SAP BusinessObjects

SAP Crystal Reports RESTful Web Services Developer Guide

- SAP BusinessObjects Business Intelligence platform 4.1

2013-09-20

SAP

Copyright

© 2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice. Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors. National product specifications may vary. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty. SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries. Please see http://www.sap.com/corporate-en/legal/copyright/index.epx#trademark for additional trademark information and notices.

2013-09-20

Contents

Chapter 1 **Start here** **5**

1.1 Introduction to SAP Crystal Reports RESTful web services 5

Chapter 2 **RESTful web services fundamentals** **7**

2.1 Making RESTful web service requests 7
2.1.1 Creating the request header 12
2.1.2 Creating the request body 14
2.1.3 Interpreting the response header 14
2.1.4 Interpreting a response body in XML format 16
2.1.5 Interpreting a response body in the JSON format 20
2.1.6 Comparison of XML and JSON attributes 25
2.1.7 Working with multilingual data 29
2.2 Retrieving the base URL for RESTful web service requests 30
2.2.1 Retrieving the base URL through the CMC 30
2.2.2 Retrieving the base URL programmatically 30
2.3 Authentication 31
2.3.1 To get a logon token from a user name and password 32
2.3.2 To get a logon token from a serialized session or session token 34
2.3.3 To get a logon token using an Active Directory Single Sign-On (AD SSO) account 36
2.3.4 To get a logon token using trusted authentication 37
2.3.5 Converting a logon token from XML-encoded text 38
2.3.6 To add a logon token to a request header 39
2.3.7 Using HTTP basic authentication 40
2.3.8 To log off the BI platform 41
2.3.9 Using authenticated sessions obtained from other SDKs 42
2.4 Retrieving the report ID 44
2.4.1 To retrieve the report ID programatically 44

Chapter 3 **Using the SAP Crystal Reports RESTful web services API** **47**

3.1 Common URIs 47
3.2 Report URI 48
3.3 Report instances 49

3.3.1 To create a transient report instance from an existing report....50
3.4 Exporting....52
3.4.1 Character separated value (CSV) parameters....53
3.4.2 Microsoft Excel parameters....54
3.4.3 Microsoft Excel Data-Only parameters....55
3.4.4 Rich Text Format (RTF) parameters....57
3.4.5 Tab separated text (TTX) paramters....57
3.4.6 XML parameters....58
3.5 Interactive parameters....59
3.6 OData Protocol....61
3.6.1 Accessing the OData service document....61
3.6.2 Grand totals....62
3.6.3 Groups....64
3.6.4 Metadata....68
3.6.5 Rows....69

Appendix A More Information....73

Index 75

Start here

This online help is designed to help you develop your own Web or desktop applications with the SAP Crystal Reports RESTful web services. It includes examples and programming references that you can use to implement the SAP Crystal Reports RESTful web services.

1.1 Introduction to SAP Crystal Reports RESTful web services

SAP Crystal Reports RESTful web services allow SAP Crystal Reports for Enterprise report data managed in a SAP BusinessObjects Business Intelligence Platform repository to be consumed and embedded in mobile devices and web-enabled technology. You can fetch report content in XML or JSON format, and manipulate a report using the RESTful API and OData services. RESTful web services allow you to create applications using the development language of your choice.

SAP Crystal Reports RESTful web services allow you to:

- Export a report to a number of different file types
- Retrieve report metadata.
- Get rows of data and calculations.
- Push a row of data to the report.
- Get and set report parameters.
- Retrieve data in XML or JSON format.

Note:
You can only work with reports created with SAP Crystal Reports for Enterprise when using the RESTful web services.

RESTful web services fundamentals

In this section, you explore the fundamentals of the RESTful web services API. These fundamentals overlap with functionality provided by the SAP Business Intelligence platform RESTful web services API.

For more information see the *Business Intelligence Platform RESTful Web Services Developer Guide*

2.1 Making RESTful web service requests

To access the Business Intelligence platform RESTful web service SDK, you send HTTP requests to the URL that hosts the RESTful web services. The RESTful web service processes the request and returns a response that contains the requested information. You can access RESTful web services with any programming language or tool that supports HTTP requests. RESTful web services follow HTTP standards and the AtomPub specification, but also include custom attributes.

Requests consist of two main components, the request header and the request body. The request header defines the format of the request body, the accepted response format, and other custom settings such as the preferred language and the logon token. The request body may be left blank, or it may contain additional information needed to complete the request. For example, an authentication request passes the user name and password as formatted XML in the request body.

To make a RESTful web service request, you need the following:

- URL - The URL that hosts the RESTful web service.
- Method - The type of HTTP method to use for sending the request, for example `GET`, `PUT`, `POST`, or `DELETE`.
- Request header - The attributes that describe the request.
- Request body - Additional information that is used to process the request.

Once the request has been processed, you will receive a response. Responses contain the requested information, and include supporting information that you need to complete your next step. For example, responses may contain XML templates that can be used to populate the request body of subsequent requests, or they may contain links to related RESTful URLs, including parent folders, child folders, pages of additional information, and related links. By following the information provided by a RESTful response, you can navigate the requested data and obtain the templates you need in order to complete subsequent requests.

The Business Intelligence platform RESTful web service responses may be formatted as XML or JSON depending on the capabilities of the BI platform client application.

RESTful web service responses contain two main components:

- Response header - A list of attributes that describes the response format, and includes an HTTP response code.
- Response body - The requested information, and additional information that enables you to complete subsequent requests.

The examples in this document define the URL, method, request header attributes, and request body content that is required for each RESTful request. You can access the RESTful web services using any programming language or tool that supports HTTP requests.

Example: A RESTful POST request using the /logon/long API and response using the XML format

This example shows a RESTful request that logs on to the BI platform repository.

Request

URL: `http://localhost:6405/biprws/logon/long`

Method: `POST`

Request header attributes:

Attribute	Value
Content-Type	application/xml
Accept	application/xml

Request body:

```
<attrs xmlns="http://www.sap.com/rws/bip">
  <attr name="userName" type="string">username</attr>
  <attr name="password" type="string">password</attr>
  <attr name="auth" type="string" possibilities="secEnterprise,secLDAP,secWinAD">secEnterprise</attr>
</attrs>
```

Response

Response header:

Attribute	Value
Status code	200 OK
Server	Apache-Coyote/1.1
X-SAP-LogonToken	"COMMANDCOM-LCM:6400@{3&2=5542,U3&p=40680.8979564815,Y7&4F=12,U3&63=secEnterprise,0P&66=60,03&68=secEnterprise:Administrator,0P&qe=100,U3&vz=TZnoIE2yQyeLCkAlnHtaaYUHon5.p0yTkSaUiLC8SSM,UP}"
Date	Tue, 17 May 2011 21:33:03 GMT
Content-Type	application/xml
Content-Length	586

Response body:

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <author><name>@COMMANDCOM-LCM:6400</name></author>
  <id>tag:sap.com,2010:bip-rs/logon/long</id>
  <title type="text">Logon Result</title>
  <updated>2011-05-17T21:33:03.471Z</updated>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="logonToken" type="string">COMMANDCOM-LCM:6400@{3&amp;2=5542,U3&am....YUHon5.p0yTk
SaUiLC8SSM,UP}</attr>
    </attrs>
  </content>
</entry>
```

Example: A RESTful GET and POST request using the /logon/long API and response using the JSON format

This example shows a RESTful request that uses a `GET` request to retrieves a `JSON` formatted request body to use to enter the name and password and authentication type, then using a `POST` request to retrieve a logon token from the BI platform repository.

Request

URL: `http://localhost:6405/biprws/logon/long`

Method: `GET`

Request header attributes:

Attribute	Value

Attribute	Value
Accept	application/json

The request body that is returned in `JSON` format after a `GET` request appears as follows:

```
{"userName":"","password":"","auth":"secEnterprise"}
```

Request body that has an name label, for example `BOEuser` and password, for example `BOEPassword999` included before sending it as a `POST` request as showed in the following code snippet:

```
{"userName":"BOEuser","password":"BOEPassword999","auth":"secEnterprise"}
```

Note:
The `auth` default value is `secEnterprise`. The authentication types that may be used include are as follows:

- `secEnterprise` - Enterprise authentication
- `secLDAP` - Lightweight Directory Access Protocol authentication
- `secWinAD` - Windows Active Directory authentication
- `secSAPR3` - SAP authentication

Response header after a `POST` request:

Attribute	Value
Status code	200 OK
Server	Apache-Coyote/1.1
X-SAP-LogonToken	:"COMMANDCOM-LCM:6400@{3&2=5571,U3&p=40897.0049317824,Y7&4F=12,U3&63=secEnterprise,0P&66=60,03&68=secEnterprise:BOEuser,0P&qe=100,U3&vz=odiw9uLc1kVlJf9lggLFEWPAX3qsFWBT1LkdE2DTGhY,UP}"
Date	Tue, 17 December 2011 21:33:03 GMT
Content-Type	application/json
Content-Length	204

Response body in JSON format:

```
{"logonToken":"COMMANDCOM-LCM:6400@{3&2=5571,U3&p=40897.0049317824,Y7&4F=12,U3&63=secEnterprise,
0P&66=60,03&68=secEnterprise:Administrator,0P&qe=100,U3&vz=odiw9uLc1kVlJf
9lggLFEWPAX3qsFWBT1LkdE2DTGhY,UP}"}
```

Example: A RESTful infostore JSON-formatted request

This example shows a RESTful request that uses a `GET` request and the `/infostore` API with a logon token to request information from BI platform repository that is returned in `JSON` format.

Request

URL: `http://commandcom-lcm:6405/biprws/infostore`

Method: `GET`

Request header attributes:

Attribute	Value
Accept	application/json
X-SAP-LogonToken	COMMANDCOM-LCM:6400@{3&2=5542,U3&p=40680.897...UiLC8SS

Request body: (blank)

Response

Response header:

Attribute	Value
Status code	200 OK
Server	Apache-Coyote/1.1
X-SAP-LogonToken	COMMANDCOM-LCM:6400@{3&2=5542,U3&p=40680.897...UiLC8SSM,UP}
Date	Tue, 17 December 2011 21:33:03 GMT
Content-Type	application/json
Content-Length	6919

Response body formatted as JSON. For clarity in the following code snippet, the back slash for escaped characters such as (/) and (") have been removed.

```
{
   "__metadata":
     {"uri":"http://localhost:9998/biprws/infostore/Root%20Folder/children?page=1&amp;pageSize=3"},
   "first":
     {"__deferred":
     {"uri":"http://localhost:9998/biprws/infostore/Root%20Folder/children?page=1&amp;pageSize=3"}
     },
   "next":
     {"__deferred":
     {"uri":"http://localhost:9998/biprws/infostore/Root%20Folder/children?page=2&amp;pageSize=3"}
     },
   "last":
     {"__deferred":
```

```
    {"uri":"http://localhost:9998/biprws/infostore/Root%20Folder/children?page=3&amp;pageSize=3"}
    },
  "entries":
  [
    {"__metadata":
      {"uri":"http://localhost:9998/biprws/infostore/4005"},
      "id":4005,
      "cuid":"FnKsrkkctAcA8BAAALB7kkQAADAFzVMX",
      "name":"Data Federation",
      "type":"Folder"
    },
    {"__metadata":
      {"uri":"http://localhost:9998/biprws/infostore/3931"},
      "id":3931,
      "cuid":"AclaKZlzj5VJmMQi5LDa53s",
      "name":"LCM",
      "type":"Folder"
    },
    {"__metadata":
      {"uri":"http://localhost:9998/biprws/infostore/5056"},
      "id":5056,
      "cuid":"Acu9FvxWBZ9Htt0_08a25b4",
      "description":"",
      "name":"Monitoring Report Sample",
      "type":"Folder"
     }
  ]
}
```


Related Topics

- Retrieving the base URL for RESTful web service requests

2.1.1 Creating the request header

The request header of an HTTP request contains a set of attributes that describe the request. The RESTful web services SDK recognizes a set of standard HTTP attributes as well as custom attributes.

The following table describes request headers that are recognized by the RESTful web service:

Attribute	Description
`Accept`	The accepted content types. Specifies the content in the request body. Use this value to specify the format. Examples of acceptable values: • `application/xml` • `application/atom+xml` • `application/json`
`Accept-Charset`	The accepted character sets.
`Accept-Encoding`	The accepted character encodings.
`Accept-Language`	The preferred language used to retrieve system and error messages. This corresponds to the Procuct Locale (PL) of the BI Platform.
`Content-Length`	The number of characters in the request body.
`Content-Type`	The type of content in the request body. Examples of acceptable values: • `application/xml` • `application/atom+xml` • `application/atom+xml;type=feed`

Attribute	Description
`Host`	The internet host and port number of the RESTful web service.
`X-SAP-LogonToken`	The value of the logon token recieved from the authentication process. **Note:** The logon token must be surrounded in quotation marks when added to the request header. For example: `X-SAP-LogonToken: "COMMANDCOM-LCM:6400@{3&2=5328,U3&p=40676.8926203819,Y7&4F=12,U3&63=secEnterprise,0P&66=60,03&68=secEnterprise:Administrator,0P&qe=100,U3&vz=IVD21LbMCB0eRiI4atz9sNL18Ux5anRBdYB9fFv5NrY,UP}"`
`X-SAP-PVL`	The preferred language used to retrieve BI platform content. This corresponds to the Preferred Viewing Language (PVL)

Related Topics

- Authentication

2.1.2 Creating the request body

The request body contains the information that RESTful web services needs to complete the request. For example, the request body of an authentication request contains the logon information, including user name and password. This provides the authentication URL with the information it needs to accept or reject the logon request.

You set an attribute in the request header to define the format of the request body. Set the `Content-Type` attribute in the message header to specify the format.

2.1.3 Interpreting the response header

The response header contains attributes that describe whether the request was successful, and describe the contents of the response body. Most of the response header attributes belong to the HTTP standard. However, the `X-SAP-LogonToken` header attribute is a custom attribute used only by the BI platform.

Status code

The status code contains a standard HTTP status code that describes whether the request was successful.

HTTP Response Code	Error	Description
400	Bad request	The requested resource exists, but the request contains errors.
401	Failed to logon or invalid session	Logon failed. Check that the username, password, and servername are correct.
403	Access denied	You do not have permission to operate on the requested resource. The current session may have expired. Log on to obtain a new session.
404	Service is not available	The requested service is not provided by the RESTful web services SDK.
405	Invalid request method	A request was made using a method that was not supported by the resource. For example, using a PUT request on a read-only resource.
406	Not acceptable	The requested resource cannot generate the content type specified by the `Accept` attribute of the request header.
408	BI platform server timeout	The server timed out waiting for the request.
415	Unsupported media type	The request contains a media type that the server or resource does not support.
500	RESTful web service internal error	An unclassified error occurred. See the response body for more information.
503	RESTful web service plugin not found	RESTful web services are not available. Verify that RESTful web services are configured correctly.

Server

The server that was used to process the request.

Date

The date and time of the response.

Content-Type

The format of the response body. For example, most web service responses use the value `application/xml` to show that the response body is formatted as XML.

Content-Length

The length of the response body.

Transfer-Encoding

The type of encoding that has been used to transport the message.

Content-Location

An alternative link that can be used to find the resource.

X-SAP-LogonToken

A token that can be used with subsequent requests to prove that you have been authenticated to access the BI platform. Authentication requests return the `X-SAP-LogonToken` custom attribute in the response header. Include the logon token in the request header of subsequent requests, and enclose it in quotation marks.

Note:
A copy of the `X-SAP-LogonToken` value is returned in the response body of authentication responses. However, the response body is formatted as XML and converts the logon token to an XML-encoded version. This copy of the logon token must be converted back to its original format before it can be used.

Related Topics

- Converting a logon token from XML-encoded text

2.1.4 Interpreting a response body in XML format

The Business Intelligence platform RESTful web service SDK provides responses in XML format, according to the Atom specification, available at *http://www.w3.org*. This section describes how XML tags apply to RESTful web services. The following screen illustrates how the BI launchpad returns XML data in response to a typical /infostore request.

```
<?xml version="1.0" ?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>tag:sap.com,2010:bip-rs/infostore</id>
  <title type="text">InfoStore (@COMMANDCOM-LCM:6400)</title>
  <updated>2012-02-02T22:35:11.975Z</updated>
  <link href="http://localhost:6405/biprws/infostore/4/children?page=1&pageSize=5" rel="self"/>
  <link href="http://localhost:6405/biprws/infostore/4/children?page=1&pageSize=5" rel="first"/>
  <link href="http://localhost:6405/biprws/infostore/4/children?page=2&pageSize=5" rel="next"/>
  <link href="http://localhost:6405/biprws/infostore/4/children?page=7&pageSize=5" rel="last"/>
  <entry>
    <title type="text">Alert Notifications</title>
    <id>tag:sap.com,2010:bip-rs/ARZB.BFCQk9PqaqDpcFwo1w</id>
    <author>
      <name>System Account</name>
    </author>
    <link href="http://localhost:6405/biprws/infostore/Alert%20Notifications" rel="alternate"/>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id"          type="int32">64</attr>
        <attr name="cuid"        type="string">ARZB.BFCQk9PqaqDpcFwo1w</attr>
        <attr name="description" type="string">Description here</attr>
        <attr name="name"        type="string">Alert Notifications</attr>
        <attr name="type"        type="string">Folder</attr>
      </attrs>
    </content>
  </entry>
  <entry>
    <title type="text">Application Folder</title>
      :
      :
    </content>
  </entry>
</feed>
```

<feed>

The `<feed>` element defines a list of `<entry>` elements. JSON uses curly brackets { and } to enclose a response.

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <entry> ... </entry>
  <entry> ... </entry>
  ...
</feed>
```

<entry>

A single item. The `<entry>` tag may include the `xmlns` attribute.

```
<entry xmlns="http://www.w3.org/2005/Atom">
  ...
</entry>
```

<author>

The owner of the resource that was accessed. The `<author>` element includes a `<name>` element that defines the name of the owner of the resource. The following element shows that the owner of the resource is `System Account`.

```
<author><name>System Account</name></author>
```

<id>

A unique identifier of the resource.

```
<id>tag:sap.com,2010:bip-rs/AdoctK9h1sBHp3I6uG0Sh7M</id>
```

<title>

The name of the resource. This example shows that the name of the resource is `Application Folder`.

```
<title type="text">Application Folder</title>
```

<updated>

The date and time the resource was last updated.

```
<updated>2011-04-14T10:27:50.672Z</updated>
```

<link>

The `link` element defines links to URLs that can be used with other RESTful web service requests. These may include parent or child folders, or other information that is relevant to the request. By following these links, you can navigate through the BI platform repository.

The `href` attribute of the link tag defines the hyperlink, and the `rel` attribute describes the type of link. The following list describes possible values of the `rel` attribute:

<link> Related Attribute Name	Description
`self`	A link back to this URL.
`first`	A link to the first page of results.
`next`	A link to the next page of results.
`previous`	A link to the previous page of results.
`last`	A link to the last page of results.
`alternate`	Another link to the same resource.
`up`	A link to the parent of the current resource.
`related`	A link to a related resource.
`http://www.sap.com/rws/bip#children`	A link to the children of the current resource.
`http://www.sap.com/rws/bip#opendocument`	A link that can be used to view the resource with OpenDocument.
`http://www.sap.com/rws/bip#schedule`	A link that can be used to schedule a resource.

For example, the following link element describes a link to the next page of results:

```
<link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=3&pageSize=3" rel="next"></link>
```

Responses that provide links to documents also provide an OpenDocument URL that can be used to view documents using OpenDocument.

```
<link href="http://localhost:8080/BOE/OpenDocument/opendoc/openDocument.jsp?sIDType=CUID&iDocID=Aa0U0jQbtKx
Cn.D3JDLOaHs" rel="http://www.sap.com/rws/bip#opendocument" title="OpenDocument">
```

For more information about OpenDocument, see *Viewing Documents Using OpenDocument*.

Note:
You can use logon tokens obtained from this SDK to authenticate with OpenDocument.

<content>

The payload of the RESTful response. The `<content>` element contains an `<attrs>` element, which itself contains a set of `<attr>` elements.

```
<content>
  <attrs>
    <attr>...</attr>
    <attr>...</attr>
  </attrs>
</content>
```

<attrs>

A list of properties of the content. The `<attrs>` element contains a set of `<attr>` elements.

```
<attrs>
  <attr>...</attr>
  <attr>...</attr>
</attrs>
```

<attr>

A property of the content.

Each `<attr>` element defines a property of the content. The `<attr>` tag uses two attributes, `name`, which describes the name of the property, and `type`, which describes the type of the property. The following example shows that the `id` property of the content is the value `43` (an integer), and the `name` property of the content is `Application Folder` (a string).

```
<attr name="id" type="int32">43</attr>
<attr name="name" type="string">Application Folder</attr>
```

This table describes the possible values for the `name` and `type` attributes of the `<attr>` tag.

Name	Type	Description
name	string	The name of the resource.
id	int32	The ID of the resource.
cuid	string	A unique identifier of the resource.
type	string	The type of resource, for example Folder or InfoView.
description	string	A description of the resource.
logonToken	string	A logon token.

<error>

Error codes.

Each <error_code> and <message> element refers to a RESTful Web Services error code reference in teh format RWS 000xx and includes a brief description. For more details, see the BusinessObjects XI *Error Messages Explained* guide.

2.1.5 Interpreting a response body in the JSON format

The Business Intelligence platform RESTful web service SDK provides responses in JSON format with the request header `accept : application/json`. This section describes how JSON tags apply to RESTful web services.

{ ... }

A JSON object is enclosed by curly brackets { and }, which is similar to the XML `<feed>` element.

```
{

    "__metadata": {
        "uri": "http://commandcom-lcm:6405/biprws/infostore/4/children?page=1&pageSize=50"
    },
    "first": {
        "__deferred": {
            "uri": "http://commandcom-lcm:6405/biprws/infostore/4/children?page=1&pageSize=50"
        }
    },
    "last": {
        "__deferred": {
            "uri": "http://commandcom-lcm:6405/biprws/infostore/4/children?page=1&pageSize=50"
        }
    },
    "entries":
    [
        {
            "__metadata": {
                "uri": "commandcom-lcm:6405/biprws/infostore/Alert%20Notifications"
            },
            "id": 64,
            "cuid": "ARZB.BFCQk9PqaqDpcFwo1w",
            "name": "Alert Notifications",
            "type": "Folder",
            "uri":"alslsls"
        }
     .
     .
     .
    ]

}
```

"entries":

Entries are JSON objects within an array. The format is `"entries" : [{contentsOfEntryItem#1}, {contentsOfEntryItem#2}]`. The following example is a result of an `../infostore` RESTful Web Service API request. The `"entries":` part of the response shows two children named `"Alert Notifications"` and `"Users"`.

```
"entries":
    [
        {
            "__metadata": {
                "uri": "commandcom-lcm:6405/biprws/infostore/Alert%20Notifications"
            },
            "id": 64,
            "cuid": "ARZB.BFCQk9PqaqDpcFwo1w",
```

```
            "name": "Alert Notifications",
            "type": "Folder",
            "uri":"alslsls"
        },
    .
    .
    .
        {
            "__metadata": {
                "uri": "http://commandcom-lcm:6405/biprws/infostore/Users"
            },
            "id": 19,
            "cuid": "AXhmigik4CBKra9ZYzR2ezE",
            "description": "",
            "name": "Users",
            "type": "Folder"
        }
    ]
```

__metadata: { uri:

The `__metadata: { uri:` element equates to the XML <link> element. This defines links to URLs that can be used with other RESTful web service requests. These may include parent or child folders, or other information that is relevant to the request. By following these links, you can navigate through the BI platform repository.

The `href` attribute of the link tag defines the hyperlink, and the `rel` attribute describes the type of link. The following list describes possible values of the `rel` attribute. Note that the XML tags `alternate` and `related` have no JSON equivalent.

Attribute	Format	Example	Description
self	__metadata: { uri:	"__metadata":{"uri":"http://localhost:6405/biprws/infostore/4/children?page=1&pageSize=5"}	A link back to this URL.
first	first: { __deferred: { uri:	"first":{"__deferred":{"uri":"http://localhost:6405/biprws/infostore/4/children?page=1&pageSize=5"}	A link to the first page of results.
next	next: { __deferred: { uri:	"next":{"__deferred":{"uri":"http://localhost:6405/biprws/infostore/4/children?page=2&pageSize=5"}	A link to the next page of results.
previous	previous: { __deferred: { uri:	"previous":{"__deferred":{"uri":"http://localhost:6405/biprws/infostore/4/children?page=6&pageSize=5"}	A link to the previous page of results.

Attribute	Format	Example	Description
last	last: { __deferred: { uri:	"last":{"__deferred":{"uri":"http://localhost:6405/biprws/infostore/4/children?page=7&pageSize=5"}	A link to the last page of results.
up	up: { __deferred: { uri:	"up":{"__deferred":{"uri":"http://localhost:6405/biprws/infostore"}	A link to the parent of the current resource.
children	children: { __deferred: { uri:	"Children":{"__deferred":{"uri":"http://localhost:6405/biprws/infostore/User%20Folders/children"}	A link to the children of the current resource.
openDocument	opendocument { __deferred: { uri:	"openDocument":{"__deferred":{"uri":"http://commandcom-lcm:8080/BOE/OpenDocument/opendoc/openDocument.jsp?sIDType=CUID&iDocID=AQtkbbSqN4NOj3ydf.Sw1lY"}	A link that can be used to view the resource with OpenDocument.
schedule	schedule { __deferred: { uri:	"Scheduling forms":{"__deferred":{"uri":"http://localhost:6405/biprws/infostore/4930/scheduleForms"} Use `Post` and include the schedule: "__metadata":{"uri":"http://localhost:6405/biprws/infostore/4930/scheduleForms/hourly"}	A link that can be used to schedule a resource. Use Get to retrieve the template, use Post to send the request.

For example, the following link element describes a link to the last page of results:

```
"last": {
        "__deferred": {
            "uri": "http://commandcom-lcm:6405/biprws/infostore/4/children?page=1&pageSize=50"
        }
```

Responses that include document types, such as Web Intelligence and Crystal Reports, also provide an openDocument URL that can then be emailed or attached to a button control on a report.

In the following example, the `../infostore` API is used to retrieve the listing of a Web Intelligence openDocument-formatted links.

```
http://commandcom-lcm:6405/biprws/infostore/4930
```


```
{
   "up":{
      "__deferred":{
         "uri":"http://10.162.204.68:6405/biprws/infostore/4904"
      }
   },
   "Scheduling forms":{
      "__deferred":{
         "uri":"http://10.162.204.68:6405/biprws/infostore/4907/scheduleForms"
      }
   },
   "id":4907,
   "cuid":"AQtkbbSqN4NOj3ydf.Sw1lY",
   "openDocument":{
      "__deferred":{
         "uri":"http://commandcom-lcm:8080/BOE/OpenDocument/opendoc/openDocument.jsp?
                sIDType=CUID&iDocID=AQtkbbSqN4NOj3ydf.Sw1lY"
      }
   },
   "description":"",
   "name":"Formatting Sample",
   "type":"Webi"
}
```


For more information about OpenDocument, see *Viewing Documents Using OpenDocument*.

Note:
You can use the `../logon/long` API to obtain a logon token string that can be added to an openDocument URL so recipients do not have to provide their logon credentials.

Entry properties

Several properties make up the content of each entry item. The following example shows that the `id` property of the content is the value `64` (an integer), and the `name` property of the content is `Alert Notifications` (a string).


```
{
    "__metadata": {"uri": "commandcom-lcm:6405/biprws/infostore/Alert%20Notifications"},
    "id": 64,
    "cuid": "ARZB.BFCQk9PqaqDpcFwo1w",
    "name": "Alert Notifications",
    "type": "Folder",
    "uri":"alslsls"
},
```


This table describes the available `name` and `type` properties for a JSON entry.

Name	Type	Example	Description
name	string	"name": "Alert Notifications"	The name of the resource.
id	int32	"id": 64	The ID number of the resource.
cuid	string	"cuid": "ARZB.BFCQk9PqaqDpcFwo1w"	A unique identifier of the resource.
type	string	"type": "Folder"	The type of resource, for example Folder or InfoView.

Name	Type	Example	Description
description	string	"description": "Contains the ..."	A description of the resource.
logonToken	string	"type": "COMMANDCOM-LCM:6400@{3&...Sv3b6vUJZe9...}"	A logon token.
uri	string	"uri": "http://localhost:6405/biprws/infostore/Custom%20Roles"	URI value.
openDocument	string	"openDocument":{"__deferred":{"uri":"http://commandcom-lcm:8080/BOE/OpenDocument/opendoc/openDocument.jsp?sIDType=CUID&iDocID=AQtkbbSqN4NOj3ydf.Sw1lY"}	An openDocument formatted URI value.

error_code

Each `error_code` and `message` element refers to a BI platform error or a RESTful Web Services error (RWS prefix) and includes a brief description. For more information, see the SAP BusinessObjects XI *Error Messages Explained* guide.

```
{
   "error_code":"FWM 01003",
   "message":"Server COMMANDC-OM-LCM:6400 not found or server may be down (FWM 01003)"
}
```

JSON escape characters

RESTful Web Services returns ASCII characters that are considered special by JSON by prefacing them with a back slash (\ ). The JSON specification for which characters must be escaped can be found at `http://www.ietf.org/rfc/rfc4627.txt` The following table lists several common ASCII++ characters that RESTful Web Service JSON requests will return prefaced with backslashes:

RWS - JSON	Unicode UTF-8	Description
\b	U+0008	Backspace
\f	U+000C	Form feed
\n	U+000A	New line
\r	U+000D	Carriage return
\t	U+0009	Tab
\v	U+000B	Vertical tab
\'	U+0027	Single quote
\"	U+0022	Double quote

RWS - JSON	Unicode UTF-8	Description
\\	U+005C	Back slash or reverse solidus
\/	U+005D	Forward slash or solidus
\u	U+xxxx	four-hex-digits

2.1.6 Comparison of XML and JSON attributes

RESTful Web Services requests that use XML always return some data to comply with the Atom specification. The following XML tags that do not have equivalents in the JSON data format, and it helps to be aware of them:

- <author>
- <id>
- <title>
- <updated>
- <link rel=alternate>
- <link rel=related>
- <content>
- <attrs>

Supported XML tags and JSON objects

The following table lists the XML tags and their equivalent JSON objects and entries supported by the BI platform RESTful Web Services implementation.

Table 2-1: Supported XML tags and JSON objects

XML			JSON		Description
XML Tag	Sample	Type	Value	Type	
<feed>			{	JSON object	In a JSON result, the response is represented as a JSON Object. The XML <feed> tag equates to JSON's outermost curly brackets { }.

XML			JSON		Description
XML Tag	Sample	Type	Value	Type	
<entry>			entries : [{contentsOfEntryItem#1}, {contentsOfEntryItem#2}]		A request for a list of children, a collection of entries is returned, each one a JSON object. The collection of JSON objects is represented as an array in the "entries" name and value pair.
<author>			No JSON equivalent		These elements are not exposed in JSON.
<id>					
<title>					
<updated>					

XML			JSON		Description
XML Tag	Sample	Type	Value	Type	
<link>	rel=self		__metadata: { uri:		A link to your current location.
	rel=first		first: { __deferred: { uri::		A link to the first page of results.
	rel=next		next: { __deferred: { uri::		A link to the next page of results.
	rel=previous		previous: { __deferred: { uri::		A link to the previous page of results .
	rel=last		last: { __deferred: { uri::		A link to the last page of results .
	rel=alternate		No JSON equivalent.		An alternate link to your current location.
	rel=up		up: { __deferred: { uri::		A link to the parent of the current resource.
	rel=related		No JSON equivalent.		A link to a related resource.
	rel=http://www.sap.com/rws/bip#children		children: { __deferred: { uri::		A link to the children of the current resource.
	rel=http://www.sap.com/rws/bip#opendocument		opendocument: { __deferred: { uri::		A link that can be used to open a document such as a report or Adobe Acrobat PDF file.
	rel=http://www.sap.com/rws/bip#schedule		schedule: { __deferred: { uri::		A link that can be used to schedule a resource.
<content>			No JSON equivalent.		For XML only, this is a container for the `<attrs>` element. `<content>` is required for the Atom feed specification, but not for JSON.

XML			JSON		Description
XML Tag	Sample	Type	Value	Type	
<attrs>					The XML element that contains one or more `<attr>` elements. In JSON, the attributes are presented as name and value pairs immediately within the JSON object representing the resource, rather than grouped as with the XML `<attrs>` tag.
<attr>	name=name	string	name:	JSON string	The name of the resource.
	name=id	int32	id:	JSON number	The numerical identification number of the resource.
	name=cuid	string	cuid:	JSON string	The 23 character alphanumeric cluster unique identifier.
	name=type	string	type:	JSON string	The type of resource, for example Folder or InfoView.
	name=description	string	description:	JSON string	The description of the resource.
	name=logonToken	string	logonToken:	JSON string	The logon token string.

Example: A comparison of XML and JSON format from an /infostore request

The following code snippet shows the hierarchy of RESTful Web Service elements with a typical /infostore `GET` request. On the left, is the XML listing. On the right, is the JSON listing of the same request. The corresponding lines of information are arranged for easier side-by-side comparison. To reduce the length of the code snippet, only the first object called "Alert Notifications" is shown. Note that this screenshot does not contain all available tags listed in the preceeding table.

XML	JSON
<?xml version="1.0" ?>	
<feed xmlns="http://www.w3.org/2005/Atom">	{
<id>tag:sap.com,2010:bip-rs/infostore</id>	
<title type="text">InfoStore (@COM...CM:6400)</title>	
<updated>2012-01-13T20:47:42.942Z</updated>	
<link href="http://...?page=1&pageSize=5" rel="self"/>	"__metadata": {"uri": "http://...?page=1&pageSize=5"},
<link href="http://...?page=1&pageSize=5" rel="first"/>	"first": {"__deferred": {"uri": "http://...?page=1&pageSize=5"}},
<link href="http://...?page=6&pageSize=5" rel="previous"/>	"previous": {"__deferred": {"uri": "http://...?page=6&pageSize=5"}},
<link href="http://...?page=7&pageSize=5" rel="last"/>	"last": {"__deferred": {"uri": "http://...?page=7&pageSize=5"}},
	"entries":
	[
<entry>	{
<title type="text">Alert Notifications</title>	"name": "Alert Notifications",
<id>tag:sap.com,2010:bip-rsARZB.BF...aqDpcFwo1w</id>	
<author><name>System Account</name></author>	
<link href="...infostore/Alert%20Notifications" rel="alternate"/>	{"__metadata": {"uri": "http://...infostore/Alert%20Notifications"},
<content type="application/xml">	
<attrs xmlns="http://www.sap.com/rws/bip">	
<attr name="id" type="int32">64</attr>	"id": 64,
<attr name="cuid" type="string">ARZB.BF...pcFwo1w</attr>	"cuid": "ARZB.BF...aqDpcFwo1w",
<attr name="description" null="true" type="string"/>	
<attr name="type" type="string">Folder</attr>	"type": "Folder"
</attrs>	
</content>	}
</entry>	]
</feed>	}

2.1.7 Working with multilingual data

In multilingual environments, you can request the content and system messages to be returned in your preferred language. There are two request header attributes used to define the preferred language for content and system messages: `Accept-Language` and `X-SAP-PVL`.

When the BI platform software is installed, the user interface and system error messages are displayed in the Product Locale (PL). The available PL languages include the language packs that are installed with the BI platform software.

The system messages, including error messages, are returned in the language specified by the PL. You can request to use a specific language for system messages by setting the `Accept-Language` request header attribute. For example, to retrieve system messages in Japanese, set the `Accept-Language` request header attribute to `ja-JP`.

Note:
If the requested PL is not available, the system messages are returned in the PL that was used when the BI platform software was installed.

The content in the BI platform may be stored in multiple languages. For example, the BI platform could store a report that has been translated into French, Japanese, and German. Use the `X-SAP-PVL` request header attribute to specify the preferred language of the content to be returned. If the content is not available in the requested language, it is returned in the closest available language. For example, to request content that is available in French, set the `X-SAP-PVL` request header attribute to `fr-FR`.

For more information about HTML language codes, see the HTML 4.01 specification at http://www.w3.org.

2.2 Retrieving the base URL for RESTful web service requests

To use the Business Intelligence platform RESTful web service SDK, you must know the protocol, server name, port number, and path of the service that listens to RESTful web service requests. Collectively, these form the base URL. Whenever you make a request to RESTful web services, the beginning of the request starts with the base URL and is followed by the specific details of the request.

Basic installations of the BI platform that are installed on a single server use the default base URL, `http://<servername>:6405/biprws/`.

In complex deployment scenarios, there can be multiple instances of the Web Application Container Server (WACS), which hosts the RESTful web service. In this case, RESTful web services may be hosted at a different location. The BI platform administrator defines the location base URL that is used to access RESTful web services, and you can discover the base URL programmatically or through the Central Management Console (CMC).

2.2.1 Retrieving the base URL through the CMC

You can find the base URL for RESTful web service requests by logging on to the Central Management Console (CMC) user interface and navigating to the RESTful web services setting.

1. Log on to the CMC.
2. Click **Applications**.
3. Right-click **RESTful Web Service** and click **Properties**.

   The "RESTful Web Service" properties window appears.
4. Retrieve the base URL from the **Access URL** text box.

2.2.2 Retrieving the base URL programmatically

You can programmatically discover the base URL for RESTful web services by using one of the other BI platform SDKs, for example the BI Platform Java SDK. To programmatically find the base URL for RESTful web services, you must first query the BI platform to retrieve the `SI_ACCESS_URL` property of the RESTful web service object. You can query for the RESTful web service object by its CUID, or by its kind. You can find the CUID and kind by accessing the Java constants, `com.businessobjects.sdk.plugin.desktop.restwebservice.IRestWebService.CUID` and `com.businessobjects.sdk.plugin.desktop.restwebservice.IRestWebService.KIND`.

Note:
The CUID value for RESTful web services is `AZpJlb9HDtxPjLHwEmF8xD8` and the kind value is `RestWebService`.

```
"SELECT SI_ACCESS_URL FROM CI_APPOBJECTS WHERE SI_CUID='" + IRestWebService.CUID + "'"
```

```
"SELECT SI_ACCESS_URL FROM CI_APPOBJECTS WHERE SI_KIND=’" + IRestWebService.KIND + "’"
```

Finding the base URL by using the BI platform Java SDK version 4.1

You can use the `getURL` method of the `IRestWebService` interface to retrieve the RESTful web services base URL.

```
IInfoObjects objects= infostore.query("SELECT SI_ACCESS_URL FROM CI_APPOBJECTS WHERE SI_CUID='" + IRestWeb
Service.CUID + "'");
IInfoObject object = (IInfoObject)objects.get(0);
IRestWebService restAppObject = (IRestWebService) object;
String baseUrl = restAppObject.getURL();
```

For more information on the BI platform Java SDK, see the *SAP BusinessObjects Business Intelligence Platform Java SDK Developer Guide*.

2.3 Authentication

To access the BI platform through the Business Intelligence platform RESTful web service SDK, you need a logon token. You get one by making a request to a logon URL. The token proves you have been authenticated as a valid user, and it can be included with subsequent RESTful web services requests without exposing sensitive information such as your password.

You can use any one of the following information types to obtain authentication and a resulting logon token:

- BI platform logon credentials. This method supports WinAD, LDAP, SAP and Enterprise authentication. For more information about authentication, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.
- A session token from another session. If you have access to a session that has already been authenticated, you can use the session token to obtain a logon token.

  Note:
  The session token obtained from another SDK is not the same as the logon token, and cannot be used directly with RESTful web service requests.

- A serialized session. If you have access to a session that has already been authenticated, you can use it to obtain a logon token.

If your authentication request was successful, the response header includes a logon token. This logon token is defined by `X-SAP-LogonToken`.

Note:

The response body contains a copy of the logon token. However, this copy of the logon token is embedded in XML and has converted (encoded) illegal XML characters, such as `&`, `<` and `>` to an XML-friendly format. You must convert the XML encoded characters back to their original format before you can use this copy of the logon token. Alternatively, you can use the copy of the logon token that is provided in the response header, which has not been formatted for XML.

Each time you make a request to RESTful web services, you must add the `X-SAP-LogonToken` attribute to the request header, and set its value to be the logon token you received from being authenticated. Enclose the logon token in quotation marks, because it may contain characters that are not otherwise allowed in the request header.

The following table contains an example of a logon token:

Attribute	Sample Value
X-SAP-LogonToken	"COMMANDCOM-LCM:6400@{3&2=5604,U3&p=40623.9446463889,Y7&4F=12,U3&63=secEnterprise,0P&68=secEnterprise:Administrator,0P&qe=100,U3&vz=g5KUV8cAA.d_ARmSDnBy6T7jJVNyFCTso4s0q3dI.4k,UP}"

Related Topics

- Converting a logon token from XML-encoded text

2.3.1 To get a logon token from a user name and password

Before you can log on to the BI platform, you must have retrieved the base URL for RESTful web service requests.

To log on to the BI platform and obtain a logon token, make a request to `http://<baseURL>/logon/long` using the `POST` method, providing your user name, password, and type of authentication in the request body.

You can use the following types of authentication to log on to the BI platform:

- WinAD
- LDAP
- SAP
- Enterprise

To discover how to format the body of the logon request, make a request to the same URL, `http://<baseURL>/logon/long`, using the `GET` method. This response contains an XML template that can be used to format the request body of the logon request. The XML template includes a list of the supported authentication types.

1. Create a new HTTP request.
2. Add the `Accept` attribute to the request header, and set its value to `application/xml`.
3. Use the `GET` method to send the request to the `http://<baseURL>/logon/long` URL.

   Replace `<baseURL>` with the base URL for RESTful web services.

   ```
   GET http://localhost:6405/biprws/logon/long
   ```

   The response body contains a template.

   ```
   <attrs xmlns="http://www.sap.com/rws/bip">
     <attr name="userName" type="string"/></attr>
     <attr name="password" type="string"></attr>
     <attr name="auth" type="string" possibilities="secEnterprise,secLDAP,secWinAD,secSAPR3">secEnterprise</at
   tr>
   </attrs>
   ```

4. Create a new HTTP request.
5. Add the `Accept` attribute to the request header, and set its value to `application/xml`.
6. Add the `Content-Type` attribute to the request header, and set its value to `application/xml`.
7. Fill out the XML template with the user name, password, and authentication type, and add it to the request body of the new request.

   ```
   <attrs xmlns="http://www.sap.com/rws/bip">
     <attr name="userName" type="string">myUserName</attr>
     <attr name="password" type="string">myPassword</attr>
     <attr name="auth" type="string" possibilities="secEnterprise,secLDAP,secWinAD,secSAPR3">secEnterprise</at
   tr>
   </attrs>
   ```

8. Use the `POST` method to send the request to the same URL, `http://<baseURL>/logon/long`.

   Replace `<baseURL>` with the base URL for RESTful web services.

   ```
   POST http://localhost:6405/biprws/logon/long
   ```

   The response header returns the logon token as the `X-SAP-LogonToken` attribute.

   ```
   X-SAP-LogonToken:"COMMANDCOM-LCM:6400@{3&2=5595,U3&p=40674.9596541551,Y7&4F=12,U3&63=secEnter
   prise,0P&66=60,03&68=secEnterprise:Administrator,0P&qe=100,U3&vz=SFY6agrLPxpfQBK1ZKYCwoBZKCbfsQm7VgWZ
   FiH.RhM,UP"
   ```

   The logon token is contained between the quotation marks. In the example above, the logon token is as follows:

   ```
   COMMANDCOM-LCM:6400@{3&2=5595,U3&p=40674.9596541551,Y7&4F=12,U3&63=secEnterprise,0P&66=60,03&68=secEnter
   prise:Administrator,0P&qe=100,U3&vz=SFY6agrLPxpfQBK1ZKYCwoBZKCbfsQm7VgWZFiH.RhM,UP
   ```

   The response body contains a copy of the logon token in the `<attr>` element. If the logon token contains characters that are illegal in XML, they are replaced with their XML-encoded value. For example the `&` character is replaced with `&`. To use a logon token taken from the response body, you must convert the XML-encoded logon token back to its original format.

   The following example shows how the XML-encoded logon token appears in the response body:

   ```
   <entry xmlns="http://www.w3.org/2005/Atom">
     <author><name>@COMMANDCOM-LCM:6400</name></author>
   ```

```
  <id>tag:sap.com,2010:bip-rs/logon/long</id>
  <title type="text">Logon Result</title>
  <updated>2011-03-07T20:48:56.015Z</updated>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="logonToken" type="string">COMMANDCOM-
LCM:6400@{3&amp;2=5595,U3&amp;p=40674.9596541551,Y7&amp;4F=12,U3&amp;63=secEnter
prise,0P&amp;66=60,03&amp;68=secEnterprise:Administrator,0P&amp;qe=100,U3&amp;vz=SFY6agrLPxpfQBK1ZKYC
woBZKCbfsQm7VgWZFiH.RhM,UP}</attr>
    </attrs>
  </content>
</entry>
```

Related Topics

- Retrieving the base URL for RESTful web service requests
- Converting a logon token from XML-encoded text

2.3.2 To get a logon token from a serialized session or session token

To log on with this method, you must be able to use another BI platform SDK to access an existing authenticated session, for example, use the BI platform Java SDK. You must also know the base URL for RESTful web service requests.

You can get a logon token for RESTful web services from a valid session token or a serialized session. Make a request to the `http://<baseURL>/logon/token` URL using the `POST` method, and provide an XML-encoded version of the serialized session or session token in the request body. Replace `<baseURL>` with the base URL for RESTful web services.

To discover how to format the request body, make a request to the same URL, `http://<baseURL>/logon/token` using the `GET` method. The response from this request contains an XML template that can be used with the request body of the logon request.

By using a serialized session to obtain a logon token, you do not increase the number of concurrent user licenses used by the BI platform. However, using a session token will increase the concurrent user license count by one.

1. Create a new HTTP request.
2. Use the `GET` method to send the request to the `http://<baseURL>/logon/token` URL.

   Replace `<baseURL>` with the base URL for RESTful web services.

   ```
   GET http://localhost:6405/biprws/logon/token
   ```

   The response contains an XML template.

   ```
   <attrs xmlns="http://www.sap.com/rws/bip">
     <attr name="tokenType" type="string" possibilities="token, serializedSession">token</attr>
     <attr name="logonToken" type="string" null="true"></attr>
   </attrs>
   ```

3. Create a new HTTP request.
4. Add the `Content-Type` attribute to the request header, and set its value to `application/xml`.
5. Fill out the XML template and add it to the request body.

Set the value of the `<attr name ="tokenType" type="string">` element to be `token` if you are using a session token, and set it to `serializedSession` if you are using a serialized session. Set the value of the `<attr name="logonToken" type="string">` element to an XML-encoded version of the serialized session or session token value.

```
<attrs xmlns="http://www.sap.com/rws/bip">
  <attr name="tokenType" type="string" possibilities="token, serializedSession">serializedSession</attr>

  <attr name="logonToken" type="string">3&amp;ua=AWmaEx4Z.NVPpAEthuTGAjc,8P&amp;ub=AfRWaT5_131N1LLf5bRM
LKY,8P&amp;S5,88&amp;5U=5320JaqlNvF1mr4m8u5UQFadItj5319JWKkfBwlKLBfrgXC8Npg1jC,8P&amp;63=secEnter
prise,8P&amp;2r=COMMANDCOM-LCM:6400,8P&amp;3k=@COMMANDCOM-LCM:6400,8P&amp;1=Administrator ac
count,8P&amp;W={},?z&amp;4E=5319JWKkfBwlKLBfrgXC8Npg1jC,8P&amp;Tn={3&amp;.1={3&amp;2=726,03&amp;O=Fa
voritesFolder,0P},2z&amp;.2={3&amp;2=727,03&amp;O=PersonalCategory,0P},2z&amp;.3={3&amp;2=728,03&amp;O=In
box,0P},2z&amp;U=3,03},?z&amp;4F=12,8P&amp;Tm=36500,83&amp;uy=-1043,8L&amp;35=Administrator,8P&amp;ux=Ae
iCInd_R6lBrV98duvX1dc,8P&amp;pa,8P</attr>
</attrs>
```

Note:

This example shows a serialized session. The serialized session or session token value must be XML-encoded to remove illegal XML characters. For example, replace the `&` character with `&`.

6. Use the `POST` method to send the request to the same URL, `http://<baseURL>/logon/token`. Replace `<baseURL>` with the base URL for RESTful web services.

```
POST http://localhost:6405/biprws/logon/token
```

The response header returns the logon token as the `X-SAP-LogonToken` attribute.

```
X-SAP-LogonToken:"COMMANDCOM-LCM:6400@{3&2=5595,U3&p=40674.9596541551,Y7&4F=12,U3&63=secEnter
prise,0P&66=60,03&68=secEnterprise:Administrator,0P&qe=100,U3&vz=SFY6agrLPxpfQBK1ZKYCwoBZKCbfsQm7VgWZ
FiH.RhM,UP"
```

The logon token is contained between the quotation marks.

Note:

The response body contains a copy of the logon token in the `<attr>` element. If the logon token contains characters that are illegal in XML, they are replaced with their XML-encoded value. For example, the `&` character is replaced with `&`. To use a logon token taken from the response body, you must convert the XML-encoded logon token back to its original format.

The following example shows how the XML-encoded logon token appears in the response body:

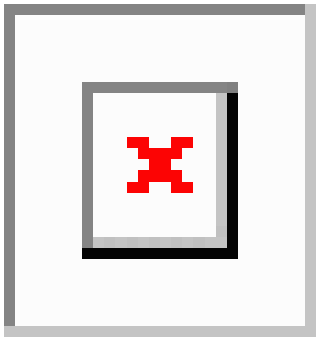
```
<entry xmlns="http://www.w3.org/2005/Atom">
  <author><name>@COMMANDCOM-LCM:6400</name></author>
  <id>tag:sap.com,2010:bip-rs/logon/token</id>
  <title type="text">Logon Result</title>
  <updated>2011-06-28T17:54:31.994Z</updated>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="logonToken" type="string">COMMANDCOM-
LCM:6400@{3&amp;2=5319,U3&amp;p=40722.7462034491,Y7&amp;4F=12,U3&amp;63=secEnter
prise,0P&amp;66=60,03&amp;68=secEnterprise:Administrator,0P&amp;qe=100,U3&amp;vz=Ke
Du7064jWSptBT_m5BkBJ5Q_NaxyvE_WStqXmigYrg,UP}</attr>
    </attrs>
  </content>
</entry>
```

Related Topics

- Retrieving the base URL for RESTful web service requests
- Using authenticated sessions obtained from other SDKs
- Converting a logon token from XML-encoded text

2.3.3 To get a logon token using an Active Directory Single Sign-On (AD SSO) account

To use the Active Directory Single Sign-On feature of RESTful Web Services, clients must have a Windows Active Directory (WinAD) account and be logged into the computer that will be using the `/logon/adsso` API. Clients must also have logon accounts on the BI platform that match the WinAD accounts. The following diagram illustrates the configuration and authentication relationship between the BI platform server, the client computer, and the Windows Active Directory server.



Once the WinAD SSO feature is enabled as described in `Administration and installation tasks > To configure web.xml to enable WinAD SSO`, clients can use their WinAD credentials to log on to their computer. Those credentials will be used to authenticate them for access to the BI platform server automatically.

Use the following steps to obtain a logon token through AD SSO.

1. Create a new HTTP request.
2. Use the `GET` method to send the request to `http://<baseURL>/logon/adsso`.

   Replace `<baseURL>` with the base URL for RESTful web services.

   For example:

   ```
   GET http://localhost:6405/biprws/logon/adsso
   ```

   The response header returns the logon token as the `X-SAP-LogonToken` attribute. An example XML response appears as follows:

   ```
   <?xml version="1.0" ?>
   <entry xmlns="http://www.w3.org/2005/Atom">
     <author>
       <name>
         @BOESRVR.ADDOM.COM
       </name>
     </author>
     <id>
       tag:sap.com,2010:bip-rs/logon/adsso
     </id>
     <title type="text">
       Logon Result
     </title>
     <updated>
       2011-11-11T11:11:11.340Z
     </updated>
     <content type="application/xml">
   ```

```
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="logonToken" type="string">
        BOESRVR.ADDOM.COM:6400@{3&2=4584,U3&p=40868.9276775116,Y7&4F=4331,U3
        &63=secWinAD,0P&66=60,03&68=secWinAD:CN%3DADUser1%2CCN%3DUsers%2CDC%3D
        ADDOM%2CDC%3DCOM,0P&qe=100,U3&vz=
        kOox8TDqAiFsfs8T3GefI3sWXIyKymc9qvytAjihC7w,UP}
      </attr>
    </attrs>
  </content>
</entry>
```

3. Use the resulting `X-SAP-LogonToken` within an HTTP request header to make further RESTful Web Service requests (for example `http://<baseURL>/infostore`.) You can also HTTP-encode the logon token and append it to an OpenDocument URL with the `&token=<logonToken>` parameter.

Related Topics

- Retrieving the base URL for RESTful web service requests
- Converting a logon token from XML-encoded text

2.3.4 To get a logon token using trusted authentication

To use the trusted authentication feature of RESTful Web Services, the features must be activated as described in *Administration and Installation tasks > To enable and configure trusted authentication*.

Trusted authentication is used to speed up access to protected resources once users have already been authenticated elsewhere; for example, after users have logged in with a Windows account.

The methods of logon token retrieval, using trusted authentication, are as follows:

- HTTP header requests using a customizable header for the user name.
- URL queries.
- Cookie authentication.

To use one of the three trusted authentication logon retrieval methods, open CMC and go to **WACS** > "Trusted Authentication Configuration", in the **Retrieving Method** menu, change the option to match the method you will be using. For all trusted authentication methods, there is an option to change the **Name Parameter**, which is found in **Servers** > **Core Services** > **WACS**. Note that all URLs and values supplied are case sensitive.

Retrieving Method	RESTful API used	Usage instructions
HTTP_HEADER	/logon/trusted	1. Create an HTTP request using the `GET` method. 2. Use the `/logon/trusted` API, for example, `http://localhost:6405/biprws/logon/trusted` 3. Create a request header with the default label `X-SAP-TRUSTED-USER`, and add a trusted user name, for example `bob`. The resulting logon token is displayed in the response header.
QUERY_STRING	/logon/trusted?<MyUser>=<Username>	1. In a web browser URL, use the `/logon/trusted` API, and add the user name parameter and the user name, for example, `http://localhost:6405/biprws/logon/trusted?MyUser=bob`. For example: • Replace `MyUser` with a customized user name parameter that is set in CMC under **Servers > Core Services > WACS >**"Trusted Authentication Configuration". • Replace `bob` with a that of a trusted user that is set in CMC under **Users and Groups** > **User List**. The resulting logon token is displayed in the browser body window.
COOKIE	/logon/trusted	1. Create a cookie, and add the following information: • The domain. For example, `localhost`. • The name label, for example the default value of `X-SAP-TRUSTED-USER`, with the value for the logon name, for example, `bob`. • The path, for example `/` (forward slash). 2. Enter the URL, for example, `http://localhost:6405/biprws/logon/trusted` and press the **Enter** key to see the resulting logon token displayed in the browser window.

Related Topics

- Retrieving the base URL for RESTful web service requests
- Converting a logon token from XML-encoded text

2.3.5 Converting a logon token from XML-encoded text

Logon tokens are returned in both the response header and the response body of authentication responses. The response body is formatted as XML, which reserves certain characters for its own use. If the logon token contains these characters, they are replaced with character sequences that are allowed to be embedded in XML but will not work in a logon token. Before you can use an XML-encoded logon token, it must be converted back to its original format.

Note:
You only need to perform this step if you retrieve the logon token from the response body. The logon token that is contained in the response header is not XML-encoded.

To convert an XML-encoded logon token to its original format, replace each XML-encoded character sequence with the character it represents. For example, replace the `&` character encoding with the `&` character.

The following table shows the examples of the most common XML encoding of illegal XML characters.

XML encoding	Character
'	'
"	"
&	&
<	<
>	>

For more information about representing characters in XML, refer to the specification for extensible markup language at *http://www.w3.org*.

Example:

This example shows a XML-encoded logon token.

```
COMMANDCOM-LCM:6400@{3&amp;2=5675,U3&amp;p=40653.0083064583,Y7&amp;4F=12,U3&amp;63=secEnter
prise,0P&amp;66=60,03&amp;68=secEnterprise:Administrator,0P&amp;qe=100,U3&amp;vz=y3EqvsvoehahHhbmPrpaPjKV
MU8raN3zEpnt2YjqDe4,UP}
```

The example shows the logon token after it has been converted to its original format.

```
COMMANDCOM-LCM:6400@{3&2=5675,U3&p=40653.0083064583,Y7&4F=12,U3&63=secEnterprise,0P&66=60,03&68=secEnter
prise:Administrator,0P&qe=100,U3&vz=y3EqvsvoehahHhbmPrpaPjKVMU8raN3zEpnt2YjqDe4,UP}
```

2.3.6 To add a logon token to a request header

Once you have obtained a logon token, you can use it to authenticate RESTful requests that access the BI platform.

Note:
If you obtained the logon token from the request body, you must convert it from its XML-encoded format back to its original format. Alternatively, you can obtain the original logon token directly from the response header.

For example, this text represents a logon token that is embedded in the XML of a response body.

```
COMMANDCOM-LCM:6400@{3&amp;2=5675,U3&amp;p=40653.0083064583,Y7&amp;4F=12,U3&amp;63=secEnter
prise,0P&amp;66=60,03&amp;68=secEnterprise:Administrator,0P&amp;qe=100,U3&amp;vz=y3EqvsvoehahHhbmPrpaPjKV
MU8raN3zEpnt2YjqDe4,UP}
```

This text represents a logon token obtained for a response header, or a token obtained from a response body that has been converted back to its original format.

```
COMMANDCOM-LCM:6400@{3&2=5675,U3&p=40653.0083064583,Y7&4F=12,U3&63=secEnterprise,0P&66=60,03&68=secEnter
prise:Administrator,0P&qe=100,U3&vz=y3EqvsvoehahHhbmPrpaPjKVMU8raN3zEpnt2YjqDe4,UP}
```

1. Create a new RESTful web service request or modify an existing request.
2. Add an attribute to the request header.
3. Set the name of the attribute to `X-SAP-LogonToken`.
4. Set the value of the attribute to the logon token value, and enclose the value in quotation marks.

Name	Value
X-SAP-LogonToken	"COMMANDCOM-LCM:6400@{3&2=5604,U3&p=40623.9456463889,Y7&4F=12,U3&63=secEnterprise,0P&68=secEnterprise:Administrator,0P&qe=100,U3&vz=g5KUU8cAA.d_ARmSDnBy6T7jJVNyFCTso4s0q3dI.4k,UP}"

Related Topics

- Converting a logon token from XML-encoded text

2.3.7 Using HTTP basic authentication

Use HTTP basic authentication to log on to the BI platform without including a logon token in the HTTP header of the RESTful web service request. Instead, you provide your user name, password, and an authentication type.

Note:
User names and passwords are not transmitted securely using HTTP basic authentication, unless they are used in conjunction with HTTPS.

HTTP basic authentication must be enabled by an administrator. The administrator may also define a default authentication type that is used if you do not specify an authentication type.

Authentication types

You can use the following authentication types with HTTP basic authentication:

- `secEnterprise` - Enterprise authentication
- `secLDAP` - LDAP authentication
- `secWinAD` - Windows AD authentication
- `secSAPR3` - SAP authentication

Making requests using HTTP authentication consumes a license. If session caching is not used, a license is consumed for the duration of the request and is released once the request is completed. If session caching is used, the license associated with the cached session is used.

Note:
The user name, password, and authentication type must be base64-encoded as defined by RFC 2716. User names that contain the `:` character cannot be used with HTTP basic authentication.

Using HTTP basic authentication in a web browser

To log on with a web browser using the default authentication type, provide your user name and password at the prompt.

To log on using a particular authentication type, use `<authenticationType>\<username>` in the user name field, and provide your password in the password prompt. Replace `<authenticationType>` with the type of authentication, and `<username>` with your user name. For example, to log on using SAP authentication with the user name `myUserName`, enter `secSAPR3\myUserName` in the user name field, and enter your password in the password field.

Using HTTP basic authentication programmatically

To use HTTP basic authentication programmatically, add the `Authorization` attribute to the request header, and set its value to be the base64-encoded value of the authorization string.

Use the following authorization string to use the default authentication type:

```
Basic <username>:<password>
```

Use the following authorization string to use a specific authentication type:

```
Basic <authtype>\<username>:<password>
```

2.3.8 To log off the BI platform

Before you can log off the BI platform, you must know the base URL for RESTful web service requests. You also must have the logon token for the session that you want to invalidate.

Logon tokens expire automatically if they are not used for a set time. By default, logon tokens expire after one hour of inactivity, but this value can be configured by an administrator. To log off of your

session before it expires automatically, make a `POST` request to the `http://<baseURL>/logoff` URL. Replace `<baseURL>` with the base URL for RESTful web services.

By logging off the BI platform, you invalidate the logon token and release any license that is associated with the session.

1. Create a new HTTP request.
2. Add the `Accept` attribute to the request header, and set its value to `application/xml`.
3. Add the `X-SAP-LogonToken` attribute to the request header, and set its value to the logon token value, enclosed in quotation marks.

Name	Value
X-SAP-LogonToken	"COMMANDCOM-LCM:6400@{3&2=5604,U3&p=40623.9456463889,Y7&4F=12,U3&63=secEnterprise,0P&68=secEnterprise:Administrator,0P&qe=100,U3&vz=g5KUU8cAA.d_ARmSDnBy6T7jJVNyFCTso4s0q3dI.4k,UP}"

4. Use the `POST` method to send the request to the `http://<baseURL>/logoff` URL.

   Replace `<baseURL>` with the base URL for RESTful web services.

```
POST http://<baseURL>/logoff
```

   If the logoff attempt was successful, the response header contains the HTTP status code 200.

Related Topics

- Retrieving the base URL for RESTful web service requests

2.3.9 Using authenticated sessions obtained from other SDKs

You can use another BI platform SDK to obtain a serialized session or session token from an existing authenticated session. You can then obtain a logon token for the Business Intelligence platform RESTful web service SDK by providing the serialized session or session token in a request to the `/logon/token` URL.

You can use serialized sessions or session tokens obtained from the following SDKs, version XI 3.0 and later:

- SAP BusinessObjects Business Intelligence platform Java SDK
- SAP BusinessObjects Business Intelligence platform .NET SDK
- SAP BusinessObjects Business Intelligence platform Web Services SDK

2.3.9.1 Getting session information with the BI platform Java SDK

You can use the BI platform Java SDK to obtain a serialized session or session token from an existing session that has already been authenticated. Provide the serialized session or session token in the body of a request to the `/logon/token` URL to obtain a logon token for the Business Intelligence platform RESTful web service SDK.

To get a serialized session, use the `getSerializedSession` method of the `IEnterpriseSession` class.

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
IEnterpriseSession enterpriseSession = sessionMgr.logon("username", "password", "cmsname", "secEnterprise");
String serializedSession = enterpriseSession.getSerializedSession();
```

To get a session token, use the `getDefaultToken` or the `createLogonToken` method of the `ILogonTokenMgr` class.

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
IEnterpriseSession enterpriseSession = sessionMgr.logon("username", "password", "cmsname", "secEnterprise");
String sessionToken = enterpriseSession.getLogonTokenMgr().getDefaultToken();
```

For more information about using the BI platform Java SDK, see the *SAP BusinessObjects Business Intelligence Platform Java SDK Developer Guide*.

2.3.9.2 Getting session information with the BI platform .NET SDK

You can use the BI platform .NET SDK to obtain a serialized session or session token from an existing session that has already been authenticated. Provide the serialized session or session token in the body of a request to the `/logon/token` URL to obtain a logon token for the Business Intelligence platform RESTful web service SDK.

To get a serialized session, use the `SerializedSession` property of the `EnterpriseSession` class.

```
SessionMgr sessionMgr = new SessionMgr();
EnterpriseSession session = sessionMgr.Logon("username", "password", "cms", "secEnterprise");
string serializedSession = session.SerializedSession;
```

To get a session token, use the `SerializedSesion` property or the `CreateLogonTokenEx` method of the `LogonTokenMgr` class.

```
SessionMgr sessionMgr = new SessionMgr();
EnterpriseSession session = sessionMgr.Logon("username", "password", "cms", "secEnterprise");
string logonTokenMgr = session.LogonTokenMgr.DefaultToken;
```

2.3.9.3 Getting session information with the BI platform Web Services SDK

You can use the BI platform Web Services SDK to obtain a serialized session or session token from an existing session that has already been authenticated. Provide the serialized session or session token in the body of a request to the `/logon/token` URL to obtain a logon token for Business Intelligence platform RESTful web service SDK.

To get a serialized session, use the `getSerializedSession` method of the `SessionInfo` class.

```
URL boConURL = new URL("http://boserver:port/dswsbobje/services/Session");
Connection connection = new Connection(boConURL);
Session session = new Session(connection);
EnterpriseCredential credential = EnterpriseCredential.Factory.newInstance();
credential.setLogin("username");
credential.setPassword("password");
credential.setDomain("domain");
credential.setAuthType("secEnterprise");
SessionInfo sessionInfo = session.login(credential);
String serializedSession = sessionInfo.getSerializedSession();
```

To get a session token, use the `getDefaultToken` method of the `SessionInfo` class.

```
URL boConURL = new URL("http://boserver:port/dswsbobje/services/Session");
Connection connection = new Connection(boConURL);
Session session = new Session(connection);
EnterpriseCredential credential = EnterpriseCredential.Factory.newInstance();
credential.setLogin("username");
credential.setPassword("password");
credential.setDomain("domain");
credential.setAuthType("secEnterprise");
SessionInfo sessionInfo = session.login(credential);
String sessionToken = sessionInfo.getDefaultToken();
```

For more information about using the BI platform Web Services Consumer Java SDK, see the *SAP BusinessObjects Business Intelligence Platform Web Services Consumer Java SDK Developer Guide*.

2.4 Retrieving the report ID

You need the report ID to access a Crystal report using the RESTful web services. You can discover the report ID programmatically or through the Central Management Console (CMC). When you right-click a report in the CMC, the report ID is be listed with the report properties.

For more information on navigating the BI platform repository, see the *Business Intelligence Platform RESTful Web Services Developer Guide*.

2.4.1 To retrieve the report ID programatically

You must have a valid logon token, and you must know where the report is stored in the BI platform repository.

Navigate through the repository using the RESTful web services to get the report ID.

1. Create a new HTTP request.
2. Add the `X-SAP-LogonToken` attribute to the request header and set its value to be a valid logon token.
3. Use the GET method to send the request to the `http://<baseURI>/infostore/Root%20Folder/children` URL

   The method returns a summary of content located in the root directory of the BI platform repository. Folder ID's are returned in the request response. In this example response the ID of the `Report Samples` folder is 5375.

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>tag:sap.com,2010:bip-rs/ASHnC0S_Pw5LhKFbZ.iA_j4/children</id>
  <title type="text">Children of Root Folder</title>
  <updated>2011-05-30T20:02:41.470Z</updated>
  ...
  <entry>
    <title type="text">Report Samples</title>
    <id>tag:sap.com,2010:bip-rs/AfwlNA25oGlFr8gN2TxHddA</id>
    <author>
      <name>Administrator</name>
      <uri>http://localhost:6405/biprws/infostore/12</uri>
    </author>
   <link href="http://localhost:6405/biprws/infostore/5375" rel="alternate"></link>
   <content type="application/xml">
   <attrs xmlns="http://www.sap.com/rws/bip">
   <attr name="cuid" type="string">AfwlNA25oGlFr8gN2TxHddA</attr>
   <attr name="description" type="string">Contains sample reports.</attr>
   <attr name="type" type="string">Folder</attr></attrs></content>
  </entry>
  ...
</feed>
```

4. Create a new HTTP request.
5. Add the `X-SAP-LogonToken` attribute to the request header and set its value to be a valid logon token.
6. Use the GET method to send the request to the `http://<baseURI>/infostore/<folderID>/children/` URL.

   Note:
   Replace `<folderID>` with the folder ID.

   The method returns a summary of the folder content and corresponding object ID's. Reports are listed by name, and the report ID is provided with the content summary. In this example response the ID of the `Drilldown` report is 5567.

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>tag:sap.com,2010:bip-rs/AYLus319xBdAh0yC3SCxU6I/children</id>
  <title type="text">Children of Feature Samples</title>
  ...
  <entry>
    <title type="text">Drilldown</title>
    <id>tag:sap.com,2010:bip-rs/AbHK3o8Bnz1MpOXtKWa_1V8</id>
    <author>
      <name>Administrator</name>
      <uri>http://localhost:6405/biprws/infostore/12</uri>
    </author>
  <link href="http://localhost:6405/biprws/infostore/5567" rel="alternate"></link>
  <content type="application/xml">
  <attrs xmlns="http://www.sap.com/rws/bip">
    <attr name="cuid" type="string">AbHK3o8Bnz1MpOXtKWa_1V8</attr>
```

```
    <attr name="description" type="string"></attr>
    <attr name="type" type="string">CrystalReport</attr>
  </attrs>
  </content>
  </entry>
  ...
</feed>
```

Note:
Repeat steps 4 through 6 to navigate through nested folders.

Related Topics

- Report URI

Using the SAP Crystal Reports RESTful web services API

The RESTful web services API is strictly a tool for manipulating data in existing reports. It cannot create new SAP Crystal reports.

SAP Crystal Reports RESTful web services comply with RESTful methodology. You can use `GET` to retrieve reports and `POST` to modify transient report instances. Retrieved results are user readable and do not contain extra markup.

RESTful web service requests are stateless. A typical report job created by the server has an inactive lifetime of an hour before being recycled. If you request a report that has been recently requested from the server, the existing report job may be shared.

Note:
The `PUT, UPDATE, DELETE, MERGE, HEAD`, and `OPTIONS` methods are not supported.

3.1 Common URIs

Frequently used URI's:

- http://<baseURI>
- http://<baseURI>/infostore/<reportID>/rpt
- http://<baseURI>/infostore/<reportID>/rpt/export?mime_type=application/pdf
- http://<baseURI>/infostore/<reportID>/rpt/data.svc
- http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows
- http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows(<INDEX>)
- http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows(<INDEX>)/<PROPERTY>
- http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows(<INDEX>)/<PROPERTY>/$<VALUE>
- http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows?$skip=<VALUE>
- http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows?$skiptoken=<VALUE>
- http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows?$inlinecount=allpages
- http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows/$count
- http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows?<PARAMETER>=<VALUE>
- http://<baseURI>/infostore/<reportID>/rpt/GrandTotals
- http://<baseURI>/infostore/<reportID>/rpt/instance

Related Topics

- Retrieving the base URL for RESTful web service requests

- Retrieving the report ID

3.2 Report URI

Allows the retrieval and interpretation of report information.

Note:
The default representation format is XML.

URI	Operations	Supported format	Is AtomPub
`<baseURI>/infostore/<reportID>/rpt`	`GET`	XML, JSON	No

Example: **GET http://<baseURI>/infostore/<reportID>/rpt/**

Make a GET request to retrieve report summary information.

Request:

- Method: `GET`
- URL: `http://<baseURI>/infostore/<reportID>/rpt`
- Request header attributes: `X-SAP-LogonToken`
- Request body: none

Response:

- An entry is returned that contains report summary information.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<report_summary_info xmlns="http://schemas.sap.com/crystalreports/report_summary_info">
  <report_name>DateAndTime.rpt</report_name>
  <report_title></report_title>
  <file_format_version>
    <major_version>12</major_version>
    <minor_version>0</minor_version>
  </file_format_version>
  <author></author>
  <subject></subject>
  <keywords></keywords>
  <comments></comments>
  <last_data_refresh_date>2010-12-17T11:59:51.000</last_data_refresh_date>
  <export_uri>http://localhost:6405/biprws/infostore/6231/rpt/export</export_uri>
  <service_uri>http://localhost:6405/biprws/infostore/6231/rpt/data.svc</service_uri>
  <edit_uri>http://localhost:6405/biprws/infostore/6231/rpt/instance</edit_uri>
</report_summary_info>
```

3.3 Report instances

A report instance is a temporary copy of a report stored in the BI platform repository. The GET method returns only report instances.

When you use the GET method to request an instance of a report that contains saved data, the RESTful web service will show results that reflect data saved the last time the report was refreshed in the repository. This is indicated by the `last_refresh_date` property in the request response body. When you use the GET method to request an instance of a report that does not contain saved data, you can view report metadata and summary information, but requests that access report data will fail.

You can POST report data only to a transient report instance. Therefore, the report contained in the repository will not be modified by the RESTful web service. The data returned by the instance will reflect data refreshed at the time the latest POST to the instance was performed.

Note:
When you POST field names to a transient report instance, field names are limited to 480 characters.

To view the data in a report without saved data, you must create a transient instance.

By default, a transient instance expires after an hour of inactivity. If the server is overloaded, an instance may be recycled to make room for a new job before one hour has elapsed.

To create a transient instance you use a GET method to the `/instance` URI. This returns a form that must be filled out and submitted using the POST method. The form contains entries containing connection information, and any parameters used by the report. The form also contains a `SuppressData` entry that is set to false by default.

- If `SuppressData` is set to `false`, rowset data will be retrieved from the report's data source and included in the transient report instance.
- If `SuppressData` is set to `true`, the transient report instance will not include any data. This is useful in scenarios where the report's data will be supplied programmatically using the POST to rows method in the data service.

  Note:
  This option is only available for reports created without connections to universes.

The POST method creates the transient instance, and the response includes the instance ID link to the new transient instance.

To use the transient instance, add the instance ID to the URI. For example:

```
http://<baseURI>/infostore/<reportID>/rpt/<instanceID>/data.svc/Rows
```

Note:
Replace `<instanceID>` with the ID of the instance you have created.

URI	Operations	Supported format	Is AtomPub
`<baseURI>/infostore/<reportID>/rpt/instance`	`GET`, `POST`	XML	yes

Related Topics

- Rows

3.3.1 To create a transient report instance from an existing report

1. Create a new HTTP request.
2. Use the GET method to send the request to this URL: `http://<baseURI>/infostore/<reportID>/rpt/instance` URL.

   Example request:

   - Method: `GET`
   - URL: `http://<baseURI>/infostore/<reportID>/rpt/instance`
   - Request header attributes: `X-SAP-LogonToken`
   - Request body: none

   Example response:

   - The response contains a form.

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="text">Crystal Reports New Instance</title>
  <entry>
    <title type="text">SuppressData</title>
    <content type="application/xml">
    <attrs>
      <attr name="value" type="boolean">false</attr>
    </attrs>
    </content>
  </entry>
  <entry>
    <title type="text">ConnectionInfo</title>
    <id>reportse--</id>
    <content type="application/xml">
    <attrs>
      <attr name="ServerName" type="string">reportse</attr>
      <attr name="DatabaseName" type="string"></attr>
      <attr name="userName" type="string"></attr>
      <attr name="password" type="string"></attr>
    </attrs>
    </content>
  </entry>
</feed>
```

3. Create another HTTP request.
4. Fill the form and use a POST method to send the request to the `http://<baseURI>/infostore/<reportID>/rpt/instance` URL.

   Example request:

- Method: `POST`
- URL: `http://<baseURI>/infostore/<reportID>/rpt/instance`
- Request header attributes: `X-SAP-LogonToken`, `Content-Type`
- Request body: Fill out the form retrieved by the GET request.

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="text">Crystal Reports New Instance</title>
  <entry>
    <title type="text">SuppressData</title>
    <content type="application/xml">
    <attrs>
      <attr name="value" type="boolean">false</attr>
    </attrs>
    </content>
  </entry>
  <entry>
    <title type="text">ConnectionInfo</title>
    <id>reportse--</id>
    <content type="application/xml">
    <attrs>
        <attr name="ServerName" type="string">reportse</attr>
        <attr name="DatabaseName" type="string">HOSTID</attr>
        <attr name="userName" type="string">USER</attr>
        <attr name="password" type="string">PASSWORD</attr>
      </attrs>
    </content>
  </entry>
</feed>
```

Note:
Replace `<HOSTID>` with the database name, `<USER>` with the user name, and `<PASSWORD>` with the password. Parameter fields may also appear in the form if the report includes parameter prompts.

Example response:

- Response Header:

```
HTTP/1.1 201 Created
Content-Type: application/xml
Location: http://localhost:6405/biprws/1667/rpt/eNotTsEKgzAU
  _5r2rBXnduhhqAdhA9GNnUt5KzLbynvdxL.fc_ySkISQ3HRex.
  ntg1AtLkm2OpO9hiAHXVbVSXZ6FEVjo_ecwDiTgOIzrQaBnck7xkjWMC3GAQF
  _AFkIVSq1TkG9jqh_Yofmuo3dmeMaN0pmHmCJmKjHaIFoCm789w8PXs0Ledmv3HX2BT49NO8
Date: Fri, 27 May 2011 21:52:19 GMT
Content-Length: 513
```

- Response body:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<reportInstance>
  <report_name>DateAndTime.rpt</report_name>
  <resource>http://localhost:6405/biprws/infostore/1667/rpt/eNotTsEKgzAU
    _5r2rBXnduhhqAdhA9GNnUt5KzLbynvdxL.fc_ySkISQ3HRex.
    ntg1AtLkm2OpO9hiAHXVbVSXZ6FEVjo_ecwDiTgOIzrQaBnck7xkjWMC3GAQF
    _AFkIVSq1TkG9jqh_Yofmuo3dmeMaN0pmHmCJmKjHaIFoCm789w8PXs0Ledmv3HX2BT49NO8</resource>
  <id>eNotTsEKgzAU_5r2rBXnduhhqAdhA9GNnUt5KzLbynvdxL.
    fc_ySkISQ3HRex.ntg1AtLkm2OpO9hiAHXVbVSXZ6FEVjo
    _ecwDiTgOIzrQaBnck7xkjWMC3GAQF_AFkIVSq1TkG9jqh
    _Yofmuo3dmeMaN0pmHmCJmKjHaIFoCm789w8PXs0Ledmv3HX2BT49NO8</id>
</reportInstance>
```

The response body contains the URI to a new instance, and the unique ID associated with the instance. The new instance ID is also accessible in the response header.

3.4 Exporting

Exports a report to the specified MIME type.

URI	Operations	Supported format	Is AtomPub
<baseURI>/infostore/<reportID/rpt/export?mime_type=<MIMETYPE>	GET		No

Note:
Replace <MIMETYPE> with the type of file that you want to export to.

MIME types	Description
text/csv	Character Separated Values (CSV)
application/vnd.ms-excel	Microsoft Excel (97-2003)
application/vnd.ms-excel&isDataOnly=true	Microsoft Excel Data-Only (97-2003)
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	Microsoft Excel Workbook Data-Only
application/msword	Microsoft Word
application/msword&isEditable=true	Microsoft Word Editable
application/PDF	PDF
application/rtf	Rich Text Format (RTF)
text/ttx	Tab Separated Text (TTX)
text/plain	Text
application/xml	XML

Note:
The request can be customized with optional parameters.

Optional parameters	Type	Description
`startPageNumber=<VALUE>`	integer	`<VALUE>` specifies the first page to export.
`endPageNumber=<VALUE>`	integer	`<VALUE>` specifies the first page to export.

Note:
All pages are exported if start and end page numbers are not specified.

Example: **GET http://<baseURI>/infostore/<reportID>/rpt/export?mime_type=application/PDF**

Make a GET request to export a report to PDF.

Request:

- Method: `GET`
- URL: `http://<baseURI>/infostore/<reportID>/rpt/export?mime_type=application/PDF`
- Request header attributes: `X-SAP-LogonToken`
- Request body: none

Response:

- Response header:

```
Status Code: 200 OK
Server: Apache-Coyote/1.1
Date: Mon 30, May 2011 22:23:22 GMT
Content-Type: application/PDF
Transfer-Encoding: chunked
```

- Response body: contains the PDF encoding of the report

3.4.1 Character separated value (CSV) parameters

The following parameters can be used to format a report exported in character separated value (CSV) format.

Optional parameter	Type	Description
?delimiter=<VALUE>	string	<VALUE> will be used as the delimiter in the report.
?separator=<VALUE>	string	<VALUE> is used as the character separator in the report.
?reportSectionsOption=<VALUE>	integer, string	<VALUE>sets the report section option using the following values: • 0 = Export • 1 = ExportIsolated • 2 = doNotExport **Note:** • You can use either the integer or the string to set the base area type. For example: ?reportSectionsOption="Export" • Other values will result in an exception being thrown.
?groupSectionsOption=<VALUE>	integer, string	<VALUE> sets the group sections option using the following values: • 0 = Export • 1 = ExportIsolated • 2 = doNotExport **Note:** • You can use either the integer or the string to set the base area type. For example: ?groupSectionsOption="Export" • Other values will result in an exception being thrown.

3.4.2 Microsoft Excel parameters

The following optional parameters can be used to format a report exported to Microsoft Excel.

Optional parameter	Type	Description
?constantWidth=<VALUE>	integer	<VALUE> sets the constant width of the report. **Note:** constantWidth and baseAreaType are mutually exclusive. If both exist in a query, constant width will take precedence.
?baseAreaType=<VALUE>	integer, string	<VALUE> sets the base area type, using one of the following values: • 1 = ReportHeader • 2 = PageHeader • 3 = GroupHeader • 4 = Detail • 5 = GroupFooter • 7 = Page Footer • 8 = ReportFooter • 255 = WholeReport **Note:** • You can use either the integer or the string to set the base area type. For example: ?baseAreaType="ReportHeader" • Other values will result in an exception being thrown. • constantWidth and baseAreaType are mutually exclusive. If both exist in a query, constant width will take precedence.
?exportPageHeaderFooter=<VALUE>	integer	<VALUE> sets the page header and footer using one of the following values. • 0 = None • 1 = Once • 2 = Each Page **Note:** Other values will result in an exception being thrown.

3.4.3 Microsoft Excel Data-Only parameters

The following parameters can be used to format or return information on a report exported to Microsoft Excel Data-Only.

Optional parameter	Type	Description
`?constantWidth=VALUE`	integer	`VALUE` sets the width of the report. **Note:** `constantWidth` and `baseAreaType` are mutually exclusive. If both exist in a query, constant width will take precedence.
`?baseAreaType=VALUE`	integer, string	`VALUE` sets the base area type using one of the following values: • 1 = ReportHeader • 2 = PageHeader • 3 = GroupHeader • 4 = Detail • 5 = GroupFooter • 7 = Page Footer • 8 = ReportFooter • 255 = WholeReport **Note:** • You can use either the integer or the string to set the base area type. For example: `?baseAreaType="ReportHeader"` • Other values will result in an exception being thrown. • `constantWidth` and `baseAreaType` are mutually exclusive. If both exist in a query, constant width will take precedence.
`?exportPageHeaderFooter=VALUE`	integer	`VALUE` exports the page header and footer using one of the following values: • 0 = None • 1 = Once • 2 = Each Page **Note:** Other values will result in an exception being thrown.
`?isFormatUsed`	boolean	Returns `TRUE` if this format is used, `FALSE` otherwise.
`?isWorksheetFuncUsed`	boolean	Returns `TRUE` if worksheet functions are used, `FALSE` otherwise.
`?isColAlignmentMaintained`	boolean	Returns `TRUE` if column alignment is maintained, `FALSE` otherwise.

Optional parameter	Type	Description
`?isRelativeObjPositionMaintained`	boolean	Returns `TRUE` if relative object position is maintained, `FALSE` otherwise.
`?isPageHeaderExported`	boolean	Returns `TRUE` if the page header is exported, `FALSE` otherwise.
`?isPageHeaderSimplified`	boolean	Returns `TRUE` if page header is simplified, `FALSE` otherwise.
`?isShowGroupOutlines`	boolean	Returns `TRUE` if group outlines are shown, `FALSE` otherwise.

3.4.4 Rich Text Format (RTF) parameters

The following parameter can be used to return information on a report exported in rich text format (RTF).

Optional parameter	Type	Description
`?isPageBreakAfterEachReportPage`	boolean	Returns `TRUE` if there is a page break after each report page, `FALSE` otherwise.

3.4.5 Tab separated text (TTX) paramters

The following parameters can be used to format or return information on a report exported in tab separated text (TTX) format.

Optional parameter	Type	Description
?charactersPerInch=<VALUE>	integer	<VALUE> sets the number of characters per inch.
?baseAreaType=<VALUE>	integer, string	<VALUE> sets the export charset types using one of the following values: • 0 = UTF-8 • 1 = UTF=16LE • 2 = UTF16BE **Note:** Other values will result in an exception being thrown.
?insertFormFeedCharacter	boolean	Returns <TRUE> if a custom separator is used between feeds, <FALSE> otherwise.
?minimumLinesPerPage=<VALUE>	integer	<VALUE> is an integer that sets the minimum number of lines per page.
?pageAreaExportType=<VALUE>	integer	<VALUE> sets the page area export type using one of the following values: • 0 = AsInReport • 1 = OncePerReport • 2 = DoNotExport **Note:** Other values will result in an exception being thrown.

3.4.6 XML parameters

The following parameters can be used to return information on a report exported in RTF format.

Optional parameter	Type	Description
?apply/XSLTIndex=1		Applies a XSL transformation to the exported XML using a zero-based index.
?apply/XSLTName=<VALUE>	string	Applies a XSL transformation to the exported XML using the XSLT's name. <VALUE> is a string that contains the name of the XSLT.

3.5 Interactive parameters

If a report contains interactive parameters, they can be modified to filter the data that is returned from an export, row, grand totals, or group request. To modify interactive parameter values for these requests, add the parameters to the request URL using ?<PARAMETERNAME>=<VALUES> .

Note:
Replace <PARAMETERNAME> with the name of the parameter, and replace <VALUES> with the value the parameter is set to.

Interactive parameter values can be modified when you create a new instance of a report. To set parameter values you must GET the report instance creation form, set the parameter values in the form that is returned, and POST the form to create a new instance that uses the parameter values you have specified.

The following values are supported by parameters:

Supported values	Example
Null	MyParameter=null
Edm.Boolean	MyParameter=true
Edm.DateTime	MyTimestamp=datetime'2000-12-12T12:34:56.403'
Edm.Double	MyCost=7.89

Supported values	Example
Edm.String	`MyCountry='Canada'`
Edm.Time	`MyTime=time'PT17H'`
Date	`MyDate=datetime'2000-03-04T00:00'` **Note:** Date values have no corresponding OData primitive type. They are represented using Edm.Datetime. The time portion must be valid, but will not affect the date being represented.

Range parameter values are specified using an open square bracket. Values are separated by commas. To exclude the lower or upper bound, you can indicate this using a parentheis.

Range	Example
Bounded	`MyLunchBreak=[time'PT12H',time'PT13H']`
No lower bound	`MyMorning=[,time'PT12H']`
No upper bound	`MyEvening=[time'PT18H',]`
Exclude lower bound	`MyLunchBreak=(time'PT12H', time'PT13H']`
Exclude upper bound	`MyLunchBreak=[time'PT12H', time'PT13H')`

Multiple parameter values are specified by a sequence separated by commas. The values in the sequence can be scalar, or range values, or a mixture of the two.

```
MyVacationDays=dateTime'2011-03-04T00:00',dateTime'2011-03-07T00:00',[dateTime'2011-03-17T00:00',dateTime'2011-03-21T00:00']
```

Example: GET http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows?<PARAMETERNAME>=<VALUES>

Make a GET request to retrieve rows filtered by the specified parameter.

Request:

- Method: `GET`
- URL: `http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows?Contact_title="Mr.'"`
- Request header attributes: `X-SAP-LogonToken`
- Request body: none

Response:

- An entry is returned that contains rows filtered by the parameter set in the request.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
```

```
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
  <title type="text">Rows</title>
  <id>http://localhost:6405/biprws/infostore/7422/rpt/data.svc/Rows?Contact_Title='Mr.'</id>
  <link href="Rows?Contact_Title='Mr.'" rel="self" title="Rows"></link>
  <entry>
    <id>http://localhost:6405/biprws/infostore/7422/rpt/data.svc/Rows(0)?Contact_Title='Mr.'</id>
    <title type="text">0</title>
    <author><name></name></author>
    <updated>2011-04-26T17:23:06.000</updated>
    <link href="Rows(0)?Contact_Title='Mr.'" rel="self" title="Row"></link>
    <category term="InteractiveKKNG.Row"
      scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"></category>
    <content type="application/xml">
      <m:properties>
        <d:ID>0</d:ID>
        <d:Contact_First_Name>Markus</d:Contact_First_Name>
        <d:Contact_Title>Mr.</d:Contact_Title>
        <d:Country>USA</d:Country>
        <d:Last_Year_s_Sales>12013.955</d:Last_Year_s_Sales>
      </m:properties>
    </content>
  </entry>
</feed>
```


Related Topics

- Exporting
- Rows
- Report instances
- Grand totals
- Groups

3.6 OData Protocol

The Open Data Protocol (OData) is used to query and update report data. OData is a set of extensions to AtomPub that allows users to discover, navigate, and retrieve data using the programming language of your choice, or a web browser.

3.6.1 Accessing the OData service document

Returns the service document at the root of the OData service.

URI	Operations	Supported format	Is AtomPub
`<baseURI>/infostore/<reportID>/rpt/data.svc`	`GET`	XML, JSON	No

Example: GET http://<baseURI>/infostore/<reportID>/rpt/data.svc

Make a GET request to retrieve the OData service document.

Request:

- Method: `GET`
- URL: `http://<baseURI>/infostore/<reportID>/rpt/data.svc`
- Request header attributes: `X-SAP-LogonToken`
- Request body: none

Response:

- An entry that contains information provided by the root of the OData service.

```
<?xml version="1.0" encoding="utf-8"?>
<app:service xmlns:app="http://www.w3.org/2007/app"
    xmlns="http://www.w3.org/2007/app"
    xmlns:atom="http://www.w3.org/2005/Atom"
    xml:base="http://localhost:6405/biprws/infostore/6311/rpt/data.svc">
  <workspace xmlns="http://www.w3.org/2007/app">
    <atom:title>Default</atom:title>
    <collection href="Rows">
      <atom:title>Rows</atom:title>
    </collection>
    <collection href="GrandTotals">
      <atom:title>GrandTotals</atom:title>
    </collection>
  </workspace>
</app:service>
```

3.6.2 Grand totals

Returns the grand totals collection in XML or JSON format. The grand totals collection includes all summaries available in the report header and footer.

Note:
Running totals are not included in the collection.

URI	Operations	Supported format	Is AtomPub
`<baseURI>/infostore/<reportID>/rpt/data.svc/GrandTotals`	`GET`	XML, JSON	Yes

Optional parameters	Type	Description
`GrandTotals(0)/<FIELDNAME>`	string	Returns the field value associated with a grand total.`<FIELDNAME>` is the a string that specifies the field that you want to retrieve.
`GrandTotals(0)/<FIELDNAME>/$value`	string	Returns only the value associated with a grand total.`<FIELDNAME>` is the a string that specifies the field that you want to retrieve.

Example: GET http://<baseURI>/infostore/<reportID>/rpt/data.svc/GrandTotals

Make a GET request to retrieve the grand totals in a report.

Request:

- Method: `GET`
- URL: `http://<baseURI>/infostore/<reportID>/rpt/data.svc/GrandTotals`
- Request header attributes: `X-SAP-LogonToken`
- Request body: none

Response:

- An that contains the grand totals of the report is returned.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
  <title type="text">GrandTotals</title>
  <id>http://localhost:6405/biprws/infostore/6141/rpt/data.svc/GrandTotals</id>
  <link href="GrandTotals" rel="self" title="GrandTotals"></link>
  <entry>
    <id>http://localhost:6405/biprws/infostore/6141/rpt/data.svc/GrandTotals(0)</id>
    <title type="text">0</title>
    <author><name></name></author>
    <updated>2011-01-28T07:58:55.000</updated>
    <link href="GrandTotals(0)" rel="self" title="GrandTotal"></link>
    <category term="SummariesALLSectionsNG_rpt.GrandTotal"
      scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"></category>
    <content type="application/xml">
      <m:properties>
        <d:ID>0</d:ID>
        <d:Count_Dealer_Price>331</d:Count_Dealer_Price>
        <d:DistinctCount_List_Price>121</d:DistinctCount_List_Price>
      </m:properties>
    </content>
  </entry>
</feed>
```

3.6.3 Groups

Returns group path, group name, and links to group contents.

Note:
Hierarchical groups are not supported.

URI	Operations	Supported format	Is AtomPub
`<baseURI>/infostore/<reportID>/rpt/data.svc/<GroupName>`	`GET`	XML, JSON	Yes

Optional parameters	Type	Description
`<GroupName>('<GroupPath>')`	integer	Returns a linked list containing the data associated with the specified group. `<GroupName>` is the name of the top level group. `<GroupPath>` is a string that represents the index of the subgroup to retrieve. Group paths are zero-based. **Note:** To navigate through subgroups, add the subgroup name and path to the URL. For example: `http://<baseURI>/infostore/<reportID>/rpt/data.svc/Country('0')/Region('0-2')/City('0-2-4')`
`<GroupName1>('<GroupPath>')/$links/<GroupName2>`	integer	Returns a linked list containing the data associated with group specified by `<GroupName2>`. `<GroupName1>` is the name of the top level group. `<GroupPath>` is a string that represents the index of the top level group. Group paths are zero-based. The following example returns a linked list of all cities associated with Country('0'): `http://<baseURI>/infostore/<reportID>/rpt/data.svc/Country('0')/$links/City`
`<GroupName>?$select=<PROPERTY>`	string	Returns row data containing only the properties specified by `<PROPERTY>`. The following values can be used: • * = return all properties • `<PROPERTY>` = the name of the property to return. • `<PROPERTY1>,<PROPERTY2>...` = the names of the properties to return. Multiple properties are separated by commas. **Note:** Case sensitive.
`<GroupName>?$select=<CATEGORY>,<PROPERTY>...`	string	

Optional parameters	Type	Description
		Returns group data containing only the properties specified by `<PROPERTY>`, and only links to the category type specified by `<CATEGORY>`. Multiple properties are separated by commas. **Note:** Case sensitive.

3.6.3.1 To navigate groups and subgroups

1. Create a new HTTP request.
2. Use the GET method to send a request to retrieve a list of links to the contents of the top-level group. For example::
   - In this example, the top level group is `Customer_Country`.
   - Method: `GET`
   - URL: `<baseURI>/infostore/<reportID>/rpt/data.svc/Customer_Country`
   - Request header attributes: `X-SAP-LogonToken`
   - Request body: none

   Example response:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
  <title type="text">Customer_Country</title>
  <id>http://localhost:6405/biprws/infostore/6246/rpt/data.svc/Customer_Country</id>
  <link href="Customer_Country" rel="self" title="Customer_Country"></link>
  <entry>
    <id>http://localhost:6405/biprws/infostore/6246/rpt/data.svc/Customer_Country('0')</id>
    <title type="text">0: Canada</title>
    <author><name></name></author>
    <updated>2010-11-09T18:53:28.000</updated>
    <link href="Customer_Country('0')" rel="self" title="Customer_Country"></link>
    <category term="ng_groupParams_rpt.Customer_Country"
      scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"></category>
    <link href="Customer_Country('0')/Customer_Region"
      rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Customer_Region"
      type="application/atom+xml;type=feed" title="Customer_Region">
    </link>
    <link href="Customer_Country('0')/Customer_City"
      rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Customer_City"
      type="application/atom+xml;type=feed" title="Customer_City">
    </link>
    <content type="application/xml">
      <m:properties>
        <d:ID>0</d:ID>
        <d:Name>Canada</d:Name>
        <d:Sum_of_Amount_of_Delivered_Items_2>377.0</d:Sum_of_Amount_of_Delivered_Items_2>
      </m:properties>
    </content>
  </entry>
  <entry>
    <id>http://localhost:6405/biprws/infostore/6246/rpt/data.svc/Customer_Country('1')</id>
    <title type="text">1: England</title>
```

```
    <author><name></name></author>
    <updated>2010-11-09T18:53:28.000</updated>
    <link href="Customer_Country('1')" rel="self" title="Customer_Country"></link>
    <category term="ng_groupParams_rpt.Customer_Country"
      scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"></category>
    <link href="Customer_Country('1')/Customer_Region"
      rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Customer_Region"
      type="application/atom+xml;type=feed" title="Customer_Region">
    </link>
    <link href="Customer_Country('1')/Customer_City"
      rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Customer_City"
      type="application/atom+xml;type=feed" title="Customer_City">
    </link>
    <content type="application/xml">
      <m:properties>
        <d:ID>1</d:ID>
        <d:Name>England</d:Name>
        <d:Sum_of_Amount_of_Delivered_Items_2>1553.0</d:Sum_of_Amount_of_Delivered_Items_2>
      </m:properties>
    </content>
  </entry>
</feed>
```

3. Use the GET method to send a request to a link retrieved in the previous step.

   Example request:

   - Retrieve regions associated with `Customer_Country('0')`.
   - Method: `GET`
   - URL: `<baseURI>/infostore/<reportID>/rpt/data.svc/Customer_Country('0')/Customer_Region`
   - Request header attributes: `X-SAP-LogonToken`
   - Request body: none

   Example response:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
  <title type="text">Customer_Region</title>
  <id>http://localhost:6405/biprws/infostore/6246/rpt/data.svc/Customer_Country('0')/Customer_Region</id>

  <link href="Customer_Country('0')/Customer_Region" rel="self" title="Customer_Region"></link>
  <entry>
    <id>http://localhost:6405/biprws/infostore/6246/rpt/data.svc/Customer_Region('0-0')</id>
    <title type="text">0-0: 130</title>
    <author><name></name></author>
    <updated>2010-11-09T18:53:28.000</updated>
    <link href="Customer_Region('0-0')" rel="self" title="Customer_Region"></link>
    <category term="ng_groupParams_rpt.Customer_Region"
      scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"></category>
    <link href="Customer_Region('0-0')/Customer_City"
      rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Customer_City"
      type="application/atom+xml;type=feed" title="Customer_City">
    </link>
    <content type="application/xml">
      <m:properties>
        <d:ID>0-0</d:ID>
        <d:Name>130</d:Name>
        <d:Sum_of_Amount_of_Delivered_Items_3>377.0</d:Sum_of_Amount_of_Delivered_Items_3>
      </m:properties>
    </content>
  </entry>
</feed>
```

4. Repeat steps 2 to 3 to drill down through group data.

3.6.4 Metadata

Returns an EDMX document that contains a complete description of the feeds, types, properties, and collections exposed by the OData service.

URI	Operations	Supported format	Is AtomPub
`<baseURI>/infostore/<reportID>/rpt/data.svc/$metadata`	`GET`	XML	Yes

Example: GET http://<baseURI>/infostore/<reportID>/rpt/data.svc/$metadata

Make a GET request to retrieve report metadata.

Request:

- Method: `GET`
- URL: `http://<baseURI>/infostore/<reportID>/rpt/data.svc/$metadata`
- Request header attributes: `X-SAP-LogonToken`
- Request body: none

Response:

- An EDMX document containing report metadata is returned.

```
<?xml version="1.0" encoding="utf-8"?>
<edmx:Edmx xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx" Version="1.0">
  <edmx:DataServices xmlns:m="http://schemas.microsoft.com/ado/2007/08/
      dataservices/metadata" m:DataServiceVersion="1.0">
    <Schema xmlns="http://schemas.microsoft.com/ado/2006/04/edm"
        xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
        xmlns:crds="http://schemas.sap.com/crystalreports/crodataservice"
        Namespace="DateTime_rpt">
      <EntityContainer Name="DateTime_rpt_Container" m:IsDefaultEntityContainer="true">
        <EntitySet Name="Rows" EntityType="DateTime_rpt.Row" crds:IsGroup="false">
        </EntitySet>
        <EntitySet Name="GrandTotals" EntityType="DateTime_rpt.GrandTotal" crds:IsGroup="false">
        </EntitySet>
      </EntityContainer>
      <EntityType Name="Row">
        <Key>
          <PropertyRef Name="ID">
          </PropertyRef>
        </Key>
        <Property Name="ID" Type="Edm.Int32">
        </Property>
        <Property Name="Customer_City" Type="Edm.String"
          crds:FormulaForm="{SAPBW.Customer\City}" MaxLength="65534">
        </Property>
      </EntityType>
      <EntityType Name="GrandTotal">
        <Key>
          <PropertyRef Name="ID">
          </PropertyRef>
        </Key>
        <Property Name="ID" Type="Edm.Int32">
        </Property>
      </EntityType>
    </Schema>
```

```
  </edmx:DataServices>
</edmx:Edmx>
```

3.6.5 Rows

Returns or updates data that corresponds to row data in the Details area of a report. Rows are represented by an Atom Feed or an array of JSON objects.

When you POST a row to a report, you can also use both XML or JSON to specify the new row information. The format of the request body is specified using the `Content-Type` attribute in the message header.

Note the following:

- You can POST database field data only to a report, not formulas.
- You cannot POST to delegated fields.
- You cannot POST to reports with hierarchical groups.
- You cannot POST to OLAP member fields.
- You can POST row data only to a transient instance of a report.

Note:
Row IDs are automatically generated.

URI	Operations	Supported format	Is AtomPub
`<baseURI>/infostore/<reportID>/rpt/data.svc/Rows`	`GET`, `POST`	XML, JSON	Yes

Optional parameters	Type	Description
`Rows(<INDEX>)`	integer	Returns a row of the report. `<INDEX>` is an integer which specifies what row to fetch information from.
`Rows(<INDEX>)/<FIELDNAME>`	integer	Returns information about a field in a report. `<INDEX>` is an integer that specifies what row to fetch information from. `<FIELDNAME>` is the name of the field value to return.
`Rows(<INDEX>)/<FIELDNAME>/$value`	integer	Returns the raw field value of a particular row index `<INDEX>` is an integer that specifies what row to fetch information from. `<FIELDNAME>`is the name of the field value to return.
`Rows?$select=<PROPERTY>`	string	Returns row data containing only the properties specified by `<PROPERTY>`. The following values can be used: • * = return all properties • `<PROPERTY>` = the name of the property to return. • `<PROPERTY1>,<PROPERTY2>...` = the names of the properties to return. Multiple properties are separated by commas. **Note:** Case sensitive.
`<GroupName>?$select=<CATEGORY>,<PROPERTY>...`	string	Returns group data containing only the properties specified by `<PROPERTY>`, and only links to the category type specified by `<CATEGORY>`. Multiple properties are separated by commas. **Note:** Case sensitive.
`Rows?$skip=<VALUE>`	integer	Returns the row data in chunks, starting with the record number specified. `<VALUE>` is an integer that indicates the first record to return.

Optional parameters	Type	Description
`Rows?$skiptoken=<VALUE>`	integer	Returns the row data in chunks, starting with the record number specified. `<VALUE>` is an integer that indicates the first record to return.
`Rows?$inlinecount=allpages`		Adds a count entry to the xml that specifies the count of all records.
`Rows/$count`		Return the count of records.

Example: GET http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows(0)

Use the GET method to request the first row of a report.

Request:

- Method: `GET`
- URL: `http://<baseURI>/infostore/<reportID>/rpt/data.svc/Rows(0)`
- Required request header attributes: `X-SAP-LogonToken`
- Optional request header attributes: `Accept`
- Request body: none

Response:

- An entry is returned that contains information about the second row of the report.

```
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
  <id>http://localhost:6405/biprws/infostore/6188/rpt/data.svc/Rows(1)</id>
  <title type="text">1</title>
  <author><name></name></author>
  <updated>2010-12-09T20:10:04.000</updated>
  <link href="Rows(0)" rel="self" title="Row"></link>
  <category term="EmployeeData.Row" scheme="http://schemas.microsoft.com/ado/2007/08/dataser
vices/scheme"></category>
  <content type="application/xml">
    <m:properties>
      <d:ID>1</d:ID>
      <d:RTotal_max_emergency_last_name>Buchanan</d:RTotal_max_emergency_last_name>
      <d:RTotal_sum_salary>110000.0</d:RTotal_sum_salary>
      <d:Employee_Birth_Date>1960-03-13T00:00:00.000</d:Employee_Birth_Date>
      <d:Employee_Emergency_Contact_Last_Name>Hellstern</d:Employee_Emergency_Contact_Last_Name>
      <d:Employee_Employee_Id>10.0</d:Employee_Employee_Id>
      <d:Employee_Extension>7559</d:Employee_Extension>
      <d:Employee_First_Name>Albert</d:Employee_First_Name>
      <d:Employee_Last_Name>Hellstern</d:Employee_Last_Name>
      <d:Employee_Position>Business Manager</d:Employee_Position>
      <d:Employee_Reports_To>2.0</d:Employee_Reports_To>
      <d:Employee_Salary>60000.0</d:Employee_Salary>
      <d:Employee_Supervisor_Id>2.0</d:Employee_Supervisor_Id>
      <d:Employee_Home_Phone>(206)555-4869</d:Employee_Home_Phone>
    </m:properties>
  </content>
</entry>
```

Example: POST http://<baseURI>/infostore/<reportID>/rpt/<instanceID>/data.svc/Rows

POST a row to an instance of a report.

Request:

- Method: `POST`
- URL: `http://<baseURI>/<reportID>/rpt/<instanceID>/data.svc/Rows`
- Request header attributes:

```
accept
accept-charset: UTF-8
content-type: application/atom+xml
host: <HostName>
content-length: 1000
X-SAP-LogonToken: COMMANDCOM-LCM:6400@{3&2=5328,U3&p=40676
  .8926203819,Y7&4F=12,U3&63=secEnterprise,0P&66=60,03&68=secEnterprise:
  Administrator,0P&qe=100,U3&vz=IVD21LbMCB0eRiI4atz9sNL18Ux5anRBdYB9fFv5NrY,UP}
```

- Request body: Add row information.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<entry xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns="http://www.w3.org/2005/Atom">
<content type="application/xml">
<m:properties>
<d:Orders_Order_Date m:type="Edm.DateTime">2011-01-28T00:00:00.000</d:Orders_Order_Date>
</m:properties>
</content>
</entry>
```

Note:
`term="Xtreme_OrderDate.Row"` is the name of the row entity in the report. You can find the names of the entities in the report by getting the report metadata.

Response:

- An entry is returned that contains the row you have added to the report.

```
<?xml version='1.0' encoding='UTF-8'?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
 xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
  <id>http://localhost:6405/biprws/1667/rpt/data.svc/Rows(523)</id>
  <title type="text">523</title>
  <category term="CRoDataService.Row" scheme="http://schemas.microsoft.com/ado/2007/08/dataser
vices/scheme">
  </category>
  <content type="application/xml">
    <m:properties>
      <d:ID>523</d:ID>
      <d:Orders_Order_Date>2011-01-28T00:00:00.000</d:Orders_Order_Date>
    </m:properties>
  </content>
</entry>
```

More Information

Information Resource	Location
SAP BusinessObjects product information	http://www.sap.com
SAP Help Portal	Navigate to http://help.sap.com/businessobjects and on the "SAP BusinessObjects Overview" side panel click **All Products**. You can access the most up-to-date documentation covering all SAP BusinessObjects products and their deployment at the SAP Help Portal. You can download PDF versions or installable HTML libraries. Certain guides are stored on the SAP Service Marketplace and are not available from the SAP Help Portal. These guides are listed on the Help Portal accompanied by a link to the SAP Service Marketplace. Customers with a maintenance agreement have an authorized user ID to access this site. To obtain an ID, contact your customer support representative.
SAP Service Marketplace	http://service.sap.com/bosap-support > Documentation • Installation guides: https://service.sap.com/bosap-instguides • Release notes: http://service.sap.com/releasenotes The SAP Service Marketplace stores certain installation guides, upgrade and migration guides, deployment guides, release notes and Supported Platforms documents. Customers with a maintenance agreement have an authorized user ID to access this site. Contact your customer support representative to obtain an ID. If you are redirected to the SAP Service Marketplace from the SAP Help Portal, use the menu in the navigation pane on the left to locate the category containing the documentation you want to access.
Docupedia	https://cw.sdn.sap.com/cw/community/docupedia Docupedia provides additional documentation resources, a collaborative authoring environment, and an interactive feedback channel.
Developer resources	https://boc.sdn.sap.com/ https://www.sdn.sap.com/irj/sdn/businessobjects-sdklibrary

Information Resource	Location
SAP BusinessObjects articles on the SAP Community Network	https://www.sdn.sap.com/irj/boc/businessobjects-articles These articles were formerly known as technical papers.
Notes	https://service.sap.com/notes These notes were formerly known as Knowledge Base articles.
Forums on the SAP Community Network	https://www.sdn.sap.com/irj/scn/forums
Training	http://www.sap.com/services/education From traditional classroom learning to targeted e-learning seminars, we can offer a training package to suit your learning needs and preferred learning style.
Online customer support	http://service.sap.com/bosap-support The SAP Support Portal contains information about Customer Support programs and services. It also has links to a wide range of technical information and downloads. Customers with a maintenance agreement have an authorized user ID to access this site. To obtain an ID, contact your customer support representative.
Consulting	http://www.sap.com/services/bysubject/businessobjectsconsulting Consultants can accompany you from the initial analysis stage to the delivery of your deployment project. Expertise is available in topics such as relational and multidimensional databases, connectivity, database design tools, and customized embedding technology.

Index

{ 20
/logon/adsso 36
/logon/long 32
/logon/trusted 37

A

Accept-Language 29
authentication 31

B

base URL 30
    retrieving 30
BI platform .NET SDK 43
BI platform Java SDK 43
BI platform Web Services SDK 44

C

CMC 30
Common URIs 47
concurrent user license 34
Content-Type 14

D

data.svc 61
DefaultToken property 43
deferred content 20

E

Enterprise authentication 32, 36, 40
entries 20
entry properties 20
error_code 20
export 52
    character separated value (CSV) 53
    Excel 54
    Excel Data-Only 55
    Rich Text Format (RTF) 57
    tab separated text (TTX) 57
    XML 58

G

getDefaultToken 43, 44
getSerializedSession 43, 44
grand totals 62
groups 64
    navigate 66
    subgroup 66

H

HTTP basic authentication 40
HTTP error code 14

I

introduction 5
IRestWebService 30

J

JSON escape characters 20
JSON format example 7

L

LDAP authentication 32, 40
logging on 31
    with serialized session 34
    with session token 34
    with user name and password 32
logoff 41
logon token 31, 39
    converting XML-encoded characters 39
    invalidating 41

M

metadata 20, 68
multilingual data 29

O

OData 61
    accessing the service 61
OpenDocument 16

P

parameter values 59

R

report ID 44
    retrieve programatically 45
report instance 49
    create 50
report URI 48
request body 7, 14
request header 7, 12, 31
response body 7
response header 7
REST 47
rows 69

S

SAP authentication 32, 40
secEnterprise 32, 40
secLDAP 32, 40
secSAPR3 32, 40
secWinAD 32, 40
serialized session 34, 42
SerializedSession property 43
session token 34, 42
SI_ACCESS_URL 30

T

transient instance 49
Trusted Authentication 37

W

WinAD authentication 32, 40
WinAD SSO authentication 36, 37

X

X-SAP-LogonToken 14, 31, 39
X-SAP-PVL 29
XML character encoding 39