2"/>
  <style>
    <border>
      <top thickness="None" rgb="#000000" style="None"/>
      <bottom thickness="None" rgb="#000000" style="None"/>
    </border>
  </style>
</element>
```
8.4.8 Inserting a Row or Column into a Table

Usage

Inserts a row or column into a table according to the position of a given cell.

Request

POST /documents/<documentID>/reports/<reportID>/elements/<elementID>?
strip=<strip>&position=<position>

Where:
- `<strip>` is a string parameter that defines the row or column to insert into the table that contains the given cell `<elementID>`. Values are row or column.
- `<position>` is a string parameter that defines the position of the row or column relative to the given cell. Values are:
  - left or right for a column
  - above or below for a row

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.
Example

Inserting a Row Above the Given Cell (JSON)

POST /documents/16654/reports/1/elements/6?strip=row&position=above

```json
{
  "success": {
    "message": "The resource of type "Report element" with identifier "6" has been successfully updated.",
    "id": 6
  }
}
```

8.4.9 Merging or Splitting Cells of a Table

Usage

Merges cells of a table into one cell or splits the previously merged cells.

Request

PUT /documents/<documentID>/reports/<reportID>/elements/<elementID>?
operation=<operation>&targetCellIds=<targetCellIDs>

Where:

- `<operation>` is a mandatory parameter of type string that defines the operation to be applied on the cell. Values are `split` or `merge`.
- `<targetCellIDs>` is a mandatory parameter that lists the identifiers of the cells to be split or merged, separated by a comma.

Operation Rules

**Merge** rules:
- The list of identifiers must contain at least two cells.
- The cells in the list of identifiers must be all inside the same cell matrix.
- The cells must form a rectangle zone in the table.

**Split** rules:
- The cell is inside a cell matrix.
- The cell has the row or column span greater than 1.
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

Merging Two Cells of a Table (XML)

PUT /documents/16654/reports/1/elements/12?operation=merge&targetCellIds=4,6

<success>
  <message>The resource of type "Report element" with identifier "12" has been successfully updated.</message>
  <id>12</id>
</success>

Example

Splitting Three Cells of a Table (JSON)

PUT /documents/16654/reports/1/elements/12?operation=split&targetCellIds=8,10,9

{"success":
  {"message":"The resource of type "Report element" with identifier "12" has been successfully updated.",
   "id":12}
}

8.4.10 Exporting a Report Element

Usage

Exports a report element in one of the following formats:

- HTML
- Zipped HTML
- MHTML (multipart HTML)
- XML
- PDF
- Microsoft Excel 2003
- Microsoft Excel 2007
- CSV
- Image of type BMP, GIF, PNG, or JPEG if the report element is a chart

Caution

You cannot export a whole page zone (header, footer, or body). This generates a 400 HTTP request.
i Note

- If you choose the HTML format, the web service generates the image links. Therefore, the logon token must still be valid when the HTML output is displayed.
- Exports to HTML will be optimized for the end-user browser if you use the User-Agent HTTP header in the REST API call.

Request

GET /documents/<documentID>/reports/<reportID>/elements/<elementID>?
<optional_parameters>

Table 179: Optional Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Supported Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>datapath</td>
<td>Optional. A string that specifies the data path, for example {elementId}? datapath=DP1.DObc:&quot;2004&quot;,DP1.DOa6:&quot;Austin&quot;.</td>
<td>All</td>
</tr>
<tr>
<td>reference</td>
<td>Optional. A string that specifies a reference, for example {elementId}?reference=1.D.7.</td>
<td>All</td>
</tr>
<tr>
<td>dpi</td>
<td>Resolution in dots per inch (dpi) for generated charts. Value between 75 and 9600. Default is 300 for PDF format, 96 for all other formats.</td>
<td>All except XML and CSV</td>
</tr>
<tr>
<td>mode</td>
<td>normal or quickDisplay.</td>
<td>All</td>
</tr>
<tr>
<td>orientation</td>
<td>Page orientation. Use to force a specific page orientation. Values are portrait and landscape.</td>
<td>If mode=normal</td>
</tr>
<tr>
<td>widthScaling</td>
<td>Number of pages per report displaying in width. Default is 0 and means no constraint in width.</td>
<td>If mode=normal</td>
</tr>
<tr>
<td>heightScaling</td>
<td>Number of pages per report displaying in height. Default is 0 means no constraint in height.</td>
<td>If mode=normal</td>
</tr>
<tr>
<td>unit</td>
<td>A string that defines the unit that sizes will be reported in. Values are &quot;metric&quot;, &quot;millimeter&quot;, &quot;point&quot;, and &quot;pixel&quot; (default).</td>
<td>XML</td>
</tr>
<tr>
<td>rawValues</td>
<td>Boolean. Default is false. If true, the raw values and their types are exported with the formatted values.</td>
<td>XML</td>
</tr>
<tr>
<td>chartOutputFormat</td>
<td>Output format for generated chart. Values are: jpeg, bmp, gif, and png (default).</td>
<td>HTML, ZIP, and MHTML</td>
</tr>
<tr>
<td>imageUrl</td>
<td>String used to customize image links when they cannot be reached from the information system.</td>
<td>HTML</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Supported Formats</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>optimized</td>
<td>Boolean. Default is false. If true, the generated output is optimized for calculations inside Microsoft Excel.</td>
<td>Microsoft Excel 2003 and Microsoft Excel 2007</td>
</tr>
<tr>
<td>textQualifier</td>
<td>Character used to surround each column value. Values are ' or &quot; .</td>
<td>CSV</td>
</tr>
<tr>
<td>columnDelimiter</td>
<td>String that defines a character put between columns. Values are comma (,), semi-colon (;) or the special string Tab.</td>
<td>CSV</td>
</tr>
<tr>
<td>charset</td>
<td>String that defines a valid server charset retrieved from the list of charsets.</td>
<td>CSV</td>
</tr>
</tbody>
</table>

⚠️ Caution

You cannot use `datapath` and `reference` parameters in the same URL. This generates a 400 HTTP request.

Customization of Image Links

You may need to customize the source of an image in a report element exported as HTML if the image is unreachable by your system:

```
<img border="0" style="position:absolute;top:0px;left:0px;width:400px;height:300px;"
    X-SAP-LogonToken={X-SAP-LogonToken}">
</img>
```

To do this, use the `imageUrl` parameter with an appropriate value in the GET call:


The resulting image link looks like:

```
<img border="0" style="position:absolute;top:0px;left:0px;width:400px;height:300px;"
    alt="3D Column Chart" src="http://myServer/myPage.jsp?documentId=6406&reportId=
    1&imageId=dxXMLDraft.drilloff_1*2*6&X-SAP-LogonToken={X-SAP-LogonToken}">
</img>
```

Response

Response type:
- text/xml for XML
- text/html for HTML
- application/zip for zipped HTML
- multipart/related for MHTML
- application/pdf for PDF
- application/vnd.ms-excel for Microsoft Excel 2003
- application/vnd.openxmlformats-officedocument.spreadsheetml.sheet for Microsoft Excel 2007
- text/csv for CSV
- image/bmp for BMP images
- image/gif for GIF images
- image/png for PNG images
- image/jpeg or image/jpg for JPEG images
  image/jpg is the returned content type in both cases.

**Example**

**HTML**

"<base_webi_REST_URL>/documents/9227/reports/1/elements/4" > reportelement4.htm

**Example**

**XML**

curl -G -s -H "accept:text/xml" -H X-SAP-LogonToken:""%tokenValue%"
"<base_webi_REST_URL>/documents/9227/reports/1/elements/4" > reportelement4.xml

**Example**

**PDF**


**Example**

**Microsoft Excel 2003**

curl -G -s -H "accept:application/vnd.ms-excel" -H "X-SAP-LogonToken:""%tokenValue%"
"<base_webi_REST_URL>/documents/9227/reports/1/elements/4?dpi=150&optimized=true" > reportelement4.xls

**Example**

**Microsoft Excel 2007**

curl -G -s -H "accept:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
-H X-SAP-LogonToken:""%tokenValue%"
"<base_webi_REST_URL>/documents/9227/reports/1/elements/4" > reportelement4.xxlsx
Example

CSV
The column delimiter in this example is the comma.

```bash
curl -G -s -H "accept:text/csv" -H X=SAP-LogonToken:""%tokenValue%""
 "<base_webi_REST_URL>/documents/9227/reports/1/elements/4?
textQualifier='&columnDelimiter=,&charset='UTF-8'" > reportelement4.csv
```

Example

Image

```bash
curl -G -s -H "accept:image/bmp" -H X=SAP-LogonToken:""%tokenValue%""
```

Related Information

Getting the List of Documents [page 195]
Getting the Charsets [page 246]
Getting the List of Reports [page 290]
Getting the List of Report Elements [page 349]

8.4.11 Deleting a Report Element

Usage

You can delete one of the following report elements:

- Cell (only free cells, not cells in a table)
- Section
- Table (VTable, HTable, XTable, and Form)
- Row or column of a table that contains a given cell
- Visualization (charts)

➤ Remember

- The content of that element is also deleted from the report.
- You cannot delete PageZone elements and table elements of type Cell.
### Request


Where:

- `<strip>` is an optional, string parameter that defines the row or column to remove from the table that contains the given cell `<elementID>`. Values are `row` or `column`. This parameter is mandatory if the cell is in the table header or footer. It is not necessary if the cell is in the table body, because the corresponding row or column will be deleted automatically depending on the table type.

### Response

Response type: `application/xml` or `application/json`

The response is a message stating the success or failure of the request.

**Example**

**Deleting a Table (XML)**

DELETE /documents/5022/reports/3/elements/4

```xml
<success>
  <message>The resource of type "Report element" with identifier "4" has been successfully deleted.</message>
  <id>4</id>
</success>
```

**Example**

**Deleting the Row Containing the Given Cell of a Table Header (JSON)**


```
{"success":{
  "message":"The resource of type "Report element" with identifier "22" has been successfully deleted.",
  "id":22
}}
```

**Example**

**Deleting the Row Containing the Given Cell of a Table Body (XML)**

DELETE /documents/5022/reports/3/elements/23

```xml
<success>
  <message>The resource of type "Report element" with identifier "23" has been successfully deleted.</message>
  <id>23</id>
</success>
```
8.4.12 Working with Axes

Below are the tasks you can perform on the axes of a report element:

- Updating or removing the formulas on axes
- Using breaks on table axes to organize the data according to a selected dimension or measure
- Sorting data on the table axes

The following table shows that the number of axes depends on the type of report element.

<table>
<thead>
<tr>
<th>Report Element</th>
<th>Axis Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section</td>
<td>One axis with identifier 0 that represents a row</td>
</tr>
<tr>
<td>HTable or Form</td>
<td>One axis with identifier 0 that represents a row</td>
</tr>
<tr>
<td>VTable</td>
<td>One axis with identifier 1 that represents a column</td>
</tr>
<tr>
<td>XTable</td>
<td>3 axes with identifiers 0 for the row, 1 for the column, and 2 for the body</td>
</tr>
<tr>
<td>Visualization</td>
<td>Different axes depending on the type of chart. See Roles and Identifiers [page 90].</td>
</tr>
</tbody>
</table>

Updating the Expressions of an Axis [page 383]
Removing the Expressions From an Axis [page 385]
Creating a Table Break [page 386]
Updating a Table Break [page 387]
Deleting a Table Break [page 388]
Updating the Sortings of an Axis [page 389]
Removing the Sortings From an Axis [page 390]

8.4.12.1 Updating the Expressions of an Axis

Usage

Updates the expressions of an axis in a report element of one of the following types:

- Section
- Table (HTable, VTable, XTable, and Form)
Visualization (charts)

Request


Request body: the expressions to update.

⚠️ Restriction

You can update the expressions of standard tables only, otherwise the table formatting will be lost. This feature corresponds to the Assign Data contextual menu used to create tables in SAP BusinessObjects Web Intelligence.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/16995/reports/1/elements/8/axes/0/expressions

Request body:

```xml
<expressions>
  <formula dataType="String">=[Resort]</formula>
  <formula dataType="Numeric">=[Revenue]</formula>
</expressions>
```

Response:

```xml
<success>
  <message>The resource of type "Axis" with identifier "0" has been successfully updated.</message>
  <id>0</id>
</success>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
Getting the List of Report Elements [page 349]
8.4.12.2 Removing the Expressions From an Axis

Usage

Deletes the expressions of an axis in a report element of one of the following types:

- Section
- Table (HTable, VTable, XTable, and Form)
- Visualization (charts)

Request


Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/17170/reports/3/elements/8/axes/1/expressions

<success>
  <message>The resource of type "Axis" with identifier "1" has been successfully updated.</message>
  <id>1</id>
</success>

Related Information

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]
- Getting the List of Report Elements [page 349]
8.4.12.3 Creating a Table Break

Usage

Creates a break in the specified axis of a report element of type table.

👉 Remember

You cannot create breaks in the following cases:
- In a Form table
- In the body of an XTable

Request


Request body: the details of the break.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/17230/reports/3/elements/8/axes/0/breaks

Request body:

```
<break onePage="true" newPage="true" sort="false" duplicate="Repeat"
repeatHeader="true" showHeader="true" showFooter="false">
  <formula dataType="String">= [Year] </formula>
</break>
```

Response:

```
<success>
  <message>The resource of type "Break" with identifier "1" has been successfully created.</message>
  <id>1</id>
</success>
```
8.4.12.4 Updating a Table Break

Usage

Modifies the content of break of an axis in a report element of type table.

Request


Request body: the break to modify.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/17230/reports/3/elements/8/axes/0/breaks/1

Request body:

```
<break onePage="true" newPage="true" sort="false" duplicate="Repeat"
repeatHeader="true" showHeader="true" showFooter="false">
  <formula dataType="String">=[Year]</formula>
</break>
```

Response:

```
<success>
  <message>The resource of type "Break" with identifier "1" has been successfully updated.</message>
  <id>1</id>
</success>
```
8.4.12.5 Deleting a Table Break

Usage

Removes a specific break of an axis in a report element of type table.

Request


Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/17230/reports/3/elements/8/axes/0/breaks/1

Response:

<success>
  <message>The resource of type "Break" with identifier "1" has been successfully removed.</message>
  <id>1</id>
</success>
8.4.12.6 Updating the Sortings of an Axis

Usage

Updates the sortings of an axis in a report element of type Table (HTable, VTable, XTable, and Form), Section, or Visualization.

Request


Request body: the sortings to update.

➤ Remember

The sortings will be applied in the specified order.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/17196/reports/3/elements/12/axes/1/sorts

Request body:

```
<sorts>
  <sort order="Ascending" dataType="String">=[Resort]</sort>
  <sort order="Ascending" dataType="String">=[Country]</sort>
</sorts>
```

Response:

```
<success>
  <message>The resource of type "Axis" with identifier "1" has been successfully updated.</message>
  <id>1</id>
</success>
```
8.4.12.7 Removing the Sortings From an Axis

Usage

Deletes the sortings of an axis in a report element of type Table (HTable, VTable, XTable, and Form), Section, or Visualization.

Request


Response

Response type: application/xml or application/json
The response is a message stating the success or failure of the request.

Example

DELETE /documents/18930/reports/2/elements/15/axes/1/sorts

<succes><message>The resource of type "Axis" with identifier "1" has been successfully updated.</message><id>1</id></success>

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
Getting the List of Report Elements [page 349]

8.4.13 Working with Rankings

Below are specific tasks you can perform on report element rankings.

Ranking allows you to isolate the top and bottom records in a data set based on a variety of criteria.

Creating a Ranking in a Report Element [page 391]
8.4.13.1 Creating a Ranking in a Report Element

Usage

Creates a ranking in a report element of type Table (HTable, VTable, XTable, and Form), Section, or Visualization.

Remember

You can define only one ranking in a report element.

Request

POST /documents/<documentID>/reports/<reportID>/elements/<elementID>/ranking

Request body: the ranking definition.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/17281/reports/1/elements/20/ranking

Request body:

```xml
<ranking calculation="Count" top="3" bottom="3">
  <basedOn>=[Number of guests]</basedOn>
  <rankedBy>=[Year]</rankedBy>
</ranking>
```

Response:

```xml
<success>
  <message>The resource of type "Ranking" has been successfully created.</message>
</success>
```
8.4.13.2 Updating the Ranking of a Report Element

Usage

Modifies a ranking in a report element of type Table (HTable, VTable, XTable, and Form), Section, or Visualization.

Request

PUT /documents/<documentID>/reports/<reportID>/elements/<elementID>/ranking

Request body: the details of the ranking to update.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/17281/reports/1/elements/20/ranking

Request body:

```
<ranking calculation="Count" top="2" bottom="4">
  <basedOn>=[Number of guests]</basedOn>
  <rankedBy>=[Year]</rankedBy>
</ranking>
```

Response:

```
<success>
  <message>The resource of type "Ranking" has been successfully updated.</message>
</success>
```
8.4.13.3 Deleting the Ranking of a Report Element

Usage

Deletes the ranking in a report element of type Table (HTable, VTable, XTable, and Form), Section or Visualization.

Request

DELETE /documents/<documentID>/reports/<reportID>/elements/<elementID>/ranking

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/17281/reports/1/elements/20/ranking

Response:

  <success>
    <message>The resource of type "Ranking" has been successfully removed.</message>
  </success>
8.4.14 Working with Custom Properties

Below are the main tasks you can perform on custom properties of a report element.

- Getting the List of Custom Properties [page 394]
- Getting the Custom Property Value [page 395]
- Adding a Custom Property to a Report Element [page 396]
- Updating a Custom Property [page 397]
- Deleting a Custom Property [page 398]

8.4.14.1 Getting the List of Custom Properties

Usage

Lists all available custom properties of the specified report element.

Request

GET /documents/<documentID>/reports/<reportID>/elements/<elementID>/properties

Response

Response type: application/xml or application/json

Response body: the list of keys and values of custom properties.

Example

GET /documents/1234/reports/1/elements/100/properties

Response:

<properties>
  <property key="KEY01">Value01</property>
  <property key="KEY#2">2nd properties</property>
</properties>
8.4.14.2 Getting the Custom Property Value

Usage

Gets the value of the specified custom property of a report element.

Request

GET /documents/<documentID>/reports/<reportID>/elements/<elementID>/properties/<propertyKey>

Response

Response type: application/xml or application/json

Example

GET /documents/1234/reports/1/elements/100/properties/NEW_KEY

Response:

<property key='NEW_KEY'>MyKeyValue</property>

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
Getting the List of Report Elements [page 349]
8.4.14.3 Adding a Custom Property to a Report Element

Usage

Adds a custom property to a report element.

Request

POST /documents/<documentID>/reports/<reportID>/elements/<elementID>/properties

Request body: the property to add.

⇒ Remember
The property key must not be null or empty.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/1234/reports/1/elements/100/properties

Request body:

<property key="NEW_KEY">MyKeyValue</property>

Response:

<success>
  <message>The resource of type "Property" has been successfully created.</message>
  <id>NEW_KEY</id>
</success>

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
Getting the List of Report Elements [page 349]
8.4.14.4 Updating a Custom Property

Usage

Changes the value of the specified custom property of a report element.

Request

PUT /documents/<documentID>/reports/<reportID>/elements/<elementID>/properties

Request body: the custom property to edit (key and value).

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/1234/reports/1/elements/100/properties

Request body:

<property key='NEW_KEY'>Modified Value</property>

Response:

<success>
  <message>The resource of type "Property" has been successfully updated.</message>
  <id>NEW_KEY</id>
</success>

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
Getting the List of Report Elements [page 349]
8.4.14.5 Deleting a Custom Property

Usage

Deletes a custom property from a report element.

Request

DELETE /documents/<documentID>/reports/<reportID>/elements/<elementID>/properties/<propertyKey>

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/1234/reports/1/elements/100/properties/MYKEY

<success>
  <message>The resource of type "Property" has been successfully removed.</message>
  <id>MYKEY</id>
</success>

Related Information

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]
- Getting the List of Report Elements [page 349]

8.4.15 Managing Data Filters in Report Elements

Below are the tasks you can perform on the data filters that apply to a report element.

- Creating a Data Filter for a Report Element [page 399]
- Getting the Details of a Data Filter for a Report Element [page 400]
8.4.15.1 Creating a Data Filter for a Report Element

Usage

Creates a data filter for a specific report element, referenced by its ID.

Request


Request body: the description of the data filter to create for the report element. The key attribute should be one of the formulaLanguageId elements of the report.

The operator is one of the available report filters:

- Equal
- NotEqual
- Greater
- GreaterOrEqual
- Less
- LessOrEqual
- Between
- NotBetween
- InList
- NotInList
- IsNull
- IsNotNull
- IsAny
- Like
- NotLike
- Both
- Except

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.
## Example

**Request body:**

```xml
<datafilter>
  <condition key="[Lines]" operator="InList">
    <value>City Trousers</value>
    <value>Shirt Waist</value>
    <value>Jackets</value>
  </condition>
  <condition key="[State]" operator="NotEqual">
    <value>Colorado</value>
  </condition>
  <or>
    <condition key="[Sales revenue]" operator="Greater">
      <value>5000</value>
    </condition>
    <condition key="[Sales revenue]" operator="Less">
      <value>10</value>
    </condition>
  </or>
</and>
</datafilter>
```

**Response:**

```xml
<success>
  <message>The resource of type "Report" with identifier "1" has been successfully updated.</message>
  <id>1</id>
</success>
```

## Related Information

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]
- Getting the List of Report Elements [page 349]

### 8.4.15.2 Getting the Details of a Data Filter for a Report Element

#### Usage

Gets the details of a report element data filter from a report.
Request


Response

Response type: application/xml or application/json

```
Example

GET /documents/4567/reports/1/elements/22/datafilter

Response:

<datafilter>
  <and>
    <condition key="[Lines]" operator="InList">
      <value>City Trousers</value>
      <value>Shirt Waist</value>
      <value>Jackets</value>
    </condition>
    <condition key="[State]" operator="NotEqual">
      <value>Colorado</value>
    </condition>
  </and>
  <or>
    <condition key="[Sales revenue]" operator="Greater">
      <value>5000</value>
    </condition>
    <condition key="[Sales revenue]" operator="Less">
      <value>10</value>
    </condition>
  </or>
</datafilter>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Reports [page 290]
Getting the List of Report Elements [page 349]

8.4.15.3 Updating the Data Filter for a Report Element

Usage

Updates the data filter applied to a given report specified in the URL parameter.
Request

PUT /documents/<documentID>/reports/<reportID>/elements/<elementID>/datafilter

Request body: the description of the data filter to create for the report. The operator is one of the available report filter conditions:

- Equal
- NotEqual
- Greater
- GreaterOrEqual
- Less
- LessOrEqual
- Between
- NotBetween
- InList
- NotInList
- IsNull
- IsNotNull
- IsAny
- Like
- NotLike
- Both
- Except

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/223/reports/1/elements/22/datafilter

Request body:

```xml
<datafilter>
  <and>
    <condition key="[Lines]" operator="InList">
      <value>City Trousers</value>
      <value>Shirt Waist</value>
      <value>Jackets</value>
    </condition>
    <condition key="[State]" operator="NotEqual">
      <value>Colorado</value>
    </condition>
  </and>
  <or>
    <condition key="[Sales revenue]" operator="Greater">
      <value>5000</value>
    </condition>
  </or>
</datafilter>
```
Response:

```xml
<success>
  <message>The resource of type "Report" with identifier "1" has been successfully updated.</message>
  <id>1</id>
</success>
```

### Related Information

- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]
- Getting the List of Report Elements [page 349]

### 8.4.15.4 Deleting a Data Filter from a Report Element

#### Usage

Deletes a data filter from a specific report element, referenced by its ID.

#### Request

```
```

#### Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

```
Example

DELETE /documents/9512/reports/67/elements/22/datafilter
```
8.5 Managing Data Providers

Below are the main tasks you can perform on data providers.

Data providers are data sources used to build queries in Web Intelligence documents.

A data provider can be:

- A universe
- A BEx query
- A Microsoft Excel 2003 or Microsoft Excel 2007 file
- A free-hand SQL script

**Note**
Multiflow free-hand SQL scripts are not supported in this release.

The present guide also describes methods specific to:

- Personal data providers based on Microsoft Excel files, see Managing Personal Data Providers [page 448]
- Free-hand SQL data providers, see Managing Connections for Free-Hand SQL Data Providers [page 445]

**Restriction**
The following workflows are not supported:

- Using Analysis Views as data source for a new data provider
- Changing data source when based on Analysis Views and Custom Data Provider

**Related Information**
- Getting the List of Documents [page 195]
- Getting the List of Reports [page 290]
- Getting the List of Report Elements [page 349]
8.5.1 Getting the List of Data Providers

Usage

Gets the list of data providers of a Web Intelligence document.

The list can contain universes (unv and unx), BEx queries, Microsoft Excel files, and free-hand SQL data providers.

Request

GET /documents/<documentID>/dataproviders

Response

Response type: application/xml or application/json

Response body: the list of <dataprovider> elements with the following information:

- <id>
- <name>
- <dataSourceId> is the data source identifier
- <dataSourceType> is the type of data source (unx, unv, bex, excel, and fhsql)
- <updated> is the date of the last update
Example

GET /documents/7738/dataproviders

XML response:

```
<dataproviders>
  <dataprovider>
    <id>DP0</id>
    <name>Query 1</name>
    <dataSourceId>6187</dataSourceId>
    <dataSourceType>unv</dataSourceType>
    <updated>2006-09-20Z</updated>
  </dataprovider>
  <dataprovider>
    <id>DP1</id>
    <name>Query 2</name>
    <dataSourceId>6191</dataSourceId>
    <dataSourceType>unx</dataSourceType>
    <updated>2014-04-23T09:08:20.000+02:00</updated>
  </dataprovider>
  <dataprovider>
    <id>DP2</id>
    <name>Query 3</name>
    <dataSourceId>6120</dataSourceId>
    <dataSourceType>bex</dataSourceType>
    <updated>2014-04-23T09:08:20.000+02:00</updated>
  </dataprovider>
  <dataprovider>
    <id>DP3</id>
    <name>Query 4</name>
    <dataSourceId>6641</dataSourceId>
    <dataSourceType>excel</dataSourceType>
    <updated>2014-04-29T13:37:24.000+02:00</updated>
  </dataprovider>
  <dataprovider>
    <id>DP4</id>
    <name>Query 5</name>
    <dataSourceId>6250</dataSourceId>
    <dataSourceType>fhsql</dataSourceType>
    <updated>2014-08-29T08:32:37.000+02:00</updated>
  </dataprovider>
</dataproviders>
```

JSON response:

```
{"dataproviders":
  {"dataprovider":
   [{"id":"DP0","name":"Query 1","dataSourceId":6187,"dataSourceType":"unv", "updated":"2014-04-23T09:08:20.000+02:00"},
    {"id":"DP1","name":"Query 2","dataSourceId":6191,"dataSourceType":"unx", "updated":"2014-04-23T09:08:20.000+02:00"},
    {"id":"DP2","name":"Query 3","dataSourceId":6120,"dataSourceType":"bex", "updated":"2014-04-23T09:08:20.000+02:00"},
    {"id":"DP3","name":"Query 4","dataSourceId":6641,"dataSourceType":"excel", "updated":"2014-04-29T13:37:24.000+02:00"},
    {"id":"DP4","name":"Query 5","dataSourceId":6250,"dataSourceType":"fhsql", "updated":"2014-08-29T08:32:37.000+02:00"}]
  }
}
```
Related Information

Getting the List of Documents [page 195]

8.5.2 Getting the Details of a Data Provider

Usage

Gets the details of a data provider for a Web Intelligence document.

Request

GET /documents/<documentID>/dataproviders/<dataProviderID>

Response

Response type: application/xml or application/json

Response body: the details of the data provider, with:

- <id>
- <name>
- <dataSourceId>
- <dataSourceType> (unx, unv, bex, excel, or fhsql)
- <dataSourcePrefix>, used as prefix for data source object IDs of .unv universes only
- <updated>, the date and time of the last update
- <ispartial>
- <rowCount>, the number of rows
- The object dictionary, with:
  - <id>
  - <name>
  - <description>
  - <dataSourceObjectId>
  - <formulaLanguageId>, used as key for data filters
  - dataType, qualification, and highPrecision attributes of expression
- The query identifier
- The properties of the data provider if any
**Note**

`highPrecision` is used in `Measure` expression definition to display the measure value with the "decimal floating-point" numeric format. This format, which is defined by the IEEE 754-2008 standard, allows the number of significant digits to go from 15 to 40, thus giving a higher precision to the value. The attribute default value is `false`.

**Example**

GET /documents/18809/dataproduers/DP0

XML response:

```xml
<dataprovider>
  <id>DP0</id>
  <name>Query 1</name>
  <dataSourceId>7599</dataSourceId>
  <dataSourceType>bex</dataSourceType>
  <dataSourcePrefix>DS0</dataSourcePrefix>
  <updated>2012-07-31T15:44:25.000+02:00</updated>
  <duration>1</duration>
  <isPartial>false</isPartial>
  <rowCount>284</rowCount>
  <flowCount>1</flowCount>
  <dictionary>
    <expression dataType="String" qualification="Dimension">
      <id>DP1.DO1</id>
      <name>City</name>
      <description>City</description>
      <dataSourceObjectId>AZ_CITY</dataSourceObjectId>
      <formulaLanguageId>[City]</formulaLanguageId>
    </expression>
    <expression dataType="String" qualification="Dimension">
      <id>DP1.DO1d</id>
      <name>Customer</name>
      <description>Customer</description>
      <dataSourceObjectId>AZ_CUSTOM</dataSourceObjectId>
      <formulaLanguageId>[Query 2].[Customer]</formulaLanguageId>
    </expression>
    <expression dataType="String" qualification="Dimension">
      <id>DP1.DO6</id>
      <name>Sales Person</name>
      <description>Sales Person</description>
      <dataSourceObjectId>AZ_SALES</dataSourceObjectId>
      <formulaLanguageId>[Query 2].[Sales Person]</formulaLanguageId>
    </expression>
    <expression dataType="Numeric" qualification="Measure">
      <id>DP1.DO5c</id>
      <name>Order Amount</name>
      <description>Order Amount</description>
      <dataSourceObjectId>MCWZ5CK839ES5HKKFA52FG78E</dataSourceObjectId>
      <formulaLanguageId>[Query 2].[Order Amount]</formulaLanguageId>
    </expression>
    <expression dataType="Numeric" qualification="Measure">
      <id>DP1.DO5f</id>
      <name>Order Quantity</name>
      <description>Order Quantity</description>
      <dataSourceObjectId>7MRKCI3EB56YK8AL86VY9Q8</dataSourceObjectId>
      <formulaLanguageId>[Query 2].[Order Quantity]</formulaLanguageId>
    </expression>
  </dictionary>
</dataprovider>
```
Related Information

Managing Data Filters in Reports [page 325]
Getting the List of Data Providers [page 405]

8.5.3 Adding a Data Provider

Usage

Adds a new data provider to a Web Intelligence document.

You can add:

- A universe (unv or unx)
- A BEx query
- A Microsoft Excel 2003 or Microsoft Excel 2007 file
- A free-hand SQL script
Request

POST /documents/<documentID>/dataproviders

Request body:

```
<dataprovider>
  <name>
  <dataSourceId>
```

Where:

- `<name>` is the data source name
- `<dataSourceId>` is the data source identifier

Request Body for a Microsoft Excel File

In the case of a Microsoft Excel file, you may add the following `<property>` elements to provide supplementary details on the data source.

```
<properties>
  <property key="selectedSheet">
  <property key="rangeSelectionMode">
  <property key="firstRowAsObjectNames">
  <property key="selectedRange">
```

Table 181: Properties

<table>
<thead>
<tr>
<th>Property Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selectedSheet</td>
<td>Specifies the name of the sheet to be used as data provider. The first sheet is used if the property is not set.</td>
</tr>
<tr>
<td>rangeSelectionMode</td>
<td>Can be:</td>
</tr>
<tr>
<td></td>
<td>• all, if the whole sheet is selected</td>
</tr>
<tr>
<td></td>
<td>• named, if one of the named cell ranges is selected</td>
</tr>
<tr>
<td></td>
<td>• freehand, if the end-user selects the cell range</td>
</tr>
<tr>
<td>firstRowAsObjectNames</td>
<td>Boolean. Default is true. If true, the first row of the sheet contains column names.</td>
</tr>
<tr>
<td>selectedRange</td>
<td>Do not use it if rangeSelectionMode is set to all. Specify one of the named cell ranges if any, or use the end-user input.</td>
</tr>
</tbody>
</table>

**Note**

The present release does not support fragmented named ranges.
Request body for a Free-Hand SQL Script

In the case of a free-hand SQL script, the request body must contain a SQL statement as `<property>`:

```xml
<dataprovider>
  <name>
  <dataSourceId>
  <properties>
    <property key="sql">
```

All SQL statements conform to standards ANSI-SQL 89/92/98 are supported. For example, SET options, stored procedures, and WITH clauses are supported. The @Variable and @Prompt functions for interactive queries are also supported. A SQL script containing DDL commands such as DROP TABLE or ALTER TABLE is not executed. Multiflow SQL statements are not supported. The query will return the first result only.

You may also add the following `<property>` elements to provide supplementary details on the desired results:

```xml
  </properties>
  </property key="maxRows">
  </property key="timeout">
```

Table 182: Properties

<table>
<thead>
<tr>
<th>Property Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxRows</td>
<td>Specifies the maximum rows retrieved. If not set or value is -1, the property is disabled.</td>
</tr>
<tr>
<td>timeout</td>
<td>Specifies the maximum retrieval time (in seconds). If not set or value is -1, the property is disabled.</td>
</tr>
</tbody>
</table>

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

POST /documents/3422/dataprov

Request body for a BEx query:

```xml
<dataprovider>
  <name>Query1</name>
  <dataSourceId>11990;Z_BOBJ;AAQUERY_SAMPLE</dataSourceId>
</dataprovider>
```

Request body for a Microsoft Excel file:

```xml
<dataprovider>
  <name>My Excel data provider</name>
  <dataSourceId>6641</dataSourceId>
  <properties>
    <property key="selectedSheet">mysheet</property>
    <property key="rangeSelectionMode">freehand</property>
    <property key="firstRowAsObjectNames">true</property>
    <property key="selectedRange">A2:F7</property>
```
Request body for a free-hand SQL script:

```xml
<datapropvider>
  <name>FHSQL dp on beachOraJdbc</name>
  <dataSourceId>6250</dataSourceId>
  <properties>
    <property key="sql">SELECT * from country</property>
    <property key='maxRows'>286</property>
    <property key='timeout'>857</property>
  </properties>
</datapropvider>
```

Response:

```xml
<success>
  <message>The resource of type "Data provider" with identifier "DP3" has been successfully created.</message>
  <id>DP3</id>
</success>
```

Related Information

- Getting the List of Documents [page 195]
- Getting the Details of a Microsoft Excel File [page 451]

8.5.4 Updating a Data Provider

Usage

Purges or updates a data provider.

The following table shows what you can update in the particular cases of a data provider based on a Microsoft Excel file or a free-hand SQL script.
Table 183:

<table>
<thead>
<tr>
<th>Data Provider</th>
<th>Possible Updates</th>
</tr>
</thead>
</table>
| Microsoft Excel file     | • The Microsoft Excel file itself  
                            • Properties  
                            • Dictionary expression details:  
                                ○ Data types  
                                ○ Qualification  
                                ○ High Precision  
                                ○ Name  
                                ○ Aggregation functions for measures  
                                ○ Associated dimensions for attributes |
| Free-hand SQL script     | • Properties  
                            • Connections  
                            • SQL script  
                            • Dictionary expression details:  
                                ○ Qualification  
                                ○ High Precision  
                                ○ Name  
                                ○ Aggregation functions for measures  
                                ○ Associated dimensions for attributes |

**Note**
- The replacing Microsoft Excel file must be stored in the CMS repository. Also, it must have the same file structure as the one used as data provider (column number, names, and order).
- The Microsoft Excel file and its properties can be changed through either one request or two distinct requests.
- The connection and SQL script can be changed through either one request or two distinct requests.
- Changes to `<dataSourceObjectId>` and `<formulaLanguageId>` are ignored during update.
- The SDK handles missing result objects, duplicate expression identifier, duplicate name, unknown expression identifier, and modified object order.

**Request**

```
PUT /documents/<documentID>/datap providers/<dataProviderID>?
purge=<purge>&purgeOptions=<purgeOptions>
```

Where:
- `<purge>` (optional) indicates whether to purge the data provider or not. Default is `false`.
- `<purgeOptions>` (optional) is prompts to control the purge operation. If not set or empty, only the data provider will be purged.

**Note**
An error occurs when `purge=false` and `purgeOptions` are both specified in the call.
The request body is optional.

**Request Body to Change the Name of a Data Provider**

```xml
<dataprovider>
  <name>
</name>
</dataprovider>
```

**Request Body to Change the Format of a Measure Value of a Data Provider**

You can set the `highPrecision` attribute of an expression of type `measure` to `true` to display the measure value with the "decimal floating-point" numeric format. This format, which is defined by the IEEE 754-2008 standard, allows the number of significant digits to go from 15 to 40, thus giving a higher precision to the value. You can make this change to any data provider.

```xml
<dataprovider>
  <dictionary>
    <expression highPrecision="Boolean">
      <id>
</expression>
</dictionary>
</dataprovider>
```

If the attribute is not in the request body, the measure is displayed as a usual numeric.

**Request Body to Change Properties of a Microsoft Excel File**

```xml
<dataprovider>
  <property key="selectedSheet"/>
  <property key="rangeSelectionMode"/>
  <property key="firstRowAsObjectNames"/>
  <property key="selectedRange"/>
</dataprovider>
```

You can also add the `isRefreshable` property to the request body to make the data provider refreshable:

```xml
<dataprovider>
  <property key="isRefreshable">true|false</property>
</dataprovider>
```

The `isRefreshable` property value has an impact on the purge operation. If this property is `false` on the Web Intelligence server, the purge is not performed even if `<purge>` is set to `true`.

**Request Body to Change the Microsoft Excel File Used as Data Provider**

```xml
<dataprovider>
  <dataSourceId>
</dataSourceId>
```
Request Body to Change Properties of a Free-Hand SQL Script

```xml
<dataprovider>
  <property key="maxRows">
  </property>
  <property key="timeout">
  </property>
</dataprovider>
```

Request Body to Change the Connection and the SQL Script of a Free-Hand SQL Data Provider

```xml
<dataprovider>
  <dataSourceId>
    <properties>
      <property key="sql">
      </property>
    </properties>
  </dataSourceId>
</dataprovider>
```

<dataSourceId> indicates the connection ID.

Connection and SQL script can be updated in separate requests.

Errors are thrown if:

- Connection and SQL script are not compatible (HTTP error code 400, WSR 00102).
- The end-user is not allowed to edit the SQL script (HTTP error code 401, WSR 00402).

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

**To Purge a Data Provider, but not Prompts**

PUT /documents/7738/datap provid ers/DP0?purge=true

```xml
<success>
  <message>The resource of type "Data provider" with identifier "DP0" has been successfully updated.</message>
  <id>DP0</id>
</success>
```

**To Purge a Data Provider and Prompts**

PUT /documents/7738/datap provid ers/DP0?purge=true&purgeOptions=prompts

```xml
<success>
  <message>The resource of type "Data provider" with identifier "DP0" has been successfully updated.</message>
</success>
```
**Example**

**To Rename a Data Provider**

PUT /documents/7738/dataproviders/DP0

Request body in an XML file:

```xml
<dataprovider>
    <name>My Renamed Data Provider</name>
</dataprovider>
```

Response:

```xml
<success>
    <message>The resource of type "Data provider" with identifier "DP0" has been successfully updated.</message>
</success>
```

**Example**

**To Modify the Free-Hand SQL Data Provider Properties**

PUT /documents/8229/dataproviders/DP0

Request body in an XML file:

```xml
<dataprovider>
    <property key="maxRows">928</property>
    <property key="timeout">101</property>
</dataprovider>
```

Response:

```xml
<success>
    <message>The resource of type "Data provider" with identifier "DP0" has been successfully updated.</message>
</success>
```

**Example**

**To Change the Connection and SLQ Script of a Free-Hand SQL Data Provider**

PUT /documents/8229/dataproviders/DP0

Request body:

```xml
<dataprovider>
    <dataSourceId>6340</dataSourceId>
    <properties>
        <property key="sql">SELECT DISTINCT SALES.INVOICE_DATE FROM SALES</property>
    </properties>
</dataprovider>
```
Response:

```xml
<success>
  <message>The resource of type "Data provider" with identifier "DP0" has been successfully updated.</message>
  <id>DP0</id>
</success>
```

**Example**

To Update the Object Dictionary of a Data Provider Based on a Microsoft Excel File

A first call allows you to get the data provider details:

GET /documents/11837/dataprovers/DP0

Response:

```xml
<dataprovider>
  <id>DP0</id>
  <name>named cell</name>
  <dataSourceId>11835</dataSourceId>
  <dataSourceType>excel</dataSourceType>
  <duration>0</duration>
  <isPartial>false</isPartial>
  <rowCount>0</rowCount>
  <flowCount>1</flowCount>
  <dictionary>
    <expression dataType="String" qualification="Dimension">
      <id>DP0.DO0</id>
      <name>OrderDate</name>
      <dataSourceObjectId>DS0.DO0</dataSourceObjectId>
      <formulaLanguageId>[OrderDate]</formulaLanguageId>
    </expression>
    <expression dataType="String" qualification="Dimension">
      <id>DP0.DO1</id>
      <name>Region</name>
      <dataSourceObjectId>DS0.DO1</dataSourceObjectId>
      <formulaLanguageId>[Region]</formulaLanguageId>
    </expression>
    <expression dataType="String" qualification="Dimension">
      <id>DP0.DO2</id>
      <name>Rep</name>
      <dataSourceObjectId>DS0.DO2</dataSourceObjectId>
      <formulaLanguageId>[Rep]</formulaLanguageId>
    </expression>
    <expression dataType="String" qualification="Dimension">
      <id>DP0.DO3</id>
      <name>Item</name>
      <dataSourceObjectId>DS0.DO3</dataSourceObjectId>
      <formulaLanguageId>[Item]</formulaLanguageId>
    </expression>
    <expression dataType="Numeric" qualification="Measure">
      <id>DP0.DO4</id>
      <name>Units</name>
      <dataSourceObjectId>DS0.DO4</dataSourceObjectId>
      <formulaLanguageId>[Units]</formulaLanguageId>
      <aggregationFunction>Sum</aggregationFunction>
    </expression>
    <expression dataType="Numeric" qualification="Measure">
      <id>DP0.DO5</id>
      <name>Cost</name>
      <dataSourceObjectId>DS0.DO5</dataSourceObjectId>
      <formulaLanguageId>[Cost]</formulaLanguageId>
      <aggregationFunction>Sum</aggregationFunction>
  </dictionary>
</dataprovider>
```
A second call allows you to change:

- The names of DP0.DO0, DP0.DO1, DP0.DO2, DP0.DO3, DP0.DO4, and DP0.DO5.
- The qualification of DP0.DO2 and DP0.DO5.
- The associated dimension of DP0.DO2.
- The aggregation function of DP0.DO4.

```
PUT /documents/11837/dataprovers/DP0
```

Request body:

```xml
<dataprovier>
  <dictionary>
    <expression dataType="String" qualification="Dimension">
      <id>DP0.DO0</id>
      <name>OrderDate1</name>
    </expression>
    <expression dataType="String" qualification="Dimension">
      <id>DP0.DO1</id>
      <name>Region1</name>
    </expression>
    <expression dataType="String" qualification="Attribute">
      <id>DP0.DO2</id>
      <name>Rep1</name>
      <associatedDimensionId>DP0.DO1</associatedDimensionId>
    </expression>
    <expression dataType="String" qualification="Dimension">
      <id>DP0.DO3</id>
      <name>Item1</name>
    </expression>
    <expression dataType="Numeric" qualification="Measure">
      <id>DP0.DO4</id>
      <name>Average</name>
      <aggregationFunction>Average</aggregationFunction>
    </expression>
    <expression dataType="Numeric" qualification="Dimension">
      <id>DP0.DO5</id>
      <name>Cost1</name>
    </expression>
  </dictionary>
</dataprovier>
```

Response:

```xml
<success>
  <message>The resource of type "Data provider" with identifier "DP0" has been successfully updated.</message>
  <id>DP0</id>
</success>
```

In a third call, you check the dictionary is updated. `<formulaLanguageId>` have changed automatically depending on the `<name>` change.
GET /documents/11837/dataproviders/DP0

Response:

```xml
<dataprovider>
  <id>DP0</id>
  <name>named cell</name>
  <dataSourceId>11835</dataSourceId>
  <dataSourceType>excel</dataSourceType>
  <duration>0</duration>
  <isPartial>false</isPartial>
  <rowCount>0</rowCount>
  <flowCount>1</flowCount>
  <dictionary>
    <expression dataType="String" qualification="Dimension">
      <id>DP0.DO0</id>
      <name>OrderDate1</name>
      <dataSourceObjectId>DS0.DO0</dataSourceObjectId>
      <formulaLanguageId>[OrderDate1]</formulaLanguageId>
    </expression>
    <expression dataType="String" qualification="Dimension">
      <id>DP0.DO1</id>
      <name>Region1</name>
      <dataSourceObjectId>DS0.DO1</dataSourceObjectId>
      <formulaLanguageId>[Region1]</formulaLanguageId>
    </expression>
    <expression dataType="String" qualification="Attribute">
      <id>DP0.DO2</id>
      <name>Rep1</name>
      <dataSourceObjectId>DS0.DO2</dataSourceObjectId>
      <formulaLanguageId>[Rep1]</formulaLanguageId>
      <associatedDimensionId>DP0.DO1</associatedDimensionId>
    </expression>
    <expression dataType="Numeric" qualification="Measure">
      <id>DP0.DO4</id>
      <name>Average1</name>
      <dataSourceObjectId>DS0.DO4</dataSourceObjectId>
      <formulaLanguageId>[Average1]</formulaLanguageId>
      <aggregationFunction>Average</aggregationFunction>
    </expression>
    <expression dataType="Numeric" qualification="Dimension">
      <id>DP0.DO5</id>
      <name>Cost1</name>
      <dataSourceObjectId>DS0.DO5</dataSourceObjectId>
      <formulaLanguageId>[Cost1]</formulaLanguageId>
    </expression>
  </dictionary>
  <properties>
    <property key="selectedSheet">Sheet1</property>
    <property key="rangeSelectionMode">freehand</property>
    <property key="selectedRange">A1:F9</property>
    <property key="firstRowAsObjectNames">true</property>
    <property key="isRefreshable">true</property>
  </properties>
</dataprovider>
```
8.5.5 Moving a Data Provider in a Document

Usage

Moves an existing data provider in a Web Intelligence document from a position to another.

Request

PUT /documents/<documentID>/dataproviders?
fromId=<FromDataProviderID>&toId=<ToDataProviderID>

Where:

- <FromDataProviderID> is the identifier of the data provider to move
- <ToDataProviderID> is the identifier of the data provider after move

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/8022/dataproviders?fromId=DP0&toId=DP2

<success>
  <message>Resources of type "Data provider" with identifier "DP0" has been successfully moved.</message>
  <id>DP2</id>
</success>

Related Information

Getting the List of Documents [page 195]
8.5.6 Deleting a Data Provider

Usage

Removes a data provider from the list of data providers of a Web Intelligence document.

Request

DELETE /documents/<documentID>/dataproviders/<dataProviderID>

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

DELETE /documents/7738/dataproviders/DP1

  <success>
    <message>The resource of type "Data provider" with identifier "DP1" has been successfully removed.</message>
    <id>DP1</id>
  </success>

Example

Removing the Last Data Provider

DELETE /documents/7738/dataproviders/DP0

  <error>
    <error_code>101</error_code>
    <message>The resource of type "Data provider" with identifier "DP0" cannot be removed.</message>
  </error>
8.5.7 Changing the Data Providers

You can change the data providers of a Web Intelligence document by replacing data objects from a current data source with data objects from a target data source.

Note

The "change source" functionality does not support text files and Microsoft Excel spreadsheets as data sources.

8.5.7.1 Getting the Possible Object Mappings Using the Default Strategies

Usage

Gets the list of possible object mappings that have been found using the default strategies.

If no data provider is specified, the request is run for all the existing data providers in the document.

Request

GET /documents/<documentID>/dataproviders/mappings?
originDataproviderIds=<dataProviderID[,...]>&targetDatasourceId=<dataSourceID>

Where:
● `<dataProviderID>[,...]>` is the list of data provider identifiers to consider for the mapping, separated by commas. This parameter is optional.

● `<dataSourceID>` is the target data source identifier to use for the proposal mapping. This parameter is mandatory.

**Response**

Response type: application/xml or application/json

Response body: the details of the mappings, described in Mappings Request and Response Body Schemas [page 53].

**Example**

To Get the Possible Mappings

Data provider IDs are DP0 and DP1. Target data source ID is 11414.

```
GET /documents/9460/dataprovers/mappings?
originDataproviderIds=DP0,DP1&targetDatasourceId=11414
```

```
<mappings>
  <content>
    <mapping status="Ok">
      <source>
        <id>DS0.DO12</id>
      </source>
      <target>
        <id>DS1.DO12</id>
      </target>
    </mapping>
    <mapping status="Ok">
      <source>
        <id>DS0.DO13</id>
      </source>
      <target>
        <id>DS1.DO13</id>
      </target>
    </mapping>
    ...
    <mapping status="Ok">
      <source>
        <id>DS0.DO7</id>
      </source>
      <target>
        <id>DS1.DO7</id>
      </target>
    </mapping>
    <mapping status="Ambiguous">
      <source>
        <id>DS0.DO84</id>
      </source>
      <target>
        <id>DS1.DO6</id>
      </target>
    </mapping>
  </content>
</mappings>
```
Example

To Get the Possible Mappings When the Target is a BEx Query

Data provider IDs are DP0 and DP1. Target data source ID is "10326:AAQUERY_RESTRICT_KF".

GET /documents/10554/dataprovers/mappings?
originDataproviderIds=DP0,DP1&targetDatasourceId=10326:AAQUERY_RESTRICT_KF

```
<mappings>
  <content>
    <mapping status="Ok">
      <source>
        <id>AZ_CITY</id>
      </source>
      <target>
        <id>AZ_CITY</id>
      </target>
    </mapping>
    <mapping status="Ambiguous">
      <source>
        <id>MA550T46E04K803277T003LN3M</id>
      </source>
      <target>
        <id>MD4NTN2159J9ONUHLMNB3U</id>
      </target>
    </mapping>
  </content>
</mappings>
```

Related Information

Change Source [page 51]
Getting the List of Documents [page 195]
Getting the List of Data Providers [page 405]

8.5.7.2 Getting the Possible Object Mappings Using Selected Strategies

Usage

Gets the list of possible object mappings that have been found using the given strategies.

These strategies apply in the order specified in the request when the mapping status is "not found", until one object match is found.

If no data provider is specified, the request is run on all the existing data providers in the document.
Request

PUT /documents/<documentID>/dataproviders/mappings?
originDataproviderIds=<dataProviderID>[,...]&targetDatasourceId=<dataSourceID>

Where:

- `<dataProviderID>[,...]` is the list of data provider identifiers to consider for the mapping, separated by commas. This parameter is optional.
- `<dataSourceID>` is the target data source identifier to use for the proposal mapping. This parameter is mandatory.

Request body:

```xml
<policy qualificationTolerance="Low|Normal|High" dataTypeTolerance="Low|Normal|High">
  <strategies mappingSourceIds="string">
    <strategy name="string" targetId="string"/>
  </strategies>
</policy>
```

See Mappings Request and Response Body Schemas [page 53] for request body description.

Response

Response type: application/xml or application/json

Response body: the details of the mappings, described in Mappings Request and Response Body Schemas [page 53].

Example

PUT /documents/7278/dataproviders/mappings?targetDatasourceId=6610

Request body:

```json
{"policy": {
  "strategies": [
    {
      "strategy": [
        {
          "@name": "SamePath",
          "@name": "SameTechnicalName",
          "@name": "SameName",
          "@name": "Removal"
        }
      ],
      "@mappingSourceIds": "DS0.DO2,DS0.DO7,DS0.DO18, DS0.DO4",
      "strategy": [
        {
          "@name": "SameId",
          "@name": "CloseName"
        }
      ],
      "@mappingSourceIds": "DS0.DO25",
      "strategy": {
        "@name": "Selection",
        "@targetId": "M[Measures].[Customer Count]"
      }
    },
    {
      "@mappingSourceIds": "DS0.DO7",
      "strategy": {
        "@name": "Removal"
      }
```
Response:

{"mappings": {
  "policy": {
    "strategies": [
      {
        "strategy": {
          "@name": "SamePath",
          "@name": "SameTechnicalName",
          "@name": "SameName",
          "@name": "Removal"}
      },
      {
        "@mappingSourceIds": "DS0.DO2, DS0.DO7, DS0.DO18, DS0.DO4",
        "strategy": {
          "@name": "SameId",
          "@name": "CloseName"}
      },
      {
        "@mappingSourceIds": "DS0.DO25",
        "strategy": {
          "@name": "Selection",
          "@targetId": "M[Measures].[Customer Count]"
        }
      },
      {
        "@mappingSourceIds": "DS0.DO7",
        "strategy": {
          "@name": "Selection",
          "@targetId": "M[Measures].[Internet Sales Amount]"
        }
      }
    ],
    "content": {
      "mapping": [
        {
          "$status": "Ambiguous",
          "source": {"id": "DS0.DO12"},
          "target": {"id": "L[Sales Territory].[Sales Territory].[Region]"}
        },
        {
          "$status": "Ambiguous",
          "source": {"id": "DS0.DO13"},
          "target": {"id": "A[Customer].[City]"}
        },
        {
          "$status": "NotFound",
          "source": {"id": "DS0.DO17"}
        },
        {
          "$status": "Ambiguous",
          "source": {"id": "DS0.DO18"},
          "target": {"id": "A[Customer].[City]"}
        },
        {
          "$status": "Ambiguous",
          "source": {"id": "DS0.DO2"},
          "target": {"id": "A[Product].[Color]"}
        },
        {
          "$status": "Ambiguous",
          "source": {"id": "A[Product]."}
        }
      ]
    }
  }
}
Related Information

Change Source [page 51]
Getting the List of Documents [page 195]
Getting the List of Data Providers [page 405]

8.5.7.3 Changing the Data Objects of a Data Provider

Usage

Replaces data objects of the current data source with the ones of the target data source using a given data object mapping and a given strategy. The use of custom strategies is new in the 4.1 SP6 release.

Request

POST /documents/<documentID>/dataproviers/mappings?
originDataproviderIds=<dataProviderID>[,...]&targetDatasourceId=<dataSourceID>

Where:
● `<dataProviderID>[,...]>` is the list of data provider identifiers to consider for the mapping, separated by commas. This parameter is optional.

● `<dataSourceID>` is the target data source identifier to use for the proposal mapping. This parameter is mandatory.

Request body: the strategies, the custom mapping, and the parameter values if the document has parameters.

See Mappings Request and Response Body Schemas [page 53] for request body description.

<table>
<thead>
<tr>
<th>Request Content</th>
<th>Object Mapping Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>No mapping and strategies</td>
<td>The mapping found by the default strategy</td>
</tr>
<tr>
<td>Strategies but no mapping</td>
<td>The mapping found by the strategies</td>
</tr>
<tr>
<td>A mapping but no strategies</td>
<td>The mapping overloads the default one</td>
</tr>
<tr>
<td>A mapping and strategies</td>
<td>The mapping overloads the one found by the strategies</td>
</tr>
</tbody>
</table>

**Response**

Response type: `application/xml` or `application/json`

The response is a message stating the success or failure of the request.

**Example**

To Change the Data Source by Providing a Mapping (XML)

POST /documents/9455/datap providers/mappings?
originDataproviderIds=DP0&targetDatasourceId=11414

In the following request body, there are no parameters and all sources are mapped to a target:

```xml
<mappings>
  <content>
    <mapping>
      <source>
        <id>DS0.DO12</id>
      </source>
      <target>
        <id>DS1.DO12</id>
      </target>
    </mapping>
    <mapping>
      <source>
        <id>DS0.DO2</id>
      </source>
      <target>
        <id>DS1.DO2</id>
      </target>
    </mapping>
    <mapping>
      <source>
        <id>DS0.DO6</id>
      </source>
      <target>
        <id>DS1.DO6</id>
      </target>
    </mapping>
  </content>
</mappings>```
To Change the Data Source Where one Source is not Mapped to a Target (XML)

POST /documents/9460/dataproviders/mappings?originDataproviderIds=DP0,DP1&targetDatasourceId=11414

In the following request body, there are no parameters and the last source (DS0.DO6) is not mapped to a target. It will be removed from the mapping.

```xml
<mappings>
  <content>
    <mapping>
      <source>
        <id>DS0.DO12</id>
      </source>
      <target>
        <id>DS1.DO12</id>
      </target>
    </mapping>
    <mapping>
      <source>
        <id>DS0.DO2</id>
      </source>
      <target>
        <id>DS1.DO2</id>
      </target>
    </mapping>
    <mapping>
      <source>
        <id>DS0.DO6</id>
      </source>
      <target/>
    </mapping>
  </content>
</mappings>
```

To Change the Data Source with a Strategy (JSON)

POST 16706/dataproviders/mappings?originDataproviderIds=DP0&targetDatasourceId=6610

Request body:

```json
{
  "mappings":{
```
"policy":{
  "strategies":{
    "strategy":{
      "@name": "SameId"
    }
  }
},
"content":{
  "mapping":{
    "source":{
      "id": "M[Measures].[Sales Amount]"
    },
    "target":{
      "id": "M[Measures].[Internet Sales Amount]"
    }
  }
}
}

Response:

{"success":
  {"message": "The resource of type "Document" with identifier "16706" has been successfully updated."
}}

Example

To Change the Data Source of a Document with Parameters (XML)

Parameters appear in the response as long as they need to be filled in the request body. When all the parameters have values, then the response is successful.

POST /documents/9460/datap provid ers/mappings?
originDataproviderIds=DP0,DP1&targetDatasourceId=11414

Request body:

<mappings>
  <content>
    <mapping>
      <source>
        <id>_Vy_yePzREeG4Q-3y4CsnNg</id>
      </source>
      <target>
        <id>A[Product].[Size]</id>
      </target>
    </mapping>
    <mapping>
      <source>
        <id>_Vy_ye_zREeG4Q-3y4CsnNg</id>
      </source>
      <target>
        <id>A[Employee].[Birth Date]</id>
      </target>
    </mapping>
    <mapping>
      <source>
        <id>_Vy_yfPzREeG4Q-3y4CsnNg</id>
      </source>
      <target>
        <id>
      </target>
  </content>
</mappings>
The response contains the mapping information and a parameter:

```
<mappings>
  <content>
    <mapping>
      <source>
        <id>_Vy_yePzREeG4Q-3y4CsnNg</id>
      </source>
      <target>
        <id>A[Product].[Size]</id>
      </target>
    </mapping>
    <mapping>
      <source>
        <id>_Vy_ye_zREeG4Q-3y4CsnNg</id>
      </source>
      <target>
        <id>A[Employee].[Birth Date]</id>
      </target>
    </mapping>
    <mapping>
      <source>
        <id>_Vy_yfPzREeG4Q-3y4CsnNg</id>
      </source>
      <target>
        <id>A[Reseller].[Bank Name]</id>
      </target>
    </mapping>
  </content>
  <parameters>
    <parameter optional="false" type="prompt" dpId="DP1">
      <id>0</id>
      <technicalName>psEnter Age:</technicalName>
      <name>Enter Age:</name>
      <answer constrained="false" type="Text">
        <info cardinality="Single"/>
      </answer>
    </parameter>
  </parameters>
</mappings>
```

Send the following request with the parameter answer:

POST /documents/9460/dataprovers/mappings?
originDataprovderIds=DP0,DP1&targetDatasourceId=11414

Request body:

```
<mappings>
  <content>
    <mapping>
      <source>
        <id>_Vy_yePzREeG4Q-3y4CsnNg</id>
      </source>
      <target>
        <id>A[Product].[Size]</id>
      </target>
    </mapping>
    <mapping>
      <source>
        <id>_Vy_ye_zREeG4Q-3y4CsnNg</id>
      </source>
      <target>
        <id>A[Employee].[Birth Date]</id>
      </target>
    </mapping>
    <mapping>
      <source>
        <id>_Vy_yfPzREeG4Q-3y4CsnNg</id>
      </source>
      <target>
        <id>A[Reseller].[Bank Name]</id>
      </target>
    </mapping>
  </content>
</mappings>
```
Response:

```
<success>
  <message>The resource of type "Document" with identifier "9460" has been successfully updated.</message>
  <id>9460</id>
</success>
```

Related Information

- Change Source [page 51]
- Getting the List of Documents [page 195]
- Getting the List of Data Providers [page 405]

8.5.8 Getting the Flow Count of a Data Provider

Usage

Gets the flow count of a data provider.
**Request**

GET /documents/<documentID>/dataproviders/<dataProviderID>/flows/count

**Response**

Response type: text/plain

Response body: the number of flows as an integer (from 1 to n).

**Example**

This example retrieves the flow count for data provider ID DPO available for the document ID 7738.

GET /documents/7738/dataproviders/DPO/flows/count

Response

1

**Related Information**

Getting the List of Documents [page 195]
Getting the List of Data Providers [page 405]

8.5.9 Getting the Details of a Flow

**Usage**

Gets the details on the data provider flow specified in XML or CSV format.

**Request**

GET /documents/<documentID>/dataproviders/<dataProviderID>/flows/<flowID>
Response

Response type: text/xml or text/plain

Response body: for one data provider, it retrieves the flow details.

- In XML format, the index cells and their values by row.
- In CSV format, values only.

Example

Plain Text

This example retrieves the first flow (flowId is 0) of the data provider DP0 available for the document ID 7744.

GET /documents/7744/dataproviders/DP0/flows/0

Response (CSV format):

<table>
<thead>
<tr>
<th>Year</th>
<th>State</th>
<th>Sales revenue</th>
<th>Margin</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>California</td>
<td>1704210.8</td>
<td>774893.4</td>
</tr>
<tr>
<td>2001</td>
<td>Colorado</td>
<td>448301.5</td>
<td>203700.6</td>
</tr>
<tr>
<td>2001</td>
<td>DC</td>
<td>693210.6</td>
<td>310356.2</td>
</tr>
<tr>
<td>2001</td>
<td>Florida</td>
<td>405985.1</td>
<td>192479.3</td>
</tr>
<tr>
<td>2001</td>
<td>Illinois</td>
<td>737914.2</td>
<td>348749.8</td>
</tr>
<tr>
<td>2002</td>
<td>California</td>
<td>2782679.5</td>
<td>1076528</td>
</tr>
<tr>
<td>2002</td>
<td>Colorado</td>
<td>768389.5</td>
<td>294482.6</td>
</tr>
<tr>
<td>2002</td>
<td>DC</td>
<td>1215158</td>
<td>457230.6</td>
</tr>
<tr>
<td>2003</td>
<td>California</td>
<td>2992679</td>
<td>1121488.5</td>
</tr>
</tbody>
</table>

Example

XML Format

This example retrieves the first flow (flowId is 0) of the data provider DP0 available for the document ID 12575.

GET /documents/12575/dataproviders/DP0/flows/0

```xml
<DataProviders>
  <DataProvider>
    <Row>
      <Cell INDEX="0">2006</Cell>
      <Cell INDEX="1">12</Cell>
      <Cell INDEX="2">Texas</Cell>
      <Cell INDEX="3">Dallas</Cell>
      <Cell INDEX="4">e-Fashion Dallas</Cell>
      <Cell INDEX="5">43302.1</Cell>
      <Cell INDEX="6">222</Cell>
      <Cell INDEX="7">17829.4</Cell>
    </Row>
    <Row>
      <Cell INDEX="0">2006</Cell>
      <Cell INDEX="1">12</Cell>
      <Cell INDEX="2">Texas</Cell>
      <Cell INDEX="3">Houston</Cell>
      <Cell INDEX="4">e-Fashion Houston</Cell>
      <Cell INDEX="5">55454.6</Cell>
      <Cell INDEX="6">258</Cell>
      <Cell INDEX="7">24614.6</Cell>
    </Row>
  </Row>
</DataProviders>
```
### Related Information

- Getting the List of Documents [page 195]
- Getting the List of Data Providers [page 405]
- Getting the Flow Count of a Data Provider [page 433]

### 8.5.10 Getting the Samples of a Flow

#### Usage

Returns the data samples of a data provider.

In Web Intelligence, the data samples display in the Query Panel.

**Caution**

The following method is only supported for Microsoft Excel and free-hand SQL data providers.

#### Request

```
GET /documents/<documentID>/dataproviders/<dataProviderID>/flows/<flowID>/samples
```

#### Response

**Response type:** application/xml or application/json

The response body contains:

- The list of the column names with identifiers and data type (`<column>`)

```xml
<ROW>
  <CELL INDEX="0">2006</CELL>
  <CELL INDEX="1">12</CELL>
  <CELL INDEX="2">Texas</CELL>
  <CELL INDEX="3">Houston</CELL>
  <CELL INDEX="4">e-Fashion Houston Leighton</CELL>
  <CELL INDEX="5">77237.7</CELL>
  <CELL INDEX="6">366</CELL>
  <CELL INDEX="7">34926.7</CELL>
</ROW>
```
The list of column values for each row of the data samples (<cvalue>)

Example

```xml
<samples>
  <columns>
    <column id="0" type="String">Country</column>
    <column id="1" type="String">Region</column>
    <column id="2" type="String">City</column>
    <column id="3" type="Numeric">Revenue</column>
    <column id="4" type="String">Retailer Name</column>
  </columns>
  <cvalues>
    <cvalue>
      <column id="0">United States</column>
      <column id="1">California</column>
      <column id="2">Los Angeles</column>
      <column id="3">155000</column>
      <column id="4">Bedford</column>
    </cvalue>
    <cvalue>
      <column id="0">United States</column>
      <column id="1">California</column>
      <column id="2">Los Angeles</column>
      <column id="3">120150</column>
      <column id="4">Jones</column>
    </cvalue>
    <cvalue>
      <column id="0">United States</column>
      <column id="1">California</column>
      <column id="2">Los Angeles</column>
      <column id="3">227000</column>
      <column id="4">Smith</column>
    </cvalue>
    ...  
  </cvalues>
</samples>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Data Providers [page 405]
Getting the Flow Count of a Data Provider [page 433]

8.5.11 Getting the Query Plan

Usage

Returns the query plan of the given data provider.
The query plan represents the SQL statements that compose the query.

**Request**

```
GET /documents/<documentID>/dataproviders/<dataProviderID>/queryplan
```

**Response**

**Response type:** application/xml or application/json

Response body: the details of the query plan, which is made of a series of SQL statements combined using joins and combination operators such as UNION, INTERSECT, and MINUS.

**Example**

```
GET /documents/9106/dataproviders/DP2/queryplan

<queryplan custom="false" editable="true">
  <union>
    <fullOuterJoin>
      <statement index="1">SELECT 'FY' || to_char(SALES.invoice_date,'yy'), count(distinct SALES.inv_id) FROM SALES GROUP BY 'FY' || to_char(SALES.invoice_date,'yy')</statement>
      <statement index="2">SELECT 'FY' || to_char(SALES.invoice_date,'yy'), sum(INVOICE_LINE.nb_guests) FROM SALES, INVOICE_LINE, SERVICE_LINE, SERVICE WHERE (SALES.INV_ID=INVOICE_LINE.INV_ID) AND (INVOICE_LINE.SERVICE_ID=SERVICE.SERVICE_ID) AND (SERVICE_LINE.SL_ID=SERVICE_LINE.SL_ID) AND (SERVICE_LINE.service_line = 'Accommodation') GROUP BY 'FY' || to_char(SALES.invoice_date,'yy')</statement>
    </fullOuterJoin>
    <fullOuterJoin>
      <statement index="3">...</statement>
      <statement index="4">...</statement>
    </fullOuterJoin>
  </fullOuterJoin>
  <intersect>
    <fullOuterJoin>
      <statement index="5">...</statement>
      <statement index="6">...</statement>
    </fullOuterJoin>
    <fullOuterJoin>
      <statement index="7">...</statement>
      <statement index="8">...</statement>
    </fullOuterJoin>
  </fullOuterJoin>
  <minus>
    <fullOuterJoin>
      <statement index="9">...</statement>
      <statement index="10">...</statement>
    </fullOuterJoin>
    <fullOuterJoin>
      <statement index="11">...</statement>
      <statement index="12">...</statement>
    </fullOuterJoin>
  </minus>
</union>
```
Example

If the data provider has contexts, they must be resolved first.

GET /documents/8722/dataprovders/DP5/queryplan

Response

```xml
<error>
  <error_code>WSR 00103</error_code>
  <message>Missing contexts for the data provider "DP5".</message>
</error>
```

GET /documents/8722/dataprovders/DP5/parameters?refresh=false

The request body contains an answer to the context:

```xml
<parameters>
  <parameter>
    <id>0</id>
    <answer>
      <values>
        <value id='_VlxrQPzQEeG4Q-3y4CsnNg'>INVOICE_LINE</value>
      </values>
    </answer>
  </parameter>
</parameters>
```

The response shows a parameter of type prompt.

This parameter does not need to be answered at this level:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP5">
    <id>1</id>
    <technicalName>pmEnter value(s) for Service</technicalName>
    <name>Enter value(s) for Service</name>
    <answer constrained="false" type="Text">
      <info cardinality="Multiple">
        <lov hierarchical="false" partial="false" refreshable="true" searchable="true">
          <id>UNIVERSELOV_DS2.DO27</id>
          <updated>2014-08-05T12:41:14.000+02:00</updated>
          <values>
            <value>Activities</value>
            <value>Bungalow</value>
            <value>Car Rent</value>
            <value>Excursion</value>
            <value>Fast Food</value>
            <value>Hotel Room</value>
            <value>Hotel Suite</value>
            <value>Poolside Bar</value>
            <value>Restaurant</value>
            <value>Sports</value>
            <value>Travel Reservation</value>
          </values>
          <columns mappingId="0">
            <column id="0" type="String">Service</column>
          </columns>
        </lov>
        <columns mappingId="0"/>
      </info>
    </answer>
  </parameter>
</parameters>
```
GET /documents/8722/datap provid ers/DP5/queryplan

Response:

```xml
<queryplan custom="false" editable="true">
  <statement index="1">
    SELECT "CITY"."CITY", "SERVICE"."PRICE" FROM "CITY",
    "SERVICE", "INVOICE_LINE", "SALES", "CUSTOMER" WHERE
    ( "INVOICE_LINE"."SERVICE_ID"
      "SERVICE"."SERVICE_ID" ) AND
    ( "SALES"."INV_ID"="INVOICE_LINE"."INV_ID" ) AND
    ( "CUSTOMER"."CUST_ID"="SALES"."CUST_ID" ) AND
    ( "CITY"."CITY_ID"="CUSTOMER"."CITY_ID" )
    AND "SERVICE"."SERVICE" IN @Prompt('Enter value(s) for
    Service', 'A', 'Service\Service',
    Multi,Free,Not_Persistent,,User:0)
  </statement>
</queryplan>
```

Related Information

- Getting the List of Documents [page 195]
- Getting the List of Data Providers [page 405]

8.5.12 Updating the Query Plan

Usage

Updates the query plan of the given data provider.

The query plan represents the SQL statements that compose the query.

Request

PUT /documents/<documentID>/datap provid ers/<dataProviderID>/queryplan

Request body: the details of the query plan.

```xml
<queryplan custom="false" editable="true">
```

→ Remember

You can only update a query plan with editable attribute set to true.
Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

**Note**
The *custom* attribute is automatically set to *true* once the query plan has been modified.

Restoring the Query Plan

You can revert to the original query plan by passing a request body that contains the *custom* attribute set to *false*.

```
<queryplan custom="false">
```

**Example**

PUT /documents/9178/datap providers/DP3/queryplan

Request body:

```
<queryplan>
  <union>
    <fullOuterJoin>
      <statement index="1">SELECT 'FY' || to_char(SALES.invoice_date,'yy'), count( distinct SALES.inv_id) FROM SALES GROUP BY 'FY' ||
      to_char(SALES invoice_date,'yy')</statement>
    </fullOuterJoin>
    <fullOuterJoin>
      <statement index="2">SELECT 'FY' || to_char(SALES.invoice_date,'yy'),
      sum(INVOICE_LINE.nb_guests) FROM SALES, INVOICE_LINE,
      SERVICE_LINE, SERVICE WHERE (SALES.INV_ID=INVOICE_LINE.INV_ID) AND
      (INVOICE_LINE.SERVICE_ID=SERVICE.SERVICE_ID) AND
      (SERVICE.SL_ID=SERVICE LINE.SL_ID) AND
      (SERVICE LINE.service_line = 'Accommodation') GROUP BY 'FY' ||
      to_char(SALES.invoice_date,'yy')"/></statement>
    </fullOuterJoin>
  </union>
</queryplan>
```

Response:

```
<success>
  <message>The resource of type "QueryPlan" has been successfully updated.</message>
```

SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer

Web Intelligence REST API Reference © 2015 SAP SE or an SAP affiliate company. All rights reserved.
### 8.5.13 Getting the Query Specification

#### Usage

Returns the query specification attached to a given data provider.

The query specification describes the parameters and the result objects of the query.

#### Request

```
GET /documents/<documentID>/dataproviders/<dataProviderID>/specification
```

#### Response

Response type: text/xml

Response body: details of the query specification. See the example.

```
    dataProviderId="DP0">
  <queryParameters>
    <duplicatedRowsProperty activated="true" value="true"/>
    <maxRetrievalTimeInSecondsProperty value="300"/>
    <maxRowsRetrievedProperty value="90000"/>
    <removeEmptyRowsProperty activated="true" value="true"/>
    <allowOtherUserToEditQueryProperty activated="true" value="true"/>
    <resetContextOnRefreshProperty activated="true" value="true"/>
    <stripQueryProperty/>
  </queryParameters>
  <queriesTree xsi:type="queryspec:QueryOperatorNode"
      queryOperator="Union">
    <children xsi:type="queryspec:QueryDataNode"></children>
  </queriesTree>
</queryspec:QuerySpec>
```
<bOQuery name="Query" identifier="_ly8aEnwVEeGswMB7H6m1Qw">
  <resultObjects identifier="DS0.DObc" name="Year"/>
  <resultObjects identifier="DS0.DOda" name="State"/>
  <resultObjects identifier="DS0.DOa5" name="Lines"/>
  <resultObjects identifier="DS0.DO93" name="Sales revenue"/>
</bOQuery>
</queriesTree>

<i>Note</i>
In the case of .unv universes, the resultObjects identifier is a string made of the object identifier retrieved from the universe outline and prefixed by the dataSourcePrefix of the current data provider and retrieved from the data provider outline.

**Related Information**
- Getting the List of Documents [page 195]
- Getting the List of Data Providers [page 405]
- Getting the Details of a Data Provider [page 407]
- Getting the Details of a Universe (Web Intelligence) [page 505]

### 8.5.14 Updating the Query Specification

**Usage**

Updates the query specification attached to a data provider.

The query specification describes the parameters and the result objects of the query.

**Request**

PUT /documents/<documentID>/dataproviders/<dataProviderID>/specification

Request type: text/xml
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

PUT /documents/7738/dataproducers/DP0/specification

Request body in an XML file:

```xml
  DataProviderId="DP0">
  <queryParameters>
    <duplicatedRowsProperty activated="true" value="true"/>
    <maxRetrievalTimeInSecondsProperty value="300"/>
    <maxRowsRetrievedProperty value="90000"/>
    <removeEmptyRowsProperty activated="true" value="true"/>
    <allowOtherUserToEditQueryProperty activated="true" value="true"/>
    <resetContextOnRefreshProperty activated="true" value="true"/>
    <stripQueryProperty/>
  </queryParameters>
  <queriesTree xsi:type="queryspec:QueryOperatorNode" queryOperator="Union">
    <children xsi:type="queryspec:QueryDataNode">
      <bOQuery name="Query" identifier="_1y8aENsVEeGswMB7H6m1Qw">
        <resultObjects identifier="DS0.DObc" name="Year"/>
        <resultObjects identifier="DS0.DOda" name="State"/>
        <resultObjects identifier="DS0.DOa5" name="Lines"/>
        <resultObjects identifier="DS0.DO93" name="Sales revenue"/>
        <conditionPart/>
      </bOQuery>
    </children>
  </queriesTree>
</queryspec:QuerySpec>
```

Note

In the case of .unv universes, the resultObjects identifier is a string made of the object identifier retrieved from the universe outline and prefixed by the dataSourcePrefix of the current data provider and retrieved from the data provider outline.

Response:

```
<success>
  <message>The resource of type "Data provider" with identifier "DP0" has been successfully updated.</message>
  <id>DP0</id>
</success>
```
8.6 Managing Connections for Free-Hand SQL Data Providers

Data providers are data sources used to build queries in Web Intelligence documents.

You can build documents directly from SQL scripts on the top of relational secured database connections without using universes. This feature allows you to support rich and complex SQL queries using advanced specific-database functions.

Below are specific tasks you can perform with free-hand SQL scripts as data providers. For more information about data providers, see Managing Data Providers [page 404].

- Getting the List of Connections [page 445]
- Getting the Details of a Connection [page 447]

8.6.1 Getting the List of Connections

Usage

Returns the list of connections stored in the CMS repository that the end-user has access to.

Request

GET /connections?type=<type>&offset=<offset>&limit=<limit>

Where:

- `<type>` indicates the connection type to be retrieved. Values are Relational, FlattenedOlap, Olap or DataFederator. This parameter is optional. If not specified, all types of connections are retrieved.
- `<offset>` indicates the position in the list, from which connections are returned. It must be greater than or equal to 0. The default value is 0. This parameter is optional.
- `<limit>` indicates the number of connections that you can list on one page. Its range is [1, 50]. This parameter is optional. The default value is 10.
Response

Response type: application/xml or application/json

Example

Request Without Parameters

GET /connections

XML response:

```xml
<connections>
  <connection type="Olap">
    <id>6100</id>
    <cuid>AbpsL1MVilFPoM.RfGB02m0</cuid>
    <name>AAQUEY_HIERNODE_01</name>
    <folderId>6069</folderId>
  </connection>
  <connection type="Relational">
    <id>5993</id>
    <cuid>AbL85pdeN7NOqNjdij0bt1A</cuid>
    <name>BOF_ERP_en_UK_Save_LANG_IDT</name>
    <folderId>597I</folderId>
  </connection>
  <connection type="FlattenedOlap">
    <id>6118</id>
    <cuid>ARgmu_R0AQhDvFmrHnPSFO</cuid>
    <name>BW_VAR_FORMULA</name>
    <folderId>6069</folderId>
  </connection>
  <connection type="DataFederator">
    <id>6196</id>
    <cuid>AYATFleY_8Rpt0LWH4Vcfug</cuid>
    <name>DF data source BW</name>
    <folderId>548</folderId>
  </connection>
  ...
</connections>
```

Example

Request With Parameters

GET /connections?limit=3&offset=2&type=FlattenedOlap

JSON response:

```json
{
  "connections": [
    {
      "@type": "FlattenedOlap",
      "id": 6117,
      "cuid": "AZHZlbdSX85LmE3U7dsCl20",
      "name": "BW_VAR_KD_DEFVAL",
      "folderId": 6069,
    },
    {
      "@type": "FlattenedOlap",
      "id": 6112,
      "cuid": "Adr6Epit67FPu_0JebFfgqY",
      "name": "BW_VAR_RANGE",
      "folderId": 6069,
    },
    {
      "@type": "FlattenedOlap",
      "id": 6110,
      "cuid": "A5UAvwV1ptJhuNbzYmdXTw",
    }
  ]
}
```
8.6.2 Getting the Details of a Connection

Usage

Returns the details of a connection stored in the CMS repository.

Request

GET /connections/<connectionID>

Response

Response type: application/xml or application/json

Example

GET /connections/6224

XML response:

```xml
<connection type="Relational">
  <id>6224</id>
  <cuid>AZKgD3.WRiJAhYfy7vjCZ_A</cuid>
  <name>beach</name>
  <folderId>6069</folderId>
  <path>Connections/MyConnections</path>
  <updated>2014-08-06T14:11:02.000+02:00</updated>
  <createdBy>Administrator</createdBy>
  <database>Oracle 10</database>
  <networkLayer>Oracle OCI</networkLayer>
</connection>
```

JSON response:

```json
{"connection":
  {"@type":"Relational",
   "id":6224,
   "cuid":"AZKgD3.WRiJAhYfy7vjCZ_A",
   "name":"beach",
   "folderId":6069,
   "path":"Connections/MyConnections",
   "updated":"2014-08-06T14:11:02.000+02:00",
   ...}
}
```
Related Information

Getting the List of Connections [page 445]

8.7 Managing Personal Data Providers

Data providers are data sources used to build queries in Web Intelligence documents.

You can build documents from Microsoft Excel 2003 or Microsoft Excel 2007 spreadsheets. Before doing so, you need to upload the file to the CMS repository.

Below are specific tasks you can perform with Microsoft Excel files as data providers. For more information about data providers, see Managing Data Providers [page 404].

Uploading a Microsoft Excel File to the CMS Repository [page 448]
Getting the List of Microsoft Excel Files [page 450]
Getting the Details of a Microsoft Excel File [page 451]
Updating a Microsoft Excel File to the CMS Repository [page 453]
Deleting a Microsoft Excel File [page 454]

8.7.1 Uploading a Microsoft Excel File to the CMS Repository

Usage

Uploads and stores a Microsoft Excel file to the CMS repository.

Request

POST /spreadsheets
Request type: multipart/form-data
The request body is a multipart body made of the following parts:

- The file details, which can be XML or JSON. Only `name`, `folderId` elements are mandatory. `description` is optional. The other spreadsheet elements are ignored.
- The file itself as a binary stream. `Content-Type` is either `application/vnd.ms-excel` for Microsoft Excel 2003, or `application/vnd.openxmlformats-officedocument.spreadsheetml.sheet` for Microsoft Excel 2007.

**Note**
Make sure the request body has the necessary empty lines.

**Response**

**Response type:** `application/xml` or `application/json`

The response is a message stating the success or failure of the request.

**XML request body for a Microsoft Excel 2007 file:**

```plaintext
--10f3211b-66e8-4b14-93ec-b11ec5c19a43-41d3c84c-d157-4377-aba9-41353b9a4dfe
Content-Disposition: form-data; name="attachmentInfos"
Content-Type: application/xml
// mandatory carriage return
<spreadsheet>
  <name>myfile.xlsx</name>
  <folderId>6206</folderId>
</spreadsheet>
// mandatory carriage return
--10f3211b-66e8-4b14-93ec-b11ec5c19a43-41d3c84c-d157-4377-aba9-41353b9a4dfe
Content-Disposition: form-data; name="attachmentContent"
Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
// mandatory empty line
... Excel 2007 document as a binary stream...
// mandatory carriage return
--10f3211b-66e8-4b14-93ec-b11ec5c19a43-41d3c84c-d157-4377-aba9-41353b9a4dfe--
```

**XML response:**

```xml
<success>
  <message>The resource of type "Spreadsheet" with identifier "7466" has been successfully created.</message>
  <id>7466</id>
</success>
```

**JSON request body for a Microsoft Excel 2003 file:**

```plaintext
--10f3211b-66e8-4b14-93ec-b11ec5c19a43-41d3c84c-d157-4377-aba9-41353b9a4dfe
Content-Disposition: form-data; name="attachmentInfos"
Content-Type: application/json
// mandatory carriage return
{"spreadsheet":{"name":"myfile.xls","folderId":6206}}
// mandatory carriage return
--10f3211b-66e8-4b14-93ec-b11ec5c19a43-41d3c84c-d157-4377-aba9-41353b9a4dfe--
```

**JSON request body for a Microsoft Excel 2003 file:**

```plaintext
--10f3211b-66e8-4b14-93ec-b11ec5c19a43-41d3c84c-d157-4377-aba9-41353b9a4dfe
Content-Disposition: form-data; name="attachmentInfos"
Content-Type: application/json
// mandatory carriage return
{"spreadsheet":{"name":"myfile.xls","folderId":6206}}
// mandatory carriage return
--10f3211b-66e8-4b14-93ec-b11ec5c19a43-41d3c84c-d157-4377-aba9-41353b9a4dfe--
```
8.7.2 Getting the List of Microsoft Excel Files

**Usage**

Returns the Microsoft Excel files stored in the CMS repository.

**Request**

GET /spreadsheets?offset=<offset>&limit=<limit>

Where:
- `<offset>` indicates the position in the list, from which documents are returned. It must be greater than or equal to 0. The default value is 0. This parameter is optional.
- `<limit>` indicates the number of documents in the list. Its range is [0, 50]. The default value is 10. This parameter is optional.

**Response**

Response type: application/xml or application/json

Response body: the list of files. Each `<spreadsheet>` element has the following children:
- `<id>`: identifier of the file on the CMS repository
- `<cuid>`: CUID of the file on the CMS repository
- `<name>`: name of the file
- `<description>`: description of the file
- `<folderId>`: identifier of the folder of the CMS repository that contains the file
8.7.3 Getting the Details of a Microsoft Excel File

Usage

Returns the details of a Microsoft Excel file stored on the CMS repository.
Request

GET /spreadsheets/<spreadsheetID>

Response

Response type: application/xml or application/json

Response body: the details of the file with the following information:

- <id>: file identifier
- <cuid>: CUID of the file
- <name>: name of the file
- <folderId>: identifier of the folder of the CMS repository that contains the file
- <path>: path to the file in the CMS repository directory
- <updated>: date of last update
- <createdBy>: name of spreadsheet creator
- <file>: name of the file with an extension
- <size>: size of the file in bytes
- <mimeType>: mime type (application/vnd.ms-excel for Microsoft Excel 2003 or application/vnd.openxmlformats-officedocument.spreadsheetml.sheet for Microsoft Excel 2007)
- <sheet>: name of the sheet contained in the file
- <range>: name of a named cell range in the file

Example

XML response:

```xml
<spreadsheet>
  <id>7168</id>
  <cuid>AQJqfTLbR5NRnYhzV9DnCMA</cuid>
  <name>myfile</name>
  <folderId>6206</folderId>
  <path>Public Folders/Documents/</path>
  <updated>2014-07-17T14:27:44.367+02:00</updated>
  <createdBy>Administrator</createdBy>
  <file>myfile.xlsx</file>
  <size>22444</size>
  <mimeType>application/vnd.ms-excel</mimeType>
  <sheets>
    <sheet>sheet1</sheet>
    <sheet>sheet2</sheet>
    <sheet>sheet3</sheet>
    <sheet>sheet4</sheet>
  </sheets>
  <namedRanges>
    <range>items</range>
    <range>region</range>
    <range>region_item_units_cost</range>
    <range>sum_cost</range>
    <range>sum_units</range>
  </namedRanges>
</spreadsheet>
```
Related Information

Getting the List of Microsoft Excel Files [page 450]

8.7.4 Updating a Microsoft Excel File to the CMS Repository

Usage

Re-uploads a Microsoft Excel file stored on the CMS repository with a different content.

👉 Remember

The current file and the one to upload must have the same file structure (columns number, names, and order).

Request

PUT /spreadsheets/<spreadsheetID>

The request type is:

- application/vnd.ms-excel for Microsoft Excel 2003
The request body only contains the Microsoft Excel file as a binary stream. You do not need to send the file details as attachment again.

Note
Make sure the request body has the necessary empty lines.

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

XML request body for a Microsoft Excel 2007 file:

... Excel 2007 document as a binary stream...

XML response:

<success>
  <message>The resource of type "Spreadsheet" with identifier "7466" has been successfully updated.</message>
  <id>7466</id>
</success>

8.7.5 Deleting a Microsoft Excel File

Usage

Deletes a Microsoft Excel file stored on the CMS repository.

Request

DELETE /spreadsheets/<spreadsheetID>
Response

Response type: application/xml or application/json

Response body: The response is a message stating the success or failure of the request.

Example

DELETE /spreadsheets/8995

  <success>
    <message>The resource of type "Spreadsheet" with identifier "8995" has been successfully removed.</message>
    <id>8995</id>
  </success>

Related Information

Getting the List of Microsoft Excel Files [page 450]

8.8 Managing SAP BW Connections and BEx Queries

Below are the tasks you can perform to manage SAP BW connections and BEx queries.

- Getting the List of SAP BW Connections [page 455]
- Getting the Details of an SAP BW Connection [page 457]
- Browsing the Details of an SAP BW connection [page 458]
- Getting the Outline of a BEx Query [page 460]
- Getting the Capabilities of a BEx Query [page 462]

8.8.1 Getting the List of SAP BW Connections

Usage

Returns the list of available SAP BW connections. You can only see the connections that you have authorization to see.
Request

GET /bwconnections?offset=<offset>&limit=<limit>

Where:

- `<offset>` is the offset from the beginning of the list. Default value is 0.
- `<limit>` is the maximum number of connections to return. Default value is 10.

Response

Response type: application/xml or application/json

Response body: the list of BW connections that you have the authorization to see/access.

Example

```xml
<bwconnections>
  <bwconnection type="Cube">
    <id>7052</id>
    <cuid>AdDDU67.DyxBk0gpzjDjn30</cuid>
    <name>Adventure Works MSAS2005</name>
    <folderId>7131</folderId>
  </bwconnection>
  <bwconnection type="Cube">
    <id>11540</id>
    <cuid>AbUAJ7zVpZFgy2jQGMyI8</cuid>
    <name>AdventureWorks</name>
    <folderId>11484</folderId>
  </bwconnection>
  <bwconnection type="Cube">
    <id>11537</id>
    <cuid>AROnvBXnltOpsXj_iPjTo</cuid>
    <name>ADW</name>
    <folderId>11484</folderId>
  </bwconnection>
  ...
  <bwconnection type="Query">
    <id>7268</id>
    <cuid>AUEwMr2ZfzVKjnfwFTWVxvM</cuid>
    <name>BICS_Query</name>
    <folderId>4066</folderId>
  </bwconnection>
  <bwconnection type="System">
    <id>7039</id>
    <cuid>AY2QVyb3WRLioC8Gbg5VIO</cuid>
    <name>BICS_Server</name>
    <folderId>4066</folderId>
  </bwconnection>
</bwconnections>
```
8.8.2 Getting the Details of an SAP BW Connection

Usage

Returns the details of an SAP BW connection.

Request

GET /bwconnections/<bwConnectionID>

Response

Response type: application/xml or application/json

Response body: the details of an SAP BW connection that you have the authorization to access.

<path> is the connection path in the CMS repository.

Example

SAP BW Connection of Type Query

GET /bwconnections/11489

```
<bwconnection type="Query">
  <id>11489</id>
  <cuid>AXRu2fNiQphAtF4041J40Kg</cuid>
  <name>bex_simple</name>
  <folderId>11484</folderId>
  <path>Application Folder/Root Folder/Connections/</path>
  <bwnodes>
    <bwnode type="Query">
      <id>11489</id>
      <name>A simple Bex Query</name>
      <technicalName>QRY_SIMPLE</technicalName>
      <path>/QRY_SIMPLE</path>
    </bwnode>
  </bwnodes>
</bwconnection>
```

Example

Incomplete SAP BW Connection of Type System or Cube

GET /bwconnections/11990

```
<bwconnection type="System">
  <id>11990</id>
  <cuid>Aaj0N_I.bSJElyDkk08sxTU</cuid>
  <name>raylight_BOF</name>
  <folderId>11484</folderId>
</bwconnection>
```
<path>Application Folder/Root Folder/Connections/</path>
</bwnodes>
  <bwnode type="Favorites">
    <name>Favorites</name>
    <technicalName>SystemFavoritesTopLevel</technicalName>
    <path>/SystemFavoritesTopLevel</path>
  </bwnode>
  <bwnode type="InfoArea">
    <name>InfoArea</name>
    <technicalName>SystemInfoareaTopLevel</technicalName>
    <path>/SystemInfoareaTopLevel</path>
  </bwnode>
</bwnodes>
</bwconnection>

i Note
Complete the connection by searching for a BW node of type Query in the system or cube.

Related Information

Getting the List of SAP BW Connections [page 455]

8.8.3 Browsing the Details of an SAP BW connection

Usage

Browses the details of an incomplete SAP BW connection and helps you choose a BEx query which will complete the connection.

The "by path" and "by search pattern" browsing methods are available.

i Note
Only MDX compliant BEx queries are usable in Web Intelligence workflows.

Request

PUT /bwconnections/<bwConnectionID>

Request body (optional):

- By path:
  
  <bwnode>
    <path>[path/to/the/required/node]</path>
  </bwnode>
By search pattern:

```
<bnode>
  <pattern>[pattern to be used for the search]</pattern>
</bnode>
```

Response

Response type: application/xml or application/json

Response body: the list of BW connections that you have the authorization to access.

Note

A BEx query identifier is defined by the concatenation of the BW connection identifier and the technical name of the BEx query, separated by a semi-colon (;).

Example

1: To get the list of BEx queries inside a BW connection, browsing by path

PUT /bwconnections/11990

Request body in an XML file:

```
<bnode>
  <path>/SystemInfoareaTopLevel/BUSINESSOBJECTS_QA/Z_BOBJ/TEST_DATE</path>
</bnode>
```

Response:

```
<bwconnection type="System">
  <id>11990</id>
  <cuid>AajON_I.bSJElyDKk08sxTU</cuid>
  <name>raylight_BOF</name>
  <folderId>11484</folderId>
  <bwnodes>
    <bwnode type="Query">
      <id>11990;TEST_DATE</id>
      <name>test_data</name>
      <technicalName>TEST_DATE</technicalName>
      <mdxCompliant>true</mdxCompliant>
      <path>/SystemInfoareaTopLevel/BUSINESSOBJECTS_QA/Z_BOBJ/TEST_DATE</path>
    </bwnode>
  </bwnodes>
</bwconnection>
```

Example

To get the list of BEx queries inside a BW connection matching a pattern

PUT /bwconnections/11990
8.8.4 Getting the Outline of a BEx Query

Usage

Returns the details of a BEx query.

Request

PUT /bwconnections/<bwConnectionID>/outline

Request body (optional):

```
<bwnode>
</bwnode>
```
Where:

- `<id>` is the BEx query identifier

Response

Response type: application/xml or application/json

Response body: the outline of a BEx query that you have the authorization to see/access.

Example

PUT /bwconnections/11990/outline

Request body in an XML file:

```
<bwnode>
  <id>11990;ROLE_ST_BEX5</id>
</bwnode>
```

Response:

```
<outline>
  <item type="Dimension">
    <name>Country</name>
    <description>Country</description>
  </item>
  <item type="Hierarchy">
    <name>Country</name>
    <description>Country</description>
    <id>HZ_COUNTRY</id>
  </item>
  <item type="Hierarchy">
    <name>Country Hierarchy 01</name>
    <description>Country Hierarchy 01</description>
    <item type="Level">
      <name>Level 01</name>
      <description>Level 01</description>
      <id>LCOUNTRY_HIERARCHY_01|Z_COUNTRY.#1</id>
    </item>
    <item type="Level">
      <name>Level 02</name>
      <description>Level 02</description>
      <id>LCOUNTRY_HIERARCHY_01|Z_COUNTRY.#2</id>
    </item>
    <item type="Level">
      <name>Level 03</name>
      <description>Level 03</description>
      <id>LCOUNTRY_HIERARCHY_01|Z_COUNTRY.#3</id>
    </item>
    <id>H_COUNTRY_HIERARCHY_01|Z_COUNTRY</id>
  </item>
  ...
  <item type="Measure">
    <name>Voyager Training 01 - Restricted KF</name>
    <description>Voyager Training 01 - Restricted KF</description>
    <item type="Attribute">
      <name>Voyager Training 01 - Restricted KF Currency</name>
      <id>MD4NUM119ATNWC7USQ4XYSAOQY.Currency</id>
    </item>
  </item>
</outline>
```
8.8.5 Getting the Capabilities of a BEx Query

Usage

Returns the capabilities of a BEx query.

Request

PUT /bwconnections/<bwConnectionID>/capabilities

Request body (optional):

```
<bwnode>
  <id>
```

Where:

- `<id>` is the BEx query identifier

Response

Response type: application/xml or application/json

Response body: the capabilities of a BEx query that you have the authorization to see/access.

Example

PUT /bwconnections/11990/capabilities

Request body in an XML file:

```
<bwnode>
```
Response:

```xml
  <generalCapability customQueryScriptSupported="false" showHideScopeSupported="false" />
  <dataProcessingCapability removeEmptyRowsAvailable="true" maxRetrievalTimeAvailable="false" queryStrippingAvailable="true" />
  <filterCapability resultHierarchyInFilterSupported="false" constantComparisonSupported="true" hierarchyConstantOperandSupported="true" supportedComparisonOperators="equal" supportedComparisonOperators="notEqual" ...
  <supportedValueBasedHierarchyComparisonOperators>equal</supportedValueBasedHierarchyComparisonOperators>
  <supportedValueBasedHierarchyComparisonOperators>inList</supportedValueBasedHierarchyComparisonOperators>
  <supportedLevelBasedHierarchyComparisonOperators>equal</supportedLevelBasedHierarchyComparisonOperators>
  <supportedLevelBasedHierarchyComparisonOperators>inList</supportedLevelBasedHierarchyComparisonOperators>
  <supportedLogicalOperators>and</supportedLogicalOperators>
  <supportedObjects>attribute</supportedObjects>
  <supportedObjects>dimension</supportedObjects>
  <supportedObjects>hierarchy</supportedObjects>
  <supportedObjects>level</supportedObjects>
</filterCapability>
  <resultObjectCapability useAttributeSeparatelyAvailable="true" buildLogicalSetWithMetadataFunctionsAvailable="true" buildLogicalSetWithMemberFunctionsAvailable="true" memberSelectionPromptSupported="true" supportsMemberSelectionCompletion="true" />
</datasource:QueryCapability>
```

Related Information

Getting the List of SAP BW Connections [page 455]

8.9 Refreshing Documents

Below are the refresh operations you can do on a Web Intelligence document:

- Refreshing a document without contexts and prompts
- Identifying contexts and prompts of a document
- Fill in contexts and prompts with values

Supported prompts are the following:

- Prompts that accept either string, numeric or date values
Prompts that accept one value or multiple values
Optional and non optional prompts

- Getting the Refresh Parameters of a Document [page 464]
- Refreshing a Document [page 468]
- Cancelling the Refresh of a Document [page 485]
- Getting the Details of a Parameter [page 486]

8.9.1 Getting the Refresh Parameters of a Document

Usage

Returns the parameters to be filled before running a document refresh.

Request

GET /documents/<documentID>/parameters?lovInfo=<lovInfo>

Where:

- <lovInfo> is an optional, Boolean parameter. Default value is true. If set to false, the lists of values are not computed, nor displayed.

Response

Response type: application/xml or application/json

Response body: the response provides the parameters with their expected answers, previous values if applicable, otherwise default values. See Parameter Response Body Schemas [page 95]

Example

No Parameter
Response:

```
<parameters/>
```

Example

Context
The parameter of type context provides two possible context values.
Response:

```xml
<parameters>
  <parameter optional="false" type="context" dpId="DP0">
    <id>0</id>
    <technicalName>0</technicalName>
    <name>Select a context</name>
    <answer constrained="true" type="Text">
      <info cardinality="Multiple">
        <lov hierarchical="false" partial="false" refreshable="true">
          <values>
            <value id="2">Reservations</value>
            <value id="1">Sales</value>
          </values>
        </lov>
        <previous>
          <value id="2">Reservations</value>
        </previous>
      </info>
      <values>
        <value id="2">Reservations</value>
      </values>
    </answer>
  </parameter>
</parameters>
```

**Example**

**DateTime Prompt**

The parameter of type `prompt` accepts only one answer value (cardinality `Single`).

Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP01">
    <id>0</id>
    <name>Enter Open Date:</name>
    <answer constrained="false" type="DateTime">
      <info cardinality="Single" keepLastValues="true">
        <previous>
          <value>1992-09-03T17:15:00.000+02:00</value>
        </previous>
      </info>
      <values>
        <value>1992-09-03T17:15:00.000+02:00</value>
      </values>
    </answer>
  </parameter>
</parameters>
```

**Example**

**Prompt with Multiple Values**

The parameter of type `prompt` accepts multiple answer values (cardinality `Multiple`).
Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP0">  
    <id>0</id>
    <technicalName>Enter values for Customer:</technicalName>
    <name>Enter values for Customer:</name>
    <answer constrained="false" type="Text">  
      <info cardinality="Multiple" keepLastValues="true">
        <lov hierarchical="false" partial="true" refreshable="true" searchable="false">
          <id>UNIVERSELOV_DS0.DOd</id>
          <intervals>  
            <interval id="0">  
              <value>Arai</value>
              <value>Okumura</value>
            </interval>
            <interval id="1">  
              <value>Oneda</value>
              <value>Wilson</value>
            </interval>
          </intervals>  
          <values>  
            <value>Arai</value>  <!-- possible values -->
            <value>Baker</value>
            <value>Brendt</value>
            . . .
            <value>Okumura</value>
          </values>  
          <columns mappingId="0">  
            <column id="0" type="String">Customer </column>  
          </columns>  
        </lov>  
        <previous>  
          <value>Arai</value>  <!-- previous values -->
          <value>Baker</value>
          <value>Brendt</value>
          <value>Diemers</value>
          <value>Dupont</value>
          <value>Durnstein</value>
          <value>Edwards</value>
        </previous>
      </info>
      <values>  
        <value>Arai</value>  <!-- previous values -->
        <value>Baker</value>
        <value>Brendt</value>
        <value>Diemers</value>
        <value>Dupont</value>
        <value>Durnstein</value>
        <value>Edwards</value>
      </values>
    </answer>
  </parameter>
  . . .
</parameters>
```

**Note**

(interval)/ are not included in the details of the list of values when there is only one interval.
Example

Prompt with Multiple Columns

The parameter of type prompt accepts one answer value (cardinality Single) from a list of values made of three columns. The first one (mappingId="0") is used as reference.

Response:

```xml
<parameter optional="false" type="prompt" dpId="DP15">
  <id>6</id>
  <technicalName>Enter values for Customer</technicalName>
  <name>Enter values for Customer</name>
  <answer constrained="false" type="Text">
    <info cardinality="Single" keepLastValues="true">
      <lov hierarchical="false" partial="false" refreshable="true" searchable="true">
        <id>UNIVERSELLOV_DS2.DOea</id>
        <intervals>
          <interval id="0">
            <cvalue>
              <column id="0">William</column>
              <column id="1">64</column>
              <column id="2">1995-12-19T02:00:00.000+01:00</column>
            </cvalue>
            <cvalue>
              <column id="0">Silke</column>
              <column id="1">63</column>
              <column id="2">1994-03-25T02:00:00.000+01:00</column>
            </cvalue>
          </interval>
          <interval id="1">
            . . .
          </interval>
        </intervals>
        <cvalues>
          <cvalue>
            <column id="0">Werner</column>
            <column id="1">42</column>
            <column id="2">1995-06-08T08:28:00.000+02:00</column>
          </cvalue>
          . . .
          <cvalue>
            <column id="0">Tony</column>
            <column id="1">55</column>
            <column id="2">1995-07-05T04:00:00.000+02:00</column>
          </cvalue>
        </cvalues>
        <columns mappingId="0">
          <column id="0" type="String">Customer</column>
          <column id="1" type="Numeric">Age</column>
          <column id="2" type="DateTime">Invoice Date</column>
        </columns>
      </lov>
      <previous>
        <value>Andre</value>
      </previous>
    </info>
    <values>
      <value>Andre</value>
    </values>
  </answer>
</parameter>
```
8.9.2 Refreshing a Document

Usage

Refreshes a Web Intelligence document by filling the refresh parameters if needed and running the query.

You can ask for the refresh without providing any parameters (no request body). In this case, the web service returns the context or prompt that needs to be filled. If no parameter has to be filled, the document is refreshed.

Request

PUT /documents/<documentID>/parameters?<optional_parameters>

Where:

Table 185: Optional Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;lovInfo&gt;</td>
<td>Optional, Boolean parameter. Default value is true. If set to false, the lists of values are not computed, nor displayed.</td>
</tr>
<tr>
<td>&lt;refresh&gt;</td>
<td>Optional, Boolean parameter. Default value is true. To avoid a data refresh at the end of the parameter workflow.</td>
</tr>
</tbody>
</table>

Request body: the answers to the parameters retrieved using the GET .../parameters call. See Answer Request Body Schemas [page 104]

Response

Response type: application/xml or application/json

When all parameters have been answered, the last PUT call returns a message stating the success of the request.

If refresh is true, then the message is:

```
<success>
  <message>The resource of type 'Document' with identifier 'XX' has been successfully updated.</message>
  <id>XX</id>
</success>
```
If `refresh` is `false`, then the message is:

```xml
<success>
  <message>The resource of type 'Document' with identifier 'XX' has not been modified.</message>
  <id>XX</id>
</success>
```

Example - Refreshing a Document with a Context [page 469]
Example - Refreshing a Document with Context and Prompt Parameters [page 469]
Example - Refreshing a Document by Using a Query [page 471]
Example - Refreshing a Document by Using a Query with a Search Pattern [page 472]
Example - Refreshing a Document with a Hierarchical Parameter [page 474]
Example - Refreshing a Document with a Hierarchical Parameter of Multiple Columns [page 476]
Example - Refreshing a Document with Hierarchical Indexed Lists of Values [page 480]
Example - Refreshing a Document with Cascading Parameters [page 483]

### 8.9.2.1 Example - Refreshing a Document with a Context

In the `PUT` call, the request body contains an answer to the context. The following request body does not show the `<info>` element, which is not mandatory:

```xml
<parameters>
  <parameter optional="false" type="context">
    <id>0</id>
    <name>Select a context</name>
    <answer constrained="true" type="Text">
      <values>
        <value id="2">Reservations</value>
      </values>
    </answer>
  </parameter>
</parameters>
```

The `PUT` call returns a successful message.

```xml
<success>
  <message>The resource of type 'document' with identifier 'xxx' has been successfully updated.</message>
  <id>xxx</id>
</success>
```

### 8.9.2.2 Example - Refreshing a Document with Context and Prompt Parameters

You must first answer the context before answering the subsequent returned prompt. If no context is provided, the document cannot be refreshed, and the response of the `PUT` call contains the details about the context parameter.
Response:

```xml
<parameters>
  <parameter optional="false" type="context">
    <id>0</id>
    <name>Select a context</name>
    <answer constrained="true" type="text">
      <info cardinality="Single">
        <values>
          <value id="2">Reservations</value> <!-- possible values -->
          <value id="1">Sales</value>
        </values>
      </info>
      <previous>
        <value id="1">Sales</value> <!-- previous value -->
      </previous>
    </answer>
  </parameter>
</parameters>
```

In a second PUT call, an answer is provided to the context.

Request body:

```xml
<parameters>
  <parameter optional="false" type="context">
    <id>0</id>
    <name>Select a context</name>
    <answer type="text">
      <values>
        <value id="1">Sales</value>
      </values>
    </answer>
  </parameter>
</parameters>
```

Once the context is resolved, the response of the PUT call contains the previous value for the second parameter of type prompt.

Response:

```xml
<parameters>
  <parameter optional="false" type="prompt">
    <id>1</id>
    <name>Enter a value for Country:</name>
    <answer constrained="false" type="text">
      <info cardinality="Single">
        <previous>
          <value>France</value> <!-- previous value -->
        </previous>
      </info>
      <values>
        <value>France</value>
      </values>
    </answer>
  </parameter>
</parameters>
```

In another PUT call, the context value and prompt response are given as inputs. Parameters could be in any order, provided that you give all the necessary parameters and their correct identifiers.
8.9.2.3 Example - Refreshing a Document by Using a Query

The request body of the PUT call provides a query that specifies how values should be returned. It does not specify any answer value.

Request body:

```xml
<parameters>
  <parameter type="context">
    <id>0</id>
    <answer type="text">
      <value id="1">Sales</value>
    </answer>
  </parameter>
  <parameter type="prompt">
    <id>1</id>
    <answer type="text">
      <value>France</value>
    </answer>
  </parameter>
</parameters>
```

Response:

```xml
<success>
  <message>The resource of type 'document' with identifier '23535' has been successfully updated.</message>
  <id>23535</id>
</success>
```
8.9.2.4 Example - Refreshing a Document by Using a Query with a Search Pattern

The request body of the PUT call provides a query that specifies which values should be returned according to a pattern. It does not specify any answer value.

Request body (JSON):

```json
{"parameters": {"parameter": {"id": "1", "answer": {"info": {"lov": {"query": {"search": "20*11??"} } } } } }}
```
Response (JSON):

```json
{"parameters":{
  "parameter":{
  ...
  },
  {"@optional":"false",
   "@type":"prompt",
   "@dpId":"DP0",
   "id":1,
   "technicalName":"Enter value(s)",
   "name":"Enter value(s)",
   "answer":{
     "@constrained":"false",
     "@type":"Numeric",
     "info":{
       "@cardinality":"Multiple",
       "lov":{
         "@hierarchical":"false",
         "@partial":"false",
         "@refreshable":"false",
         "@searchable":"true",
         "@mandatorySearch":"true",
         "id":"UNIVERSELoves_DS0.DO3",
         "updated":"2014-12-24T12:51:26.000+01:00",
         "intervals":{
           "interval":{
             "{@id":"0","value":[20121123,20051120]}
           ,
             "{@id":"1","value":[20051112,20071110]}
           ,
             "{@id":"2","value":[20071111,20081130]}
           ,
             "{@id":"3","value":[20111117,20101129]}
           ,
             "{@id":"4","value":[20101113,20091102]}"
          },
         "values":{
           "value":
           [20121123,20121124,20121125,20121126,20121127,20121128,20121129,
            20121130,20121101,20121102,20121103,20121104,20121105,20121106,
            20121107,20121108,20121109,20121110,20121111,20121112,20121113,
            20121114,20121115,20121116,20121117,20121118,20121119,20121120,
            20121121,20121122,20051101,20051102,20051103,20051104,20051105,
            20051106,20051107,20051108,20051109,20051110,20051111,20051112,20051113,20051114,20051115,20051116,20051117,20051118,20051119,20051120]
           },
           "columns":{
             "%@mappingId":"0",
             "@column":{
               "%@id":"0",
               "%@type":"Numeric",
               "%$":"Time Key"
             }
          },
          "previous":{
            "value":20050101
          }},
          "values":{
```
8.9.2.5 Example - Refreshing a Document with a Hierarchical Parameter

A GET .../parameters call returns a list of values for a specific parameter of type prompt that asks you to select a city. Values are countries.

The response contains the prompts at the first level in the hierarchy (hierarchical="true"). final="false" means the value is a node of the hierarchy, while final="true" or no tag means value is a leaf.

Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DPe">
    <id>0</id>
    <technicalName>Select a city:</technicalName>
    <name>Select a city:</name>
    <answer constrained="true" type="Text">
      <info cardinality="Multiple">
        <lov hierarchical="true" partial="false" refreshable="true" searchable="false">
          <id>UNIVERSELOV_DS2.DObb</id>
          <values>
            <value final="false">Australia</value>
            <value final="false">France</value>
            <value final="false">Germany</value>
            <value final="false">Holland</value>
            <value final="false">Japan</value>
            <value final="false">Madagascar</value>
            <value final="false">Middle East</value>
            <value final="false">Nepal</value>
            <value final="false">South Africa</value>
            <value final="false">UK</value>
            <value final="false">US</value>
          </values>
          <columns mappingId="0">
            <column id="0" type="String">Country</column>
          </columns>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

In the first PUT call, a value of the first level of the list of values is given as answer to the parameter. Hierarchical lists of values accept an extra <path> element in the <query> section to go down to each level of the hierarchical parameter. The answer is France.

Request body:

```xml
<parameters>
  <parameter>
    <id>0</id>
    <answer>
      France
    </answer>
  </parameter>
</parameters>
```
The response contains the list of values of the second level of the parameter. Values represent regions. The path attribute of the <lov> element describes the path to any value of the list of values returned in the call. The number represents the data type (0 for string, 1 for date, 2 for numeric).

Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DPe">
    <id>0</id>
    <technicalName>Select a city:</technicalName>
    <name>Select a city:</name>
    <answer constrained="true" type="Text">
      <info cardinality="Multiple">
        <lov hierarchical="true" partial="false" refreshable="true" searchable="false" path="[[0,\ France]]">%
          <id>UNIVERSELOV_DS2.DObb</id>
          <values>
            <value final="false">French Alps</value>
            <value final="false">Normandy</value>
            <value final="false">Paris</value>
            <value final="false">Provence</value>
          </values>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

In a second PUT call, the value of the list of values of the second level is given as answer to the parameter (Provence). The first level value is recalled (France).

Request body:

```xml
<parameters>
  <parameter>
    <id>0</id>
    <answer>
      <info>
        <lov>
          <query>
            <path>
              <value>France</value>
              <value>Provence</value>
            </path>
          </query>
          <path>
            <value>France</value>
          </path>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```
The response contains the list of values of the third and last level of the parameter. Values represent cities.

Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DPe">
    <id>0</id>
    <technicalName>Select a city:</technicalName>
    <name>Select a city:</name>
    <answer constrained="true" type="Text">
      <info cardinality="Multiple">
        <lov hierarchical="true" partial="false" refreshable="true" searchable="false" path="[[0, \ France], [0, \ Provence]]">
          <id>UNIVERSELOV_DS2.DOobb</id>
          <values>
            <value final="false">Bordeaux</value>
            <value final="false">Marseille</value>
            <value final="false">Nice</value>
          </values>
          <columns mappingId="0">
            <column id="0" type="String">City</column>
          </columns>
        </lov>
        <info>
          <answer/>
        </info>
      </answer>
    </answer>
  </parameter>
</parameters>
```

A last PUT call refreshes the document with a `<value>`.

Request body:

```xml
<parameters>
  <parameter>
    <id>0</id>
    <answer>
      <values>
        <value path="[[0, \ France], [0, \ Provence]]">Bordeaux</value>
      </values>
    </answer>
  </parameter>
</parameters>
```

Response:

```xml
<success>
  <message>The resource of type 'Document' with identifier '9586' has been successfully updated.</message>
  <id>9586</id>
</success>
```

### 8.9.2.6 Example - Refreshing a Document with a Hierarchical Parameter of Multiple Columns

A first GET .../parameters call returns a list of values for a specific parameter of type prompt.
The response contains the prompts at the first level in the hierarchy (hierarchical="true"). There is only one column of type String. final="false" means the value is a node of the hierarchy, while final="true" or no tag means value is a leaf.

Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP0">
    <id>0</id>
    <technicalName>Enter one or more Invoice Date values</technicalName>
    <name>Enter one or more Invoice Date values</name>
    <answer constrained="false" type="DateTime">
      <info cardinality="Multiple">
        <lov hierarchical="true" partial="false" refreshable="true" searchable="false">
          <id>UNIVERSELOV_DS0.DO119</id>
          <values>
            <value final="false">FY1992</value>
            <value final="false">FY1993</value>
            <value final="false">FY1994</value>
            <value final="false">FY1995</value>
          </values>
          <columns mappingId="0">
            <column id="0" type="String">Year</column>
          </columns>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

In the first PUT call, a value of the first level of the list of values is given as answer to the parameter. Hierarchical lists of values accept an extra <path> element in the <query> section to go down to each level of the hierarchical parameter. The answer is FY1993.

Request body:

```xml
<parameters>
  <parameter>
    <id>0</id>
    <answer>
      <info>
        <lov>
          <query>
            <path>FY1993</path>
          </query>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

The response contains the list of values of the second level of the parameter. There are two columns of type String. The first one is used as reference (mappingId="0"). The path attribute of the <lov> element describes the path to any value of the list of values returned in the call. The number represents the data type (0 for string, 1 for date, 2 for numeric).

Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP0">
    <id>0</id>
    <technicalName>Enter one or more Invoice Date values</technicalName>
    <name>Enter one or more Invoice Date values</name>
    <answer constrained="false" type="DateTime">
      <info cardinality="Multiple">
        <lov hierarchical="true" partial="false" refreshable="true" searchable="false">
          <id>UNIVERSELOV_DS0.DO119</id>
          <values>
            <value final="false">FY1993</value>
          </values>
          <columns mappingId="0">
            <column id="0" type="String">Year</column>
          </columns>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```
The **PUT** calls are repeated until the last level of the parameter.

Request body:

```xml
<parameters>
  <parameter>
    <id>0</id>
    <answer>
      <info>
        <lov>
          <query>
            <path>
              <value>FY1993</value>
              <value>Q2</value>
              <value>06</value>
              <value type="Numeric">25</value>
            </path>
          </query>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

There is no more `final="false"` attribute in `cvalue` in the response. The last level of the hierarchy is reached, and it contains three columns where the reference is the first one (mappingId="0").
Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP0">
    <id>0</id>
    <technicalName>Enter one or more Invoice Date values</technicalName>
    <name>Enter one or more Invoice Date values</name>
    <answer constrained="false" type="DateTime">
      <info cardinality="Multiple">
        <lov hierarchical="true" partial="false" refreshable="true" searchable="false" path="[[0, FY1993], [0, Q2], [0, 06], [2, 25]]">
          <id>UNIVERSELOV_DS0.DO119</id>
          <cvalues>
            <cvalue>
              <column id="0">1993-06-19T02:00:00.000+02:00</column>
              <column id="1">06</column>
              <column id="2">FY1993</column>
            </cvalue>
          . . .
          <cvalue>
            <column id="0">1995-06-24T02:00:00.000+02:00</column>
            <column id="1">06</column>
            <column id="2">FY1995</column>
          </cvalues>
          <columns mappingId="0">
            <column id="0" type="DateTime">Invoice Date</column>
            <column id="1" type="String">Month</column>
            <column id="2" type="String">Year</column>
          </columns>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>

A last PUT call refreshes the document with a <value>.

Request body:

```xml
<parameters>
  <parameter>
    <id>0</id>
    <answer>
      <values>
        <value path='[[0, FY1993], [0, Q2], [0, 06], [2, 25]]'>1995-06-19T02:00:00.000+02:00</value>
      </values>
    </answer>
  </parameter>
</parameters>

Response:

```xml
<success>
  <message>The resource of type 'Document' with identifier '9586' has been successfully updated.</message>
  <id>9586</id>
</success>
```
8.9.2.7 Example - Refreshing a Document with Hierarchical Indexed Lists of Values

A GET .../parameters call returns a list of values for a specific parameter of type prompt and of which lists of values are indexed ("<value id="...">...<value>"). Only the value ID is mandatory.

The response contains the prompts at the first level in the hierarchy (hierarchical="true"). There is only one column of type String. final="false" means the value is a node of the hierarchy, while final="true" or no tag means value is a leaf.

Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP0">
    <id>0</id>
    <technicalName>customer Hierarchy Node variable mandatory</technicalName>
    <name>customer Hierarchy Node variable mandatory</name>
    <answer constrained="true" type="Text">
      <info cardinality="Single">
        <lov hierarchical="true" partial="false" refreshable="true" searchable="false">
          <id>UNIVERSELOV_DS0.DO48</id>
          <cvalues>
            <cvalue id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[WORLD 0HIER_NODE]" final="false">
              <column id="0">WORLD 0HIER_NODE</column>
              <column id="1">WORLD</column>
            </cvalue>
            <cvalue id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[REST_H 1HIER_REST]" final="false">
              <column id="0">REST_H 1HIER_REST</column>
              <column id="1">Not Assigned Country (s)</column>
            </cvalue>
          </cvalues>
          <columns mappingId="0">
            <column id="0" type="String">LovHierNodeL00 CountryBase </column>
            <column id="1" type="String">LovHierNodeL00 Country </column>
          </columns>
        </lov>
        <previous>
          <value id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[0013]">0013</value>
        </previous>
      </info>
      <values>
        <value id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[0013]">0013</value>
      </values>
    </answer>
  </parameter>
</parameters>
```

In the first PUT call, the ID of the first level of the list of values is given as answer to the parameter. Hierarchical lists of values accept an extra <path> element in the <query> section to go down to each level of the hierarchical parameter.

Request body:

```xml
<parameters>
  <parameter>
    <id>0</id>
    <answer>
```
The response contains the list of values of the second level of the parameter. The path attribute of the lov element describes the path to any value of the list of values returned in the call. The syntax \[0|1|2,\ ,\ \ ID\] describes the value type and ID of a level (0 for string, 1 for date, 2 for numeric).

Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP0">
    <id>0</id>
    <technicalName>customer Hierarchy Node variable mandatory</technicalName>
    <name>customer Hierarchy Node variable mandatory</name>
    <answer constrained="true" type="Text">
      <info cardinality="Single">
        <lov hierarchical="true" partial="false" refreshable="true" searchable="false" path="\[\[0,\ ,\ [Z_COUNTRY COUNTRY_HIERARCHY_01].[WORLD OHIER_NODE]]\]">
          <id>UNIVERSELOV_DS0.DO48</id>
          <cvalues>
            <cvalue id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[EUROPE OHIER_NODE]" final="false">
              <column id="0">EUROPE OHIER_NODE</column>
              <column id="1">EUROPE</column>
            </cvalue>
            <cvalue id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[NORTH_AMERICA OHIER_NODE]" final="false">
              <column id="0">NORTH_AMERICA OHIER_NODE</column>
              <column id="1">NORTH_AMERICA</column>
            </cvalue>
            <cvalue id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[ASIA_PAC OHIER_NODE]" final="false">
              <column id="0">ASIA_PAC OHIER_NODE</column>
              <column id="1">ASIA_PAC</column>
            </cvalue>
            <cvalue id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[MIDDLE_EAST OHIER_NODE]" final="false">
              <column id="0">MIDDLE_EAST OHIER_NODE</column>
              <column id="1">MIDDLE_EAST</column>
            </cvalue>
          </cvalues>
          <columns mappingId="0">
            <column id="0" type="String">LovHierNodeL01 CountryBase</column>
            <column id="1" type="String">LovHierNodeL01 Country</column>
          </columns>
        </lov>
        <previous>
          <value id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[0013]">0013</value>
        </previous>
      </info>
    </answer>
  </parameter>
</parameters>
```
In a second **PUT** call, the value ID of the list of values of the second level is given as answer to the parameter ("[Z_COUNTRY COUNTRY_HIERARCHY_01].[ASIA_PAC_0HIER_NODE]"). The first level value ID is recalled ([Z_COUNTRY COUNTRY_HIERARCHY_01].[WORLD_0HIER_NODE]).

**Request body:**

```xml
<parameters>
  <parameter>
    <id>0</id>
    <answer>
      <info>
        <lov>
          <query>
            <path>
              <value id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[WORLD_0HIER_NODE]" />
              <value id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[ASIA_PAC_0HIER_NODE]" />
            </path>
          </query>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

There is no more `final="false"` attribute in `<cvalue>` in the response. The last level of the hierarchy is reached.

**Response:**

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP0">
    <id>0</id>
    <technicalName>customer Hierarchy Node variable mandatory</technicalName>
    <name>customer Hierarchy Node variable mandatory</name>
    <answer constrained="true" type="Text">
      <info cardinality="Single">
        <lov hierarchical="true" partial="false" refreshable="true" searchable="false" path="[Z_COUNTRY COUNTRY_HIERARCHY_01].[WORLD_0HIER_NODE]"/>
        <cvalues id="UNIVERSELOV_DS0.DO48">
          <cvalue id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[0003]">
            <column id="0">0003</column>
            <column id="1">Australia</column>
          </cvalue>
          <cvalue id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[0006]">
            <column id="0">0006</column>
            <column id="1">Bangladesh</column>
          </cvalue>
          <cvalue id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[0015]">
            <column id="0">0015</column>
            <column id="1">China</column>
          </cvalue>
          ...
          <cvalue id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[0063]">
            <column id="0">0063</column>
            <column id="1">Thailand</column>
          </cvalue>
```

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
A last PUT call refreshes the document with a <value>.

Request body:

```xml
<parameters>
  <parameter>
    <id>0</id>
    <answer>
      <values>
        <value id="[Z_COUNTRY COUNTRY_HIERARCHY_01].[0015]"/>
      </values>
    </answer>
  </parameter>
</parameters>
```

Response:

```xml
<success>
  <message>The resource of type 'Document' with identifier '8816' has been successfully updated.</message>
  <id>8816</id>
</success>
```

### 8.9.2.8 Example - Refreshing a Document with Cascading Parameters

A first GET .../parameters call returns:

- The list of possible values for the answer to a parameter of id=0, type prompt and cardinality Single (age)
- A second parameter of id=1, type prompt, and cardinality Single (customer name)

The first parameter needs to be answered so that you can get the list of values of the second parameter. See the <parameters> element inside the <lov> element of the second parameter.

Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP11">
    <id>0</id>
    <technicalName>Age:</technicalName>
  </parameter>
```
A first PUT .../parameters call answers the first parameter.

Request body:

```xml
<parameters>
  <parameter>
    <id>0</id>
    <answer>
      <values>
        <value>25</value>
      </values>
    </answer>
  </parameter>
</parameters>
```

The response contains the possible values for the second parameter.

Response:

```xml
<parameters>
  <parameter optional="false" type="prompt" dpId="DP11">
    <id>1</id>
    <technicalName>Enter customer:</technicalName>
    <name>Enter customer:</name>
    <answer constrained="false" type="Text">
      <info cardinality="Single">
        <lov hierarchical="false" refreshable="true">
          <id>UNIVERSELOV_DS2.DO10c</id>
          <values>
            ...<value>484</value>
          </values>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```
8.9.3 Cancelling the Refresh of a Document

Usage

Cancels the refresh of a Web Intelligence document that is being refreshed. If no execution is currently running, this has no effect.

Request

PUT /documents/<documentID>/parameters/execution?cancel=<mode>

Where:

- `<mode>` defines how the cancel is managed:
  - partial: when the cancel is performed, this displays the new values retrieved so far in the appropriate parts of the document. The rest of the document will display the values retrieved the last time the query was run.
  - restore: when the cancel is performed, this restores the values to the document that were retrieved the last time the query was run. The values displayed will not be the most up to date information available on the database. You can run the query later to return the up to date values from the database.
  - purge: when the cancel is performed, this displays the document empty of values. The structure and formatting of the document is retained. You can run the query later to return the up to date values from the database.
Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

```xml
PUT /documents/8022/parameters/execution?cancel=partial

<success>
  <message>The resource of type "Document" with identifier "8022" has been successfully updated.</message>
  <id>8022</id>
</success>
```

Related Information

Getting the List of Documents [page 195]

8.9.4 Getting the Details of a Parameter

Usage

Returns the details of a parameter.

Request

```bash
GET /documents/<documentID>/parameters/<parameterID>
```

Response

Response type: application/xml or application/json

The response provides the parameter with its expected answers, previous values if applicable, otherwise default values. See Parameter Response Body Schemas [page 95] to learn about the content structure and element details.
Getting the Details of a Parameter that Depends on Another Parameter

Usage

Returns the details of a parameter of which values depend on the answers of another parameter.

Request

PUT /documents/<documentID>/parameters/<parameterID>

Request body: the details of a parameter on which the parameter you search for depends and a query to return the list of values of the parameter. See Answer Request Body Schemas [page 104] to learn about the content structure and element details.

Response

Response type: application/xml or application/json
The response provides the parameters with their expected answers, previous values if applicable, otherwise default values. See Parameter Response Body Schemas [page 95] to learn about the content structure and element details.

Example

The details of the parameter "1" depend on the answers to the parameter "3".

PUT /documents/6713/parameters/1

Request:

```
<parameters>
  ...
  <parameter>
    <id>3</id>
    <answer>
      <values>
        <value id='2'>Reservations</value>
      </values>
    </answer>
  </parameter>
  <parameter>
    <id>1</id>
    <answer>
      <info>
        <lov>
          <query intervalSize='Unlimited'/>
        </lov>
      </info>
    </answer>
  </parameter>
</parameters>
```

Response:

```
<parameter optional="false" type="prompt" dpId="DP0">
  <id>1</id>
  <technicalName>Enter values for CustomLOV_withcontext:</technicalName>
  <name>Enter values for CustomLOV_withcontext:</name>
  <answer constrained="false" type="Text">
    <info cardinality="Multiple" keepLastValues="true">
      <lov hierarchical="false" partial="false" refreshable="true" searchable="true" mandatorySearch="false">
        <id>UNIVERSELOV_DS0.DOda</id>
        <updated>2015-09-08T10:23:04.000+02:00</updated>
        <cvalues>
          <cvalue>
            <column id="0">Abby</column>
            <column id="1">US</column>
          </cvalue>
          <cvalue>
            <column id="0">Andre</column>
            <column id="1">Nepal</column>
          </cvalue>
          ...
          <cvalue>
            <column id="0">Joos</column>
            <column id="1">Nepal</column>
          </cvalue>
        </cvalues>
      </lov>
    </info>
  </answer>
</parameter>
```
8.10 Refreshing Data Providers

Below are the refresh operations you can do on a data provider of a Web Intelligence document:

- Refreshing the data provider without contexts and prompts
- Identifying contexts and prompts of a data provider
- Fill in contexts and prompts with values

Supported prompts are the following:

- Prompts that accept either string, numeric or date values
- Prompts that accept one value or multiple values
- Optional and non-optional prompts

Getting the Refresh Parameters of a Data Provider [page 489]

Refreshing a Data Provider [page 490]

8.10.1 Getting the Refresh Parameters of a Data Provider

Usage

Returns the parameters to be filled before running a data provider refresh.

Request


Where:
**8.10.2 Refreshing a Data Provider**

**Usage**

Refreshes a Web Intelligence data provider by filling the refresh parameters if needed and running the query.

You can ask for the refresh without providing any parameters (no request body). In this case, the web service returns the context or prompt that needs to be filled. If no parameter has to be filled, the data provider is refreshed.

**Request**

```
PUT /documents/<documentID>/dataproviders/<dataproviderID>/parameters?lovInfo=<lovInfo>
```

Where:

- `<lovInfo>` is an optional, Boolean parameter. Default value is `true`. If set to `false`, the lists of values are not computed, nor displayed.

Request body: the answers to the parameters retrieved using the GET .../parameters call. See Answer XML Grammar [page 104] to learn about the XML content structure and element details.
Response

Response type: application/xml or application/json

When all parameters have been answered, the last PUT call returns a message stating the success of the request.

```
<success>
  <message>The resource of type "Data provider" with identifier "XX" has been successfully updated.</message>
  <id>XX</id>
</success>
```

Example

See Refreshing a Document [page 468].

Related Information

Getting the List of Documents [page 195]
Getting the List of Data Providers [page 405]

8.11 Scheduling Documents

Below are the tasks you can perform to schedule Web Intelligence documents.

When you run a schedule, the document is sent in the specified format to some destination at the given time. For more information on scheduling, see the SAP BusinessObjects Business Intelligence Launch Pad User Guide.

- Getting the List of Schedules [page 491]
- Getting the Details of a Schedule [page 493]
- Adding a Schedule [page 495]
- Deleting a Schedule [page 499]

8.11.1 Getting the List of Schedules

Usage

Gets the list of existing schedules for a Web Intelligence document.
Note

You can find scheduling information in the details of a document. The `<scheduled>` element returned in the response body provides `true` if the document has been scheduled, otherwise `false`.

Request

GET /documents/<documentID>/schedules

Response

Response type: application/xml or application/json

Response body: the list of schedules for the document, with the following information:

- `<id>`
- `<name>`
- `<format>` (Webi, Pdf, Xls, or Csv)
- `<status>` (Pending, Running, Paused, Completed, or Failed)

Example

```
<schedules>
  <schedule>
    <id>28600</id>
    <name>instanceWebi2Inbox</name>
    <format>Webi</format>
    <status>Completed</status>
  </schedule>
  <schedule>
    <id>28609</id>
    <name>instancePDF</name>
    <format>Pdf</format>
    <status>Completed</status>
  </schedule>
  <schedule>
    <id>28651</id>
    <name>instanceWebi2Inbox</name>
    <format>Webi</format>
    <status>Completed</status>
  </schedule>
  <schedule>
    <id>28810</id>
    <name>instanceWebI</name>
    <format>Webi</format>
    <status>Completed</status>
  </schedule>
</schedules>
```
8.11.2 Getting the Details of a Schedule

Usage

Gets the details of a schedule of a Web Intelligence document.

Request

GET /documents/<documentID>/schedules/<scheduleID>

Response

Response type: application/xml or application/json

Response body:

```
<schedule>
  <id>
  <name>
    <format type="webi|pdf|xls|csv">
    <status id="0|1|3|8|9">
    <updated>
    <destination>
    <error>
      recurrence_expression
      <serverGroup id="integer" required="Boolean"/>
      <parameters>
    </schedule>
```


Example

GET /documents/8023/schedules/9439

```
<schedule>
  <id>9439</id>
  <name>now-schedule</name>
  <format type="webi"/>
  <status id="1">Completed</status>
  <updated>2015-09-07T08:51:53.214+02:00</updated>
  <destination>
    <ftp>
      <host>vs0202</host>
```

SAP BusinessObjects RESTful Web Service SDK User Guide for Web Intelligence and the BI Semantic Layer

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
In the case of a failure, the response contains an error message. It can be as follows:

```xml
<schedule>
  <id>9439</id>
  <name>now-schedule</name>
  <format type="webi"/>
  <status id="3">Failed</status>
  <error>
    <error_code>FWB 00031</error_code>
    <message>Destination disabled. [CrystalEnterprise.Ftp]. Please note the name of the job server used for your request and contact your system administrator to make sure the specified destination is enabled. (FWB 00031)</message>
  </error>
  <destination>
    <ftp>...
    </ftp>
  </destination>
  ...
</schedule>
```

Related Information

- Getting the List of Documents [page 195]
- Getting the List of Schedules [page 491]
8.11.3 Adding a Schedule

Usage

Adds a new schedule to a Web Intelligence document.

A schedule can be run now, once, daily, hourly, weekly, or monthly. A schedule can be triggered by a prompt. You can also select a server group that the system uses to run the schedule.

Note

You can find scheduling information in the document details retrieved from the call GET /documents/<documentID>. The <scheduled> element returned in the response body provides true if the document has been scheduled, otherwise false.

Request

POST /documents/<documentID>/schedules

Request body: see Schedules [page 132] for a description of the request body

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of a request.

- Example - File System Destination for a Now Schedule [page 495]
- Example - FTP Destination for a Once Schedule [page 496]
- Example - Mail Destination for a Hourly Schedule [page 496]
- Example - Inbox Destination for a Daily Schedule with a Prompt [page 497]
- Example - Inbox Destination for a Monthly Schedule to Specific Users [page 498]
- Example - Schedule for a Server Group [page 498]

8.11.3.1 Example - File System Destination for a Now Schedule

This example shows how to schedule “now” a Web Intelligence document in PDF format with file system destination. The document is saved in the C:\tmp directory. The default folder is not used.

Example

POST /documents/8002/schedules
8.11.3.2 Example - FTP Destination for a Once Schedule

This example shows how to run a schedule once of a Web Intelligence document in Microsoft Excel format via FTP.

Example

POST /documents/8002/schedules

Request body:

```
<schedule>
  <name>nameOfSchedule</name>
  <format type="xls"/>
  <destination>
    <ftp>
      <host>vs0202.dhcp.pgdev.sap.corp</host>
      <port>21</port>
      <username>admin</username>
      <password>admin</password>
      <directory>\vs0202.dhcp.pgdev.sap.corp\FTP</directory>
    </ftp>
  </destination>
  <once retriesAllowed="2" retryIntervalInSeconds="60">
    <startdate>2012-08-26T15:58:51.000+02:00</startdate>
    <enddate>2013-08-27T15:58:51.000+02:00</enddate>
  </once>
</schedule>
```

8.11.3.3 Example - Mail Destination for a Hourly Schedule

This example shows how to run an hourly schedule of a Web Intelligence document in the default format to an email address (every hour between 26/08/2012 and 14/09/2012). If the schedule fails, there will be 2 retries with 60 seconds between each of them.

Example

POST /documents/8002/schedules

Request body:

```
<schedule>
  <name>nameOfSchedule</name>
  <format type="pdf"/>
  <destination>
    <filesystem>
      <username>{name of user}</username>
      <password>{user's password}</password>
      <directory>C:/tmp</directory>
    </filesystem>
  </destination>
  <once retriesAllowed="2" retryIntervalInSeconds="60">
    <startdate>2012-08-26T15:58:51.000+02:00</startdate>
    <enddate>2013-08-27T15:58:51.000+02:00</enddate>
  </once>
</schedule>
```
8.11.3.4 Example - Inbox Destination for a Daily Schedule with a Prompt

This example shows how to run a daily schedule of a Web Intelligence document in the default format to an inbox. By default, the scheduled document is always sent to the sender.

Since the document contains a prompt, the request body also provides the response to the prompt. Prompt text is named "Enter State:" with value "Texas" on a data provider with "DP0" as identifier.

Example

POST /documents/8002/schedules

Request body:

```xml
Request body:

  <schedule>
    <name>nameOfSchedule</name>
    <format type="webi"/>
    <destination>
      <inbox/>
    </destination>
    <daily retriesAllowed="2" retryIntervalInSeconds="60">
      <startdate>2012-08-26T15:58:51.000+02:00</startdate>
      <enddate>2012-09-14T15:58:51.000+02:00</enddate>
      <dayinterval>1</dayinterval>
    </daily>
    <parameters>
      <parameter optional="false" type="prompt" dpId="DP0">
        <id>0</id>
        <technicalName>Enter State:</technicalName>
        <answer constrained="false" type="text">
          <values>
            <value>Texas</value>
          </values>
        </answer>
      </parameter>
    </parameters>
  </schedule>
```
8.11.3.5 Example - Inbox Destination for a Monthly Schedule to Specific Users

This example shows how to run a monthly schedule of a Web Intelligence document in the default format to an inbox that specifies four user IDs. The scheduled document is sent as a shortcut by specifying the tag sendAs with the value shortcut.

Example

POST /documents/8002/schedules

Request body:

```xml
<schedule>
  <name>nameOfSchedule</name>
  <format type="webi"/>
  <destination>
    <inbox>
      <to>11,12,1,2</to>
      <sendAs type="shortcut"/>
    </inbox>
  </destination>
  <monthly retriesAllowed="2" retryIntervalInSeconds="60">
    <startdate>2012-08-26T15:58:51.000+02:00</startdate>
    <enddate>2013-09-14T15:58:51.000+02:00</enddate>
    <month>1</month>
  </monthly>
</schedule>
```

8.11.3.6 Example - Schedule for a Server Group

This example shows how to run a schedule of a Web Intelligence document on servers in a specific server group.

Example

```xml
<schedule>
  <name>nameOfSchedule</name>
  <format type="webi"/>
  <serverGroup id="6839" required="true"/>
</schedule>
```
8.11.4 Deleting a Schedule

Usage

Deletes a schedule of a Web Intelligence document.

Request

DELETE /documents/<documentID>/schedules/<scheduleID>

Response

Response type: application/xml or application/json

The response is a message stating the success or failure of the request.

Example

```
<success>
  <message>The resource of type "schedule" with identifier "9372" has been successfully removed.
  <id>9372</id>
</success>
```

Related Information

Getting the List of Documents [page 195]
Getting the List of Schedules [page 491]

8.12 Searching for Resources

Below is the task you can perform to get interesting resources located on the CMS repository. In this release, you can search for resources of type folder, universe, connection, Web Intelligence documents, Microsoft Excel files, and CSV files.

Getting Resources [page 500]
8.12.1 Getting Resources

Usage

Returns a list of resources stored on the CMS repository according to a search pattern.

You can search for the following resources:

- Folders
- Universes
- Connections
- Web Intelligence documents
- Microsoft Excel files
- CSV files

The search is targeted to meaningful business intelligence resources, therefore you cannot search for folders only.

Folders, spreadsheets, documents, and CSV files can be searched by identifier. Universes and connections can be searched by resource type and by folder identifier. If no folder identifier is specified, the search is performed on the root folder of the CMS repository.

Request

POST /searches

Request type: application/xml or application/json

Request body:

```xml
<search>
  <folder>
    <folderId>
    <connection>
      <type>
        <folderId>
    <universe>
      <type>
        <folderId>
    <spreadsheet>
      <folderId>
    <document>
      <folderId>
    <csvfile>
      <folderId>
  </folder>
</search>
```

Where:

- `<folderId>` is the folder identifier. Is optional.
- `<type>` is the universe type (`unx` or `unv`) or connection type (`Relational`, `Olap`, `FlattenedOlap`, or `DataFederator`). Is optional.
Response

Response type: application/xml or application/json

Response body: the list of resources requested with the following information:

- Folder ID, CUID, name, and parent folder ID
- Universe ID, CUID, name, type, and folder ID
- Connection type and folder ID
- Document ID, CUID, name, and folder ID
- Microsoft Excel file ID, CUID, name, and folder ID
- CSV file ID, CUID, name, and folder ID

Example

Searching for Universes from the Root Folder (XML)

Request and response types are application/xml.

Request body:

```xml
<search>
    <universe/>
</search>
```

Response:

```xml
<search>
    <universe>
        <id>6912</id>
        <cuid>AViz5oUgKhRLrfeDYOPufRk</cuid>
        <name>Refbeach</name>
        <type>unv</type>
        <folderId>532</folderId>
    </universe>
</search>
```

Example

Searching for Universes and Folders from the Root Folder (XML)

Request and response types are application/xml.

Request body:

```xml
<search>
    <folder/>
    <universe/>
</search>
```

Response:

```xml
<search>
    <folder>
        <id>6011</id>
        <cuid>AVHaHy88GNRHmJCGmL6wBQo</cuid>
        <name>CMSREF_UNIVERSES</name>
        <folderId>532</folderId>
    </folder>
</search>
```
Example

Searching for Folders and Universes in a Specific Folder (JSON)

The search is performed in the folder of ID 7619.

Request and response types are application/json.

Request body:

```json
{"search":
  
  
  "folder":
  
  
  
  
  
  
  
  "universe":
  
  
  "folderId":"7619",
  "type":"unv",
  "folderId":"7619"
}
```

Response:

```json
{"search":
  
  "folder":
  
  
  "universe":
  
  ["id":7777, 
  "cuid":"AbP4G5D9C6BmOL6k0Nz1pK", 
  "name":"ADW.unx", 
  "type":"unv", 
  "folderId":"7619"], 
  ["id":7784, 
  "cuid":"Ad1M10EJrmNCmQP0UUttgog", 
  "name":"Beach For Calculations.unx", 
  "type":"unx", 
  "folderId":"7619"]}
```

Example

Searching for Folders and Microsoft Excel Files in a Specific Folder (XML)

The search is performed in the folder of ID 6335.

Request and response types are application/xml.

Request body:

```xml
<search>
  <folder>
    <folderId>6335</folderId>
  </folder>
</search>
```
8.13 Managing Universes with the Web Intelligence RESTful Web Service SDK

Below are the tasks you can perform on universes with the help of the SAP BusinessObjects Web Intelligence RESTful Web Service SDK.

- Getting the List of Universes (Web Intelligence) [page 503]
- Getting the Details of a Universe (Web Intelligence) [page 505]
- Getting the Query Capabilities of a Universe (Web Intelligence) [page 508]

8.13.1 Getting the List of Universes (Web Intelligence)

Usage

Gets the list of universes a user has access to, depending on user rights.
Request

GET /universes?type=<type>&offset=<offset>&limit=<limit>

Request type: application/xml or application/json

Query parameters:

- `<type>` indicates the type of the universe. Possible values are unv, unx and all. The default value is all. This parameter is optional.
- `<offset>` indicates the position in the list, from which universes are returned. It must be greater than or equal to 0. The default value is 0. This parameter is optional.
- `<limit>` indicates the number of universes that you can list on one page. Its range is [1, 50]. The default value is 10. This parameter is optional.

Response

Response type: application/xml or application/json

Response body: the list of universes for a user with the following information:

- `<id>`
- `<cuid>`
- `<name>`
- `<type>`
- `<folderId>`

Example

GET /universes?type=unx&limit=2

```xml
<universes>
  <universe>
    <id>6773</id>
    <cuid>AXyRzvmPrJxLqUm6_Jbf71E</cuid>
    <name>efashion.unx</name>
    <type>unx</type>
    <folderId>6771</folderId>
  </universe>
  <universe>
    <id>5612</id>
    <cuid>AYCKrid6ngFGvrKlwVfZKj4</cuid>
    <name>Salary.unv</name>
    <type>unv</type>
    <folderId>509</folderId>
  </universe>
  ...
</universes>
```
8.13.2 Getting the Details of a Universe (Web Intelligence)

Usage

Gets the details of a universe referenced by its ID.

Request

GET /universes/<universeID>?aggregated=<aggregated>

Where:

- `<aggregated>` is an optional, Boolean parameter that is supported if the universe is `UNX` only. It is ignored if the universe is `UNV`.

The universe details returned depend on the `<aggregated>` value:

- If true, the call returns the outline containing all folders and objects granted to the user. This outline merges all granted objects from the granted business view and contains all objects properties such as ID, path, and name.
- If false, the call returns the master view if granted or the default view if the master view is denied. The default view name is returned in the outline using the `<businessViewName>` element. This behavior is the one implemented in the SDK versions prior to 4.1 SP5.

If the master view is granted, then the outline returned with the calls .../<universeID>?aggregated=false and .../<universeID>?aggregated=true are the same, except the aggregated outline attribute value.

Response

Response type: application/xml or application/json

Response body: details of the universe, see Universes [page 136].

Example

The master view is granted

GET /universes/9100?aggregated=false

or

GET /universes/9100

```
<universe>
  <id>9100</id>
  <cuid>AftJgs7FPGNBmkdNDnoG8Aw</cuid>
  <name>unx2</name>
  <type>unx</type>
  <folderId>9089</folderId>
  <path>Application Folder/Root Folder/Universes/</path>
```
<connected>true</connected>
<outline aggregated="false">
  <folder>
    <name>City</name>
    <item type="BODimension" dataType="Numeric">
      <name>City Id</name>
      <id>DO1</id>
      <path>City\folder\City Id\dimension</path>
    </item>
    <item type="BODimension" dataType="String">
      <name>City</name>
      <id>DO2</id>
      <path>City\folder\City\dimension</path>
    </item>
    <item type="BODimension" dataType="Numeric">
      <name>Region Id</name>
      <id>DO3</id>
      <path>City|folder\Region Id|dimension</path>
    </item>
  </folder>
  <folder>
    <name>Country</name>
    <item type="BODimension" dataType="Numeric">
      <name>Country Id</name>
      <id>DO4</id>
      <path>Country|folder\Country Id|dimension</path>
    </item>
    <item type="BODimension" dataType="String">
      <name>Country</name>
      <id>DO5</id>
      <path>Country|folder\Country|dimension</path>
    </item>
  </folder>
  <folder>
    <name>Customer</name>
    <item type="BODimension" dataType="Numeric">
      <name>Cust Id</name>
      <id>DO6</id>
      <path>Customer|folder\Cust Id|dimension</path>
    </item>
    <item type="BODimension" dataType="String">
      <name>First Name</name>
      <id>DO7</id>
      <path>Customer|folder\First Name|dimension</path>
    </item>
    ...
    <item type="BODimension" dataType="Numeric">
      <name>City Id</name>
      <id>DOc</id>
      <path>Customer|folder\City Id|dimension</path>
    </item>
    ...
  </folder>
  <folder>
    <name>Reject</name>
    <item type="BODimension" dataType="Numeric">
      <name>Item Id</name>
      <id>DO1e</id>
      <path>Reject|folder\Item Id|dimension</path>
    </item>
    <item type="BODimension" dataType="dateTime">
      <name>Inspection Time</name>
      <id>DO1f</id>
      <path>Reject|folder\Inspection Time|dimension</path>
    </item>
    <item type="BODimension" dataType="String">
      <name>Defect Type</name>
      <id>DO20</id>
    </item>
  </folder>
</outline>
Example

The master view is denied

GET /universes/9100?aggregated=false

or

GET /universes/9100

Response:

<universe>
  <id>9100</id>
  <cuid>AftJgs7FPGNBmkdNDnoG8Aw</cuid>
  <name>unx2</name>
Example

The call requests an aggregated outline

GET /universes/9100?aggregated=true

Response:

<universe>
  <id>9100</id>
  <cuid>AftJgs7FPGNBmkdNDnoG8Aw</cuid>
  <name>unx2</name>
  <type>unx</type>
  <folderId>9089</folderId>
  <path>Application Folder/Root Folder/Universes/</path>
  <connected>true</connected>
  <outline aggregated="true">
    <folder>
    ...
    </outline>
  </universe>

Related Information

Getting the List of Universes (Web Intelligence) [page 503]

8.13.3 Getting the Query Capabilities of a Universe (Web Intelligence)

Usage

Gets the query capabilities of a universe.

Request

GET /universes/<universeID>/capabilities
Response

Response type: text/xml

Response body: the following capabilities of a universe, depending on the user rights:

- General query capabilities
- Data processing capabilities
- Filter capabilities: subqueries supported, result hierarchy in filter, object comparison, constant comparison, query on query, and the following lists:
  - The list of supported comparison operators
  - The list of supported logical operators
  - The list of supported object types
  - The list of query on query supported comparison operators
- Result object capabilities

Example

GET /universes/2234/capabilities

```xml
  <generalCapability combinedQueriesSupported="true" viewQueryScriptAvailable="true" maxValuesForInList="999"/>
  <dataProcessingCapability removeDuplicateRowsAvailable="true"/>
  <filterCapability subQueriesSupported="true" resultHierarchyInFilterSupported="false" objectComparisonSupported="true" constantComparisonSupported="true" queryOnQuerySupported="true">
    <supportedComparisonOperators>equal</supportedComparisonOperators>
    <supportedComparisonOperators>notEqual</supportedComparisonOperators>
    ...
  </supportedLogicalOperators>
  ...
  <supportedObjects>attribute</supportedObjects>
  <supportedObjects>dimension</supportedObjects>
  <supportedObjects>measure</supportedObjects>
  ...
  <queryOnQueryCapability>
    <supportedCorrelationTypesByComparisonOperator>
      <value>Any</value>
    </supportedCorrelationTypesByComparisonOperator>
    <supportedCorrelationTypesByComparisonOperator key="notEqual">
      <value>All</value>
    </supportedCorrelationTypesByComparisonOperator>
    <supportedCorrelationTypesByComparisonOperator key="greater">
      <value>All</value>
    </supportedCorrelationTypesByComparisonOperator>
    ...
  </supportedCorrelationTypesByComparisonOperator>
  ...
  <supportedComparisonOperatorsByCorrelationType>
    <value>inList</value>
  </supportedComparisonOperatorsByCorrelationType>
  <supportedComparisonOperatorsByCorrelationType key="All">
    <value>greater</value>
  </supportedComparisonOperatorsByCorrelationType>
</datasource:QueryCapability>
```
Related Information

Getting the List of Universes (Web Intelligence) [page 503]
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 wilful misconduct. All links are categorized for transparency (see: http://help.sap.com/disclaimer).