Business Intelligence Platform RESTful Web Service Developer Guide

■ SAP BusinessObjects Business Intelligence platform 4.1

2013-07-15

2013-07-15

Contents

Document History

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

Version	Date	Description
SAP BusinessObjects Business Intelligence platform 4.1	May, 2013	First release of this document.

Getting Started

The Business Intelligence platform RESTful web service SDK lets you access the BI platform using the HTTP protocol. You can use this SDK to log on to the BI platform, navigate the BI platform repository, access resources, and perform basic resource scheduling. You can access this SDK by writing applications that use any programming language that supports the HTTP protocol, or by using any tool that supports making HTTP requests. Both XML and JSON (JavaScript Object Notation) request and response formats are supported. For more information on the JSON format, see `www.json.org` and `tools.ietf.org/html/rfc4627`. For more information on the XML format, see `www.w3.org/XML/`

When to use this SDK

Use the RESTful web services SDK under the following conditions:

- You want to access BI platform repository objects or perform basic scheduling.
- You want to use a programming language that is not supported by other BI platform SDKs.
- You do not want to download and install BI platform libraries as a part of your application.

If you want to programmatically access the advanced functionality of the BI platform, including server administration, security configuration, and modifying the repository, use one of the BI platform SDKs that support these features. For example, use the SAP BusinessObjects Business Intelligence platform Java SDK, the SAP BusinessObjects Business Intelligence platform .NET SDK, or the SAP BusinessObjects Business Intelligence platform Web Services SDK to access the advanced features of the BI platform.

Start using the SDK

This guide is divided into these sections:

- Setting up the development environment - The supported development environments for using the RESTful web services SDK.
- Using the SDK - How to use the RESTful web services SDK, including how to log on to the BI platform, navigate the BI platform repository, schedule and access objects, interpret error messages, and work with multilingual content.
- Administration and installation tasks - How to install and configure RESTful web services on your BI platform deployment. This section is for BI platform system administrators.
- API Reference - A reference for RESTful web service requests.

Setting up the development environment

To develop applications that use the Business Intelligence platform RESTful web service SDK, you must be able to log on to a BI platform deployment that has RESTful web services installed, and know how to make HTTP requests:

- The BI platform deployment must have an instance of the RESTful web service installed and configured. Contact your BI platform administrator if RESTful web services are not installed and configured on your deployment.
- You must look up the base URL and port number that is used to listen for RESTful web service requests. You can find the base URL and port number either programmatically or by viewing it in the Central Management Console (CMC).
- You must be able to log on to the BI platform. You can do this by using a valid user ID and password, or by using a serialized session or session token that you have obtained from another SDK.
- You must know how to make HTTP requests, either by using your preferred programming language or a tool that supports making HTTP requests.

Related Topics

- Administration and installation tasks
- Authentication
- Retrieving the base URL for RESTful web service requests

3.1 Supported programming languages

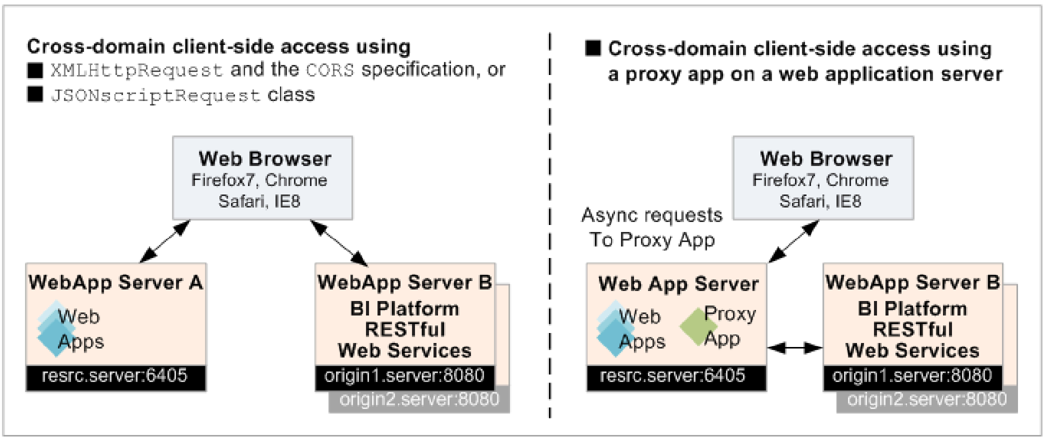
You can access the BI platform RESTful web services using any programming language that supports making HTTP requests. You are not required to include any libraries in your application.

Most advanced programming languages contain support for making HTTP requests. The cURL programming language has excellent support for HTTP requests, and provides cURL-based libraries for most major programming languages. For more information about cURL, see `http://curl.haxx.se/`.

You can also make HTTP requests without writing code by using tools that make HTTP requests. For example, you can obtain a REST Client plugin for Mozilla Firefox that allows you to make RESTful HTTP calls by specifying the URL, method, request header, and request body.

3.2 Using Ajax and JavaScript with RESTful web services across domains

Cross-domain HTTP requests to the BI platform RESTful web services are restricted by a security policy built into the JavaScript and Ajax languages. The intent is to prevent the operation of malicious scripts that may be unintentionally run from an untrusted server. This may include scripts hosted on different domains or from different ports on the same server or scripts hosted on the same server that uses a different protocol, for example `http` instead of `https`. Use one of the following workarounds to enable JavaScript or Ajax applications to make cross-domain requests to RESTful web services.



Using XMLHttpRequest and the CORS specification

When only client-side technologies are used such as HTML, CSS and JavaScript, cross-domain access is achieved by implementing `CORS` (Cross-Origin Resource Sharing) on the RESTful web server and the client-side web browser using the `XMLHttpRequest`. For more information on CORS, see `http://www.w3.org/TR/cors/`.

To restrict which domains may be accessed from the browser using CORS, the RESTful web server must be configured to include those domains.

Because various web browsers implement the CORS specification differently, use a library that allows the use of a single interface that works for all browsers and versions you intend to support.

Note:
All of the Business Intelligence platform RESTful web service requests return results in XML or JSON format, so `XMLHttpRequest` may be used to process both response types. The JSONscriptRequest class is not restricted by the cross-origin requests.

Using a proxy app on a web application server

A proxy web application that runs on the same server as the JavaScript web page is used to forward HTTP requests for resources that exist on another server.

Proxies are used on websites that use server-side technologies such as JSP, Java Servlets, C# and ASP.NET. Web pages that use JavaScript and Ajax programming can make calls to other servers using a proxy application written in a programming language that does not have a same-origin security policy.

For security purposes, you can set up the proxy with suitable access restrictions to avoid unauthorized access to any internal or external networks. For example, if the required resources exist on domains `http://origin1.server:8080` and `http://origin2.server:8080`, the pass-through on the proxy server must only forward requests to only those addresses.

Related Topics

- To configure cross-origin resource sharing (CORS)

Using the SDK

This section describes how to use the Business Intelligence platform RESTful web service SDK, including how to find the RESTful web services base URL, how to log on to the BI platform, and how to navigate the BI platform repository.

4.1 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).

4.1.1 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.

4.1.2 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*.

4.2 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-LogonTo-ken	"COMMANDCOM-LCM:6400@{3&2=5542,U3&p=40680.8979564815,Y7&4F=12,U3&63=secEn-terprise,0P&66=60,03&68=secEnterprise:Administra-tor,0P&qe=100,U3&vz=TZnoIE2yQyeLCkAlnHtaaYUHon5.p0yTk-SaUiLC8SSM,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
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 `BOEPass word999` 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":"Ac1aKZlzj5VJmMQi5LDa53s",
       "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

4.2.1 Creating the request header

The request header of an HTTP request contains a set of attributes that describe the request. The BI platform RESTful web service SDK recognizes a set of standard HTTP attributes, as well as custom attributes defined specifically for the BI platform.

Note:
The BI platform passes requests to other layers of the system, including client applications. You can include request header attributes that are not recognized by the BI platform but are recognized by client applications.

The following table describes request headers that are recognized by the BI platform:

Attribute	Description	Sample Value
`Content-Type`	The format of the request body. The BI platform accepts content of type `application/xml` or `application/json`. Client applications may accept other formats.	application/xml
`Accept`	The expected format of the response body. The BI platform provides content in the `application/xml` or in `application/json` format. Client applications may provide content in other formats.	application/json
`Accept-Language`	The preferred language used to retrieve system and error messages. This corresponds to the Product Locale (PL) of the BI platform.	en-US
`X-SAP-PVL`	The preferred language used to retrieve BI platform content. This corresponds to the Preferred Viewing Language (PVL).	ja-JP

Attribute	Description	Sample Value
`X-SAP-LogonToken`	A logon token received from the authentication process. Enclose the logon token in quotation marks.	`"COMMANDCOM-LCM:6400@{3&2=55,3&p=403.0083,Y7&4F=12,U3&63=secEnterprise,0P&56=60,03&68=secEnterprise:administrator,0P&qe=100,U3&vz=y3EqvsvoVMU8raN2YjqDe4,UP}"`
`Authorization`	The authorization type to use, for example HTTP basic authentication.	`Basic <authtype>\<username>:<password>` Replace `<authtype>` with the authentication type, `<username>` with your user name and `<password>` with the password.
`X-SAP-TRUSTED-USER`	The account name of a trusted user. The label `X-SAP-TRUSTED-USER` may be changed in CMC, **Servers List > WACS**, "Trusted Authentication Configuration" to another label such as `MyUser`.	trustedUser

Related Topics

- Authentication

4.2.2 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.

4.2.3 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.

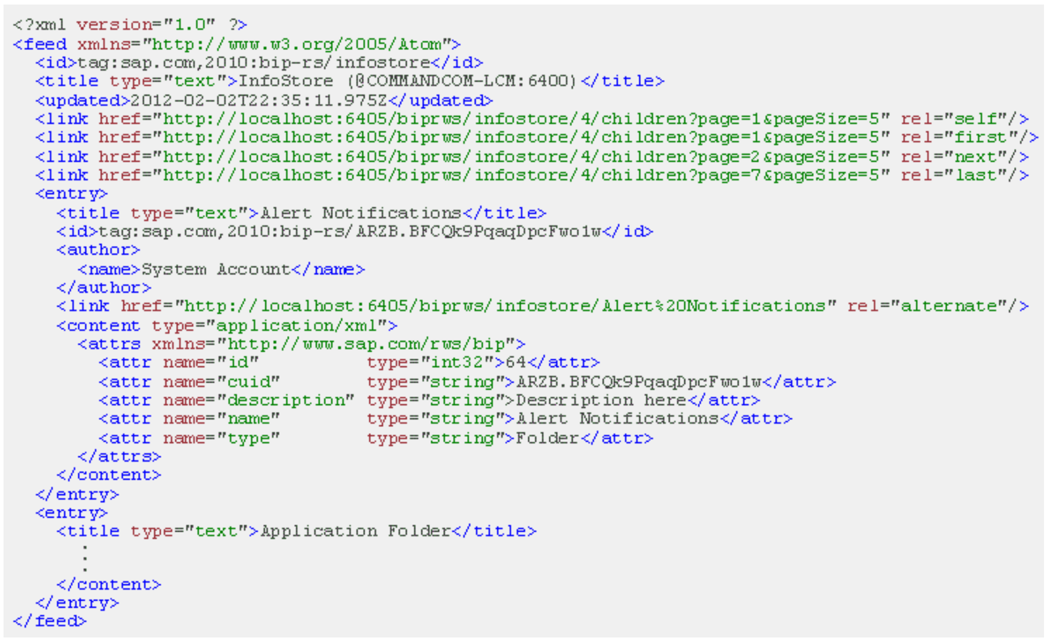
Note:
A copy of the `X-SAP-LogonToken` value is returned in the response body of authentication responses. However, the response body is formatted as XML and converts the logon token to an XML-encoded version. This copy of the logon token must be converted back to its original format before it can be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

Related Topics

• Converting a logon token from XML-encoded text

4.2.4 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.

4.2.5 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.

At-tribute	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=AQtkbb SqN4NOj3ydf.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

4.2.6 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 4-1: 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 { }.

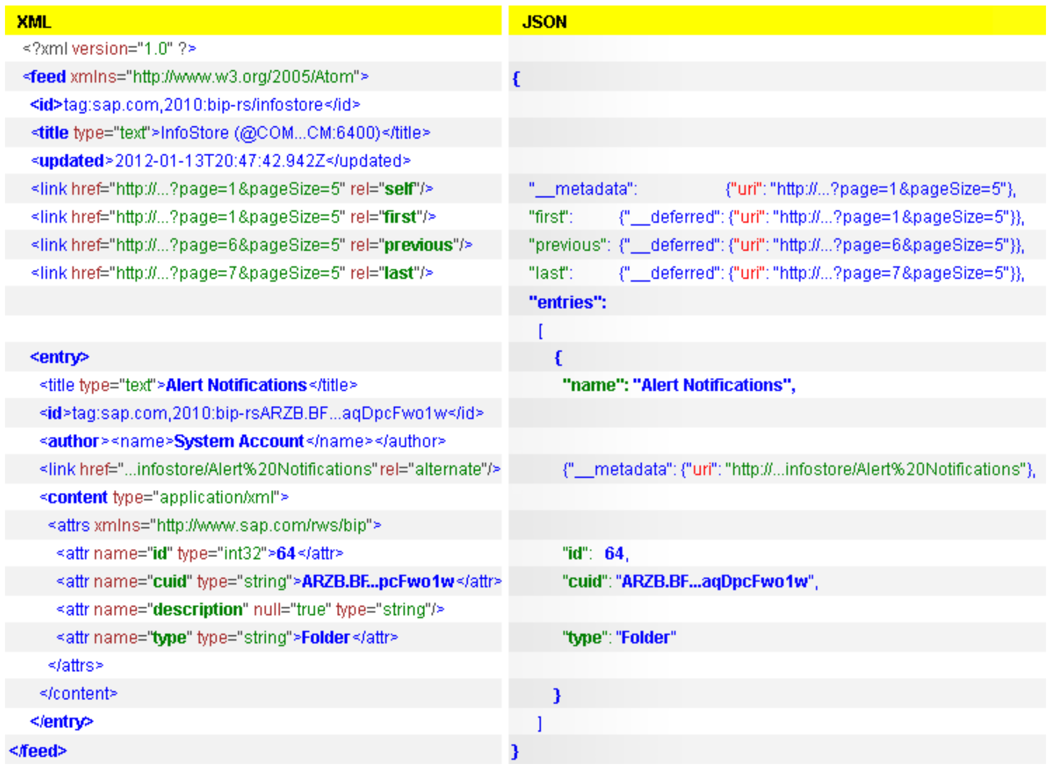
XML			JSON		Description
XML Tag	**Sample**	**Type**	**Value**	**Type**	
<entry>			entries : [{contentsOfEntryItem#1}, {contentsOfEntryItem#2}]		A request for a list of children, a collection of entries is returned, each one a JSON object. The collection of JSON objects is represented as an array in the "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>`	`}`

4.2.7 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.

4.3 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

4.3.1 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</attr>
   </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 `&amp;`. 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
- /logon/long
- Converting a logon token from XML-encoded text

4.3.2 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_131NlLLf5bRM
   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 `&amp;`.

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 `&amp;`. To use a logon token taken from the response body, you must convert the XML-encoded logon token back to its original format.

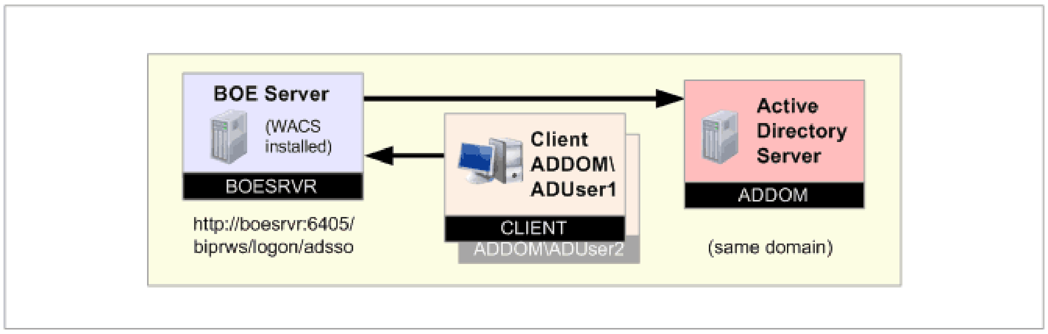
The following example shows how the XML-encoded logon token appears in the response body:

```
<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
- /logon/token
- Converting a logon token from XML-encoded text

4.3.3 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
- Using logon tokens with OpenDocument URLs
- Converting a logon token from XML-encoded text
- To configure web.xml to enable WinAD SSO

4.3.4 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
- Using logon tokens with OpenDocument URLs
- Converting a logon token from XML-encoded text
- To enable and configure trusted authentication

4.3.5 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 `&amp;` character encoding with the `&` character.

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

XML encoding	Character
`&apos;`	`'`
`&quot;`	`"`
`&amp;`	`&`
`&lt;`	`<`
`&gt;`	`>`

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}
```

4.3.6 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

4.3.7 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>
```

4.3.8 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
- /logoff

4.3.9 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

Related Topics

- /logon/token

4.3.9.1 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 `ILo gonTokenMgr` 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*.

4.3.9.2 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;
```

4.3.9.3 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*.

4.4 Using logon tokens with OpenDocument URLs

OpenDocument syntax allows you to create hyperlinks that directly link to documents stored in the BI platform. The Business Intelligence platform RESTful web services SDK provides some support for working with OpenDocument. Logon tokens obtained from the Business Intelligence platform RESTful web services SDK can be used to authenticate with OpenDocument, and some RESTful responses return OpenDocument links.

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

Obtaining OpenDocument links for documents

When you request a document, for example a Crystal report or a WebI report, the response includes a OpenDocument link that can be used to view the resource with OpenDocument.

Links to OpenDocument URLs can be identified by the `rel` attribute, `"http://www.sap.com/rws/bip#opendocument"`, and the `title` attribute, `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”>
```

Appending the logon token to the OpenDocument URL

You can authenticate an OpenDocument URL by appending a logon token obtained using the Business Intelligence platform RESTful web services SDK to the end of the URL.

The syntax of the logon token parameter is shown below. Replace `<openDocumentURL>` with the OpenDocument URL and replace `<logonToken>` with the URL-encoded logon token value.

```
<openDocumentURL>&token=<logonToken>
```

Note:
A URL-encoded logon token may contain a large number of characters. Some web browsers may limit the number of characters that are allowed in a URL.

The following example shows how to add a logon token to the end of the OpenDocument URL, `http://localhost:8080/BOE/OpenDocument/opendoc/openDocument.jsp?sIDType=CUID&iDocID=AYymBvuJZTRAlkojmuUj36w`.

1. Get a logon token by authenticating with the BI platform RESTful web services SDK.

```
COMMANDCOM-LCM:6400@{3&2=5521,U3&p=40709.9614065046,Y7&4F=12,U3&63=secEnterprise,0P&66=60,03&68=secEnter
prise:Administrator,0P&qe=100,U3&vz=sIQcJghbp2BvJrgPBNGJrRruBpfSShro9.ipdnKzqXM,UP}
```

2. URL-encode the logon token.

```
COMMANDCOM-LCM%3A6400%40%7B3%262%3D5521%2CU3%26p%3D40709.9614065046%2CY7%264F%3D12%2CU3%2663%3DsecEnter
prise%2C0P%2666%3D60%2C03%2668%3DsecEnterprise%3AAdministrator%2C0P%26qe%3D100%2CU3%26vz%3DsIQcJghbp2BvJrgPB
NGJrRruBpfSShro9.ipdnKzqXM%2CUP%7D
```

Note:
There are many free tools available that can URL-encode strings.

3. Append `&token=<logonToken>` to the end of the OpenDocument URL. Replace `<logonToken>` with the URL-encoded logon token.

```
http://localhost:8080/BOE/OpenDocument/opendoc/openDocument.jsp?sIDType=CUID&iDocID=AYymBvuJZTRAlkoj
muUj36w&token=COMMANDCOM-
LCM%3A6400%40%7B3%262%3D5521%2CU3%26p%3D40709.9614065046%2CY7%264F%3D12%2CU3%2663%3DsecEnter
prise%2C0P%2666%3D60%2C03%2668%3DsecEnterprise%3AAdministrator%2C0P%26qe%3D100%2CU3%26vz%3DsIQcJghbp2BvJrgPB
NGJrRruBpfSShro9.ipdnKzqXM%2CUP%7D
```

Adding the logon token to the OpenDocument request header

You can add the `X-SAP-LogonToken` attribute to the HTTP request header of an OpenDocument request, and set its value to be the value of the logon token. Enclose the logon token in quotation marks. Add the logon token to the request header when you want to avoid URL-encoding the logon token and appending a large number of characters to the end of the OpenDocument URL.

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

- Authentication

4.5 Navigating the BI platform repository

You can navigate through the BI platform repository, also known as the InfoStore, by requesting objects and following the links provided by the responses. Responses contain links to parent folders, child objects, and other related information. For example, when you request a folder, the response contains a link that returns the children of the folder. You can also retrieve objects directly by requesting them by their ID or CUID.

You can limit the number of entries returned by a response by requesting objects of a certain type, or by splitting a large number of entries across multiple pages.

Before you can view the contents of the BI platform repository, you must be authenticated and have obtained a logon token. Pass the logon token in the request header of each request by adding the `X-SAP-LogonToken` attribute to the request header and setting its value to be the logon token.

4.5.1 To view the top level of the BI platform repository

Before you can view the BI platform repository, you must have obtained a valid logon token and know the base URL for RESTful web service requests.

You can make a request to view the top level of the BI platform repository, also known as the InfoStore. The returned result contains links that you can follow to navigate child folders and explore the repository.

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://<baseURL>/infostore/` URL.

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

   ```
   GET http://localhost:6405/biprws/infostore
   ```

4. The response contains a feed that contains links children and entries that describe the top-level folders of the repository.

   Each `<link>` entry contains a hyperlink to a RESTful URL that can be used to access the resource directly. The list of attributes contains properties of the resource.

   ```
   <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>2011-03-31T23:55:10.852Z</updated>
     <link href="http://localhost:6405/biprws/infostore/4/children?page=1&amp;pageSize=50" rel="self"></link>

     <link href="http://localhost:6405/biprws/infostore/4/children?page=1&amp;pageSize=50" rel="first"></link>

     <link href="http://localhost:6405/biprws/infostore/4/children?page=1&amp;pageSize=50" rel="last"></link>

     <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"></link>
       <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" null="true"></attr>
           <attr name="type" type="string">Folder</attr>
         </attrs>
       </content>
     </entry>
     <entry>
       <title type="text">Application Folder</title>
       <id>tag:sap.com,2010:bip-rs/AdoctK9h1sBHp3I6uG0Sh7M</id>
       <author><name>System Account</name></author>
       <link href="http://localhost:6405/biprws/infostore/Application%20Folder" rel="alternate"></link>
       <content type="application/xml">
         <attrs xmlns="http://www.sap.com/rws/bip">
           <attr name="id" type="int32">43</attr>
           <attr name="cuid" type="string">AdoctK9h1sBHp3I6uG0Sh7M</attr>
           <attr name="description" type="string"></attr>
           <attr name="type" type="string">Folder</attr>
         </attrs>
       </content>
     </entry>
   ...
   </feed>
   ```

Related Topics

- Authentication
- /infostore

4.5.2 To retrieve an object by ID

Before you can retrieve a resource from the BI platform, you must have a valid logon token and know the base URL for RESTful web service requests. To retrieve an object by ID, you must know the ID of the resource you are requesting. You can find the ID of a resource by accessing it in the Central Management Console (CMC) and inspecting its properties, or by reading the `id` attribute of the `<attr>` entry returned in a RESTful web service response. The ID attribute corresponds to the `SI_ID` property of the object in the BI platform repository.

You can access a resource directly by using its ID.

1. Create a new HTTP request.
2. Add the `X-SAP-LogonToken` attribute to the request header and set its value to a valid logon token.
3. Add the `Accept` attribute to the request header and set its value to `application/xml`.
4. Use the `GET` method to send a request to the `http://<baseURL>/biprws/infostore/<ID>` URL.

   Replace `<baseURL>` with the base URL for RESTful web service requests, and replace `<ID>` with the ID of the object you want to retrieve.

   ```
   GET http://localhost:6405/biprws/infostore/43
   ```

   The response contains an `<entry>` element that contains an XML description of the resource. This example gets the Application Folder by its ID, 43.

   ```
   <entry xmlns="http://www.w3.org/2005/Atom">
     <author><name>System Account</name></author>
     <id>tag:sap.com,2010:bip-rs/AdoctK9h1sBHp3I6uG0Sh7M</id>
     <title type="text">Application Folder</title>
     <updated>2011-04-14T10:27:50.672Z</updated>
     <link href="http://localhost:6405/biprws/infostore/Application%20Folder/children"
   rel="http://www.sap.com/rws/bip#children"></link>
     <link href="http://localhost:6405/biprws/infostore" rel="up"></link>
     <content type="application/xml">
       <attrs xmlns="http://www.sap.com/rws/bip">
         <attr name="id" type="int32">43</attr>
         <attr name="cuid" type="string">AdoctK9h1sBHp3I6uG0Sh7M</attr>
         <attr name="description" type="string"></attr>
         <attr name="name" type="string">Application Folder</attr>
         <attr name="type" type="string">Folder</attr>
       </attrs>
     </content>
   </entry>
   ```

Related Topics

• /infostore/<id>

4.5.3 To retrieve an object by CUID

Before you can retrieve a resource from the BI platform, you must have a valid logon token and know the base URL for RESTful web service requests. To retrieve an object by CUID, you must know the CUID of the resource you are requesting. You can find the CUID of a resource by accessing it in the Central Management Console (CMC) and inspecting its properties, or by reading the `cuid` attribute of the `<attr>` entry returned in a RESTful web service response. The CUID attribute corresponds to the `SI_CUID` property of the object in the BI platform repository.

You can access a resource directly by using its CUID.

1. Create a new HTTP request.
2. Add the `X-SAP-LogonToken` attribute to the request header and set its value to a valid logon token.
3. Add the `Accept` attribute to the request header and set its value to `application/xml`.
4. Use the `GET` method to send a request to `http://<baseURL>/infostore/cuid_<CUID>`.

   Replace `<baseURL>` with the base URL for RESTful web service requests, and replace `<CUID>` with the CUID of the object you want to retrieve. This example gets the Application Folder by its CUID, `AdoctK9h1sBHp3I6uG0Sh7M`.

   ```
   GET http://localhost:6405/biprws/infostore/cuid_AdoctK9h1sBHp3I6uG0Sh7M
   ```

   The response is an `<entry>` element that contains an XML description of the resource. In this example, the object with CUID = `AdoctK9h1sBHp3I6uG0Sh7M` is returned.

   ```
   <entry xmlns="http://www.w3.org/2005/Atom">
     <author><name>System Account</name></author>
     <id>tag:sap.com,2010:bip-rs/AdoctK9h1sBHp3I6uG0Sh7M</id>
     <title type="text">Application Folder</title>
     <updated>2011-04-14T10:27:50.672Z</updated>
     <link href="http://localhost:6405/biprws/infostore/Application%20Folder/children"
   rel="http://www.sap.com/rws/bip#children"></link>
     <link href="http://localhost:6405/biprws/infostore" rel="up"></link>
     <content type="application/xml">
       <attrs xmlns="http://www.sap.com/rws/bip">
         <attr name="id" type="int32">43</attr>
         <attr name="cuid" type="string">AdoctK9h1sBHp3I6uG0Sh7M</attr>
         <attr name="description" type="string"></attr>
         <attr name="name" type="string">Application Folder</attr>
         <attr name="type" type="string">Folder</attr>
       </attrs>
     </content>
   </entry>
   ```

Related Topics

• /infostore/cuid_<cuid>

4.5.4 To access child objects

Before you can retrieve a resource from the BI platform, you must have a valid logon token and know the base URL for RESTful web service requests.

You can access the children of a parent resource by appending `/children` to the end of the RESTful web service request.

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

4. Use the `GET` method to send a request to the `http://<baseURL>/biprws/infostore/<id>/children` URL.

   Replace `<baseURL>` with the base URL for RESTful web service requests, and replace `<id>` with the ID or cuid_CUID of the parent object you want to retrieve.

   This example requests the children of the Root Folder by its ID, 23.

   ```
   http://<baseURL>/biprws/infostore/23/children
   ```

   The response contains a `<feed>` element contains `<entry>` elements for each child of the requested resource. In this example, the children of the Root Folder are returned, including entries for Data Federation, Feature Samples, Web Intelligence Samples, and more.

   ```
   <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>2011-04-15T00:31:16.609Z</updated>
     <link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=1&amp;pageSize=50"
   rel="self"></link>
     <link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=1&amp;pageSize=50"
   rel="first"></link>
     <link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=1&amp;pageSize=50"
   rel="last"></link>
     <entry>
       <title type="text">Data Federation</title>
       <id>tag:sap.com,2010:bip-rs/FnKsrkkctAcA8BAAALB7kkQAADAFzVMX</id>
       <author><name>System Account</name></author>
       <link href="http://localhost:6405/biprws/infostore/4044" rel="alternate"></link>
       <content type="application/xml">
         <attrs xmlns="http://www.sap.com/rws/bip">
           <attr name="id" type="int32">4044</attr>
           <attr name="cuid" type="string">FnKsrkkctAcA8BAAALB7kkQAADAFzVMX</attr>
           <attr name="description" type="string" null="true"></attr>
           <attr name="type" type="string">Folder</attr>
         </attrs>
       </content>
     </entry>
     <entry>
       <title type="text">Feature Samples</title>
       <id>tag:sap.com,2010:bip-rs/AfoyR1BSRYJIgOkbmWfd3zU</id>
       <author><name>Administrator</name>
       <uri>http://localhost:6405/biprws/infostore/12</uri></author>
       <link href="http://localhost:6405/biprws/infostore/5158" rel="alternate"></link>
       <content type="application/xml">
         <attrs xmlns="http://www.sap.com/rws/bip">
         <attr name="id" type="int32">5158</attr>
         <attr name="cuid" type="string">AfoyR1BSRYJIgOkbmWfd3zU</attr>
         <attr name="description" type="string">Contains examples of new features</attr>
         <attr name="type" type="string">Folder</attr>
         </attrs>
       </content>
     </entry>

   ...

     <entry>
       <title type="text">Web Intelligence Samples</title>
       <id>tag:sap.com,2010:bip-rs/AeN4lEu0h_tAtnPEjFYxwi8</id>
       <author><name>Administrator</name>
       <uri>http://localhost:6405/biprws/infostore/12</uri></author>
       <link href="http://localhost:6405/biprws/infostore/4946" rel="alternate"></link>
       <content type="application/xml">
         <attrs xmlns="http://www.sap.com/rws/bip">
         <attr name="id" type="int32">4946</attr>
         <attr name="cuid" type="string">AeN4lEu0h_tAtnPEjFYxwi8</attr>
         <attr name="description" type="string"></attr>
         <attr name="type" type="string">Folder</attr>
         </attrs>
       </content>
     </entry>
   </feed>
   ```

Related Topics

- /infostore/<id>/children

4.5.5 To use pagination with results

Before you can retrieve a resource from the BI platform, you must have a valid logon token and know the base URL for RESTful web service requests.

When a response contains a large number of entries, you can divide the entries into pages and view one page at a time. You can set the number of entries that appear on a page, and then request the page number that you want to view.

Note:
If you do not explicitly set the pagination information, then results are returned according to the default page size, which is set by an administrator. The default value is 50 entries per page.

1. Create a new HTTP request.
2. Add the `X-SAP-LogonToken` attribute to the request header and set its value to a valid logon token.
3. Add the `Accept` attribute to the request header and set its value to `application/xml`.
4. Append `?page=<n>&pageSize=<m>` to the end of the URL that requests a feed that contains multiple entries.

   Replace `<n>` with the page number of the page you want to view. Replace `<m>` with the number of entries to display on each page.

   This example requests to return the children of object with ID=23. It requests the second page of results, where each page contains three entries.

   ```
   http://<baseURL>/biprws/infostore/23/children?page=2&pageSize=3
   ```

5. Use the `GET` method to send the request.

The response contains a list of entries for the requested page. It also returns a set of links that you can use to see the first, last, next, and previous pages. This example shows the second page, where each page contains three entries.

```
<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>2011-04-07T23:50:17.983Z</updated>
<link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=2&amp;pageSize=3"
rel="self"></link>
<link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=1&amp;pageSize=3"
rel="first"></link>
<link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=3&amp;pageSize=3"
rel="next"></link>
<link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=1&amp;pageSize=3" rel="previ
ous"></link>
<link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=3&amp;pageSize=3"
rel="last"></link>
<entry>
  <title type="text">Platform Search Scheduling</title>
  <id>tag:sap.com,2010:bip-rs/AfbVaQ1CdrNDkKlzAKEK3aI</id>
  <author><name>System Account</name></author>
  <link href="http://localhost:6405/biprws/infostore/4320" rel="alternate"></link>
```

```
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="id" type="int32">4320</attr>
      <attr name="cuid" type="string">AfbVaQ1CdrNDkKlzAKEK3aI</attr>
      <attr name="description" type="string" null="true"></attr>
      <attr name="type" type="string">Folder</attr></attrs>
  </content>
</entry>
<entry>
  <title type="text">Probes</title>
  <id>tag:sap.com,2010:bip-rs/AYtU9ijcgpxFsbgLW0om5_U</id>
  <author><name>System Account</name></author>
  <link href="http://localhost:6405/biprws/infostore/4001" rel="alternate"></link>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="id" type="int32">4001</attr>
      <attr name="cuid" type="string">AYtU9ijcgpxFsbgLW0om5_U</attr>
      <attr name="description" type="string" null="true"></attr>
      <attr name="type" type="string">Folder</attr>
    </attrs>
  </content>
</entry>
<entry>
  <title type="text">Report Conversion Tool</title>
  <id>tag:sap.com,2010:bip-rs/AY9zJ8BgaF9OucZ2h2slcJM</id>
  <author><name>Administrator</name>
  <uri>http://localhost:6405/biprws/infostore/12</uri></author>
  <link href="http://localhost:6405/biprws/infostore/4082" rel="alternate"></link>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="id" type="int32">4082</attr>
      <attr name="cuid" type="string">AY9zJ8BgaF9OucZ2h2slcJM</attr>
      <attr name="description" type="string"></attr>
      <attr name="type" type="string">Folder</attr>
    </attrs>
  </content>
  </entry>
</feed>
```

Related Topics

- ?page=<n>&pageSize=<m>
- To set the default number of entries displayed on each page

4.5.6 To filter results by type

Before you can retrieve a resource from the BI platform, you must have a valid logon token and know the base URL for RESTful web service requests.

You can limit the type of results returned by a response by appending `?type=<type>` to the end of the RESTful web service request. Replace `<type>` with the type of results you want to see. The `<type>` value corresponds to the `SI_KIND` property of the object in the BI platform repository.

1. Create a new HTTP request.
2. Add the `X-SAP-LogonToken` attribute to the request header and set its value to a valid logon token.
3. Add the `Accept` attribute to the request header and set its value to `application/xml`.
4. Append `?type=<type>` to the end of a URL that requests a feed that contains multiple entries.

Replace `<type>` with the type of result you want to be returned. This example requests to return the children of the folder with ID=99 that have the type InfoView.

```
http://<baseURL>/biprws/infostore/99/children?type=InfoView
```

5. Use the `GET` method to send the request.

```
GET http://<baseURL>/biprws/infostore/99/children?type=InfoView
```

The response contains a `<feed>` element that contains `<entry>` elements for children of object `99` that are of type `InfoView`.

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>tag:sap.com,2010:bip-rs/AWItAeqx.FpBgqTpFH8LqwE/children</id>
  <title type="text">Children of Root Folder 99</title>
  <updated>2011-06-06T23:40:10.209Z</updated>
  <link href="http://localhost:6405/biprws/infostore/99/children?page=1&pageSize=50" rel="self"></link>
  <link href="http://localhost:6405/biprws/infostore/99/children?page=1&pageSize=50" rel="first"></link>

  <link href="http://localhost:6405/biprws/infostore/99/children?page=1&pageSize=50" rel="last"></link>
  <entry>
    <title type="text">BI launch pad</title>
    <id>tag:sap.com,2010:bip-rs/Ac7UIwmYafpFuhiiw6FRXLQ</id>
    <author><name>System Account</name></author>
    <link href="http://localhost:6405/biprws/infostore/474" rel="alternate"></link>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id" type="string">474</attr>
        <attr name="cuid" type="string">Ac7UIwmYafpFuhiiw6FRXLQ</attr>
        <attr name="description" type="string" null="true"></attr>
        <attr name="type" type="string">InfoView</attr>
      </attrs>
    </content>
  </entry>
</feed>
```

Related Topics

- ?type=<type>

4.5.7 To access objects with relationships

Before you can retrieve a resource from the BI platform, you must have a valid logon token and know the base URL for RESTful web service requests.

You can access objects that are related to the currently listed object by appending its object ID, then append `/relationships` to the URL followed by the name of the relationship. Further, you can make more specific queries by adding the ID or CUID of an object. A relationship can be, for example, a resource such as an account named Administrator that is associated with with other objects such as user groups, received alerts and subscribed events. Use of the `/infostore/<id>` API will return relationship information on the InfoObject with `<id>` if such associations exist. For more information on relationships, consult the Business Intelligence Platform Administator Guide.

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

4. Use the `GET` method to send a request to the `http://<baseURL>/biprws/infostore/<id>/relationships/<id>` URL.

   Replace `<baseURL>` with the base URL for RESTful web service requests, and replace `<id>` with the `ID` or `CUID` of the object you want to retrieve.

   To illustrate relationships, the following example begins by using the `/infostore` API. This will reveal if an object with an ID of `12` has any relationships to other objects in the BI platform.

   ```
   http://commandcom-lcm:6405/biprws/infostore/12
   ```

   The response shows that ID 12 is an `Administrator` object that has relationships that include include `subscribedEvents`, `userGroups` and `receivedAlerts`.

   ```
   <?xml version="1.0" ?>
   <entry xmlns="http://www.w3.org/2005/Atom">
     <author>
       <name>
         Administrator
       </name>
       <uri>
         http://commandcom-lcm:6405/biprws/infostore/12
       </uri>
     </author>
     <id>
       tag:sap.com,2010:bip-rs/AfRWaT5_131NlLLf5bRMLKY
     </id>
     <title type="text">
       Administrator
     </title>
     <updated>
       2012-01-04T20:03:20.085Z
     </updated>
     <link href="http://commandcom-lcm:6405/biprws/infostore/Users" rel="up"/>
     <link href="http://commandcom-lcm:6405/biprws/infostore/12/relationships/subscribedEvents"
           rel="http://www.sap.com/rws/bip#subscribed-events" title="Subscribed events"/>
     <link href="http://commandcom-lcm:6405/biprws/infostore/12/relationships/userGroups"
           rel="http://www.sap.com/rws/bip#user-groups" title="User groups"/>
     <link href="http://commandcom-lcm:6405/biprws/infostore/12/relationships/receivedAlerts"
           rel="http://www.sap.com/rws/bip#received-alerts" title="Received alerts"/>
     <content type="application/xml">
       <attrs xmlns="http://www.sap.com/rws/bip">
         <attr name="id" type="int32">
           12
         </attr>
         <attr name="cuid" type="string">
           AfRWaT5_131NlLLf5bRMLKY
         </attr>
         <attr name="description" type="string">
           Administrator account
         </attr>
         <attr name="name" type="string">
           Administrator
         </attr>
         <attr name="type" type="string">
           User
         </attr>
         <attr name="emailAddress" type="string"/>
         <attr name="lastLogon" type="datetime">
           2012-01-04T20:03:20.085Z
         </attr>
         <attr name="fullName" type="string"/>
       </attrs>
     </content>
   </entry>
   ```

   The following code snippet uses the `.../relationship/users` link obtained from the previous example.

   ```
   http://commandcom-lcm:6405/biprws/infostore/12/relationships/userGroups
   ```

Since a trailing ID number was not used, the response in the following code snippet lists 3 links that may be examined further. These are `../infostore/1`, `../infostore/2` and `../infostore/3`.

```
<?xml version="1.0" ?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>
      Administrator
    </name>
    <uri>
      http://commandcom-lcm:6405/biprws/infostore/12
    </uri>
  </author>
  <id>
    tag:sap.com,2010:bip-rs/AfRWaT5_131NlLLf5bRMLKY/relationships/userGroups
  </id>
  <title type="text">
    InfoObjects related to Administrator via userGroups
  </title>
  <updated>
    2012-01-04T20:08:32.441Z
  </updated>
  <entry>
    <title type="text">
      1
    </title>
    <id>
      tag:sap.com,2010:bip-rs/AfRWaT5_131NlLLf5bRMLKY/relationships/userGroups/1
    </id>
    <link href="http://commandcom-lcm:6405/biprws/infostore/12/relationships/userGroups/1" rel="self"/>
    <link href="http://commandcom-lcm:6405/biprws/infostore/1" rel="related"/>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id" type="int32">
          1
        </attr>
      </attrs>
    </content>
  </entry>
  <entry>
    <title type="text">
      2
    </title>
    <id>
      tag:sap.com,2010:bip-rs/AfRWaT5_131NlLLf5bRMLKY/relationships/userGroups/2
    </id>
    <link href="http://commandcom-lcm:6405/biprws/infostore/12/relationships/userGroups/2" rel="self"/>
    <link href="http://commandcom-lcm:6405/biprws/infostore/2" rel="related"/>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id" type="int32">
          2
        </attr>
      </attrs>
    </content>
  </entry>
  <entry>
    <title type="text">
      3
    </title>
    <id>
      tag:sap.com,2010:bip-rs/AfRWaT5_131NlLLf5bRMLKY/relationships/userGroups/3
    </id>
    <link href="http://commandcom-lcm:6405/biprws/infostore/12/relationships/userGroups/3" rel="self"/>
    <link href="http://commandcom-lcm:6405/biprws/infostore/3" rel="related"/>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id" type="int32">
          3
        </attr>
      </attrs>
    </content>
  </entry>
</feed>
```

Related Topics

• /infostore

4.6 Scheduling

The Business Intelligence platform RESTful web service SDK enables you to perform basic scheduling.

Note:
Only schedulable objects can be used with the scheduling API. Schedulable resources include documents, for example Crystal reports and WebI reports.

You can set the following scheduling properties:

- The time to schedule the resource.
- The recurrence properties of the resource, including the start time, end time, and recurrence interval.

  For example, a report could be scheduled to recur every Monday morning for the next year.
- The number of retries allowed and the retry interval.

  For example, if scheduling fails, you could allow up to three retries at hourly intervals.

4.6.1 To discover the scheduling URLs for an object

Before you can discover the URLs for scheduling an object, you must have a valid logon token and know the base URL for RESTful web service requests.

To get a list of URLs that can be used to schedule an object, append `/scheduleForms` to the end of a request for a schedulable resource. Schedulable resources include documents, for example Crystal reports and WebI reports.

1. Create a new HTTP request.
2. Add the `X-SAP-LogonToken` attribute to the request header and set its value to a valid logon token.
3. Add the `Accept` attribute to the request header and set its value to `application/xml`.
4. Use the `GET` method to send a request to the `http://<baseURL>/biprws/infostore/<id>/scheduleForms` URL.

   Replace `<baseURL>` with the base URL for RESTful web service requests, and replace `<id>` with the ID or CUID of a schedulable resource.

   ```
   GET http://localhost:6405/biprws/infostore/4738/scheduleForms
   ```

   The response contains a feed of entries that show the links for scheduling the resource. The following example shows links that you can use to schedule a report with the following recurrence:

   - now

- once
- hourly
- daily
- weekly
- monthly
- NthDayOfMonth

Note:
If the resource is not schedulable, an error is returned.

```
<feed xmlns="http://www.w3.org/2005/Atom">
<author>
  <name>Administrator</name>
  <uri>http://localhost:6405/biprws/infostore/12</uri>
</author>
<id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/scheduleForms</id>
<title type="text">Schedule Drilldown</title><updated>2011-05-18T10:31:30.092Z</updated>
  <entry>
    <title type="text">now</title><id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/now</id>
    <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/now" rel="alternate"></link>
  </entry>
  <entry>
    <title type="text">once</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/once</id>
    <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/once" rel="alternate"></link>
  </entry>
  <entry>
    <title type="text">hourly</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/hourly</id>
   <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/hourly" rel="alternate"></link>

  </entry>
  <entry>
    <title type="text">daily</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/daily</id>
   <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/daily" rel="alternate"></link>

  </entry>
  <entry>
    <title type="text">weekly</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/weekly</id>
   <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/weekly" rel="alternate"></link>

  </entry>
  <entry>
    <title type="text">monthly</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/monthly</id>
   <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/monthly" rel="alternate"></link>

  </entry>
  <entry><title type="text">NthDayOfMonth</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/NthDayOfMonth</id>
    <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/NthDayOfMonth" rel="alter
nate"></link>
  </entry>
</feed>
```

Related Topics

- /scheduleForms

4.6.2 To schedule a resource

Before you can schedule a resource, you must have obtained a valid logon token and know the base URL for RESTful web service requests.

The Business Intelligence platform RESTful web service SDK allows for basic scheduling, including setting the time to schedule the resource and recurrence information.

Note:
The scheduling APIs only work with objects that are schedulable. Schedulable resources include documents, for example Crystal reports and WebI reports.

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. Add the `Accept` attribute to the request header and set its value to `application/xml`.
4. Use the `GET` method to send a request to the `http://<baseURL>/biprws/infostore/<id>/scheduleForms/<form>` URL.

   Replace `<baseURL>` with the base URL for RESTful web service requests, and replace `<ID>` with the ID or cuid_CUID of the resource. Replace `<form>` with the frequency of scheduling to perform, for example, `now`, `daily`, `weekly`, or `once`.

   ```
   GET http://localhost:6405/biprws/infostore/4738/scheduleForms/now
   ```

   The response contains an XML template that you can use to populate the request body of a request to schedule the resource.

   ```
   <entry xmlns="http://www.w3.org/2005/Atom">
     <author>
       <name>Administrator</name>
       <uri>http://localhost:6405/biprws/infostore/12</uri>
     </author>
     <id>tag:sap.com,2010:bip-rs/ASb6Obs1HktFnk3uF8.g3tw/scheduleForms/now</id>
     <title type="text">Schedule Drilldown now</title>
     <updated>2011-05-18T10:31:30.092Z</updated>
     <content type="application/xml">
       <attrs xmlns="http://www.sap.com/rws/bip">
         <attr name="retriesAllowed" type="int32">0</attr>
         <attr name="retryIntervalInSeconds" type="int32">1800</attr>
       </attrs>
     </content>
   </entry>
   ```

5. Create a new HTTP request.
6. Add the `X-SAP-LogonToken` attribute to the request header and set its value to a valid logon token.
7. Add the `Accept` attribute to the request header and set its value to `application/xml`.
8. Add the `Content-Type` attribute to the request header and set its value to `application/xml`.
9. Fill out the template received from the `GET` request, and add it to the new request body.

   In this example, 3 retries are allowed in intervals of 1800 seconds.

   ```
   <entry xmlns="http://www.w3.org/2005/Atom">
     <author>
       <name>Administrator</name>
       <uri>http://localhost:6405/biprws/infostore/12</uri>
     </author>
     <id>tag:sap.com,2010:bip-rs/ASb6Obs1HktFnk3uF8.g3tw/scheduleForms/now</id>
     <title type="text">Schedule Drilldown now</title>
     <updated>2011-05-18T10:31:30.092Z</updated>
     <content type="application/xml">
       <attrs xmlns="http://www.sap.com/rws/bip">
         <attr name="retriesAllowed" type="int32">3</attr>
         <attr name="retryIntervalInSeconds" type="int32">1800</attr>
   ```

```
    </attrs>
  </content>
</entry>
```

10. Use the `POST` method to send the request to the scheduling URL.

    ```
    POST http://localhost:6405/biprws/infostore/4738/scheduleForms/now
    ```

    If the resource is scheduled successfully, the response header contains the status code `201 Created`, and provides a link to the location of the scheduled instance.

    ```
    Location: http://localhost:6405/biprws/infostore/5619
    ```

Related Topics

• /scheduleForms/<form>

Administration and installation tasks

This section is about installing and configuring RESTful web services on a BI platform installation.

To perform the tasks in this section, you must be a BI platform administrator. Administrators can configure the RESTful web services environment, including setting default system values, enabling features, and enhancing performance settings.

The default installation of the BI platform includes RESTful web services. However, if you have performed a custom installation of the BI platform and did not include RESTful web services, you can install it separately. RESTful web services require an instance of the Web Application Container Server (WACS), and installing RESTful web services will install a copy of the WACS server if one does not already exist.

In a complex deployment environment, for example one that uses a proxy or multiple instances of the WACS server, you may need to configure the server name and port that is used to listen to RESTful web service requests.

Note:
For additional information on complex deployment scenarios, see the “Managing Web Application Container Servers (WACS)” section of the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

5.1 To install RESTful web services on Windows

You can use the Windows installer to add RESTful web services to your custom BI platform deployment. RESTful web services requires an instance of the Web Application Container Server (WACS), which is installed with RESTful web services if it does not already exist. RESTful web services was introduced in BI platform 4.0 to Feature Pack 3.

- If your BI platform 4.0 FP3 is a new installation, RESTful Web Services is automatically included in the installation. If you choose custom install, RESTful Web Services is selected in the feature tree by default.
- If you are upgrading from 4.0 SP2 to 4.0 FP3, after completing the upgrade, use the **Programs and Features Windows Control Panel**, **Uninstall/Change** feature to add the RESTful web service.

For more information about installing the BI platform on Windows, see the *Business Intelligence Platform Installation Guide for Windows*, section 5.8.1 To modify SAP BusinessObjects Business Intelligence platform.

1. Start the Business Intelligence platform Windows installer, and follow the installation instructions for a custom installation.

2. On the **Select Features** screen, expand **Instances** > **Servers** > **Platform Services** and select **RESTful Web Services**.
3. Continue the installation.
4. On the **HTTP Listening Port Configuration** dialog, enter the port number for listening to RESTful web service requests.

   By default, the port number is 6405.
5. Complete the installation.

If your BI platform installation uses a proxy or more than one WACS server, you may need to configure the base URL for RESTful web services.

Related Topics

• To configure the base URL for RESTful web services

5.2 To install RESTful web services on Unix

You can use the Unix installer to add RESTful web services to your custom BI platform deployment. RESTful web services requires an instance of the Web Application Container Server (WACS), which is installed with RESTful web services if it does not already exist.

For more information about installing the BI platform on Unix, see the *Business Intelligence Platform Installation Guide for Unix*.

1. Start the Business Intelligence platform Unix installer, and follow the installation instructions for a custom installation.
2. On the **Select Features** dialog, expand **Instances** > **Servers** > **Platform Services** and select **RESTful Web Services**.
3. Continue with the installation.
4. On the **HTTP Listening Port Configuration** dialog, enter the port number for listening to RESTful web service requests.

   By default, this port is set to 6405.
5. Complete the installation.

If your BI platform installation uses a proxy or more than one WACS server, you may need to configure the base URL for RESTful web services.

Related Topics

• To configure the base URL for RESTful web services

5.3 To configure web.xml to enable WinAD SSO

Configuring the RESTful web services to recognize Windows Active Directory Single Sign-On (WinAD SSO) requires edits to the `web.xml` configuration file, located on the BI platform server. For more information, see “Using the SDK > Authentication > To get a logon token using an Active Directory Single Sign-On (AD SSO) account” in the *Business Intelligence Platform RESTful Web Service Developer Guide*.

To have a client computer WinAD SSO login credentials recognized by the BI platform server, you must uncomment the `Kerberos Proxy filter` section of the `web.xml` and update values for `idm.realm`, `idm.princ` and `idm.keytab` that reflect the active directory environment used.

1. Locate the `web.xml` configuration at `<boe root>\SAP BusinessObjects Enterprise XI 4.0\java\pjs\services\RestWebService\biprws\WEB-INF\`. The following filepath is an example.

   ```
   C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\java\
   pjs\services\RestWebService\biprws\WEB-INF\web.xml
   ```

2. In the `web.xml` file, uncomment the Kerberos Proxy Filter section by adding a comment close tag `-->` before the `<filter>` tag, and remove the closing comment tag `-->`

   ```
     <!-- Kerberos Proxy Filter
       - Uncomment this filter and the corresponding filter-mapping to enable Kerberos SSO
       - for Windows AD (secWinAD) authentication.
       - The following options must be specified (the rest are optional):
       -   idm.realm
       -   idm.princ
       -   idm.keytab (unless using password, see below)
     -->

     <filter>
       <filter-name>WrappedResponseAuthFilter</filter-name>
         .
         .
         .
     </filter>

     <filter-mapping>
       <filter-name>WrappedResponseAuthFilter</filter-name>
       <url-pattern>/logon/adsso</url-pattern>
     </filter-mapping>

   </web-app>
   ```

3. Update the `<param-value>` for each setting of `idm.realm`, `idm.princ` and `idm.keytab` with those used in your active directory environment.

   ```
       <init-param>
         <param-name>idm.realm</param-name>
         <param-value>ADDOM.COM</param-value>
         <description>
             Required: Set this value to the Kerberos realm to use.
         </description>
       </init-param>

       <init-param>
         <param-name>idm.princ</param-name>
         <param-value>BOE120SIAVMBOESRVR/bo.service.addom.com</param-value>

         <description>
             Set this value to the Kerberos service principal to use.
             This will be a name of the form HTTP/fully-qualified-host.
   ```

```
        For example, HTTP/example.vintela.com
        If not set, defaults to the server's hostname and the
        idm.realm  property above.
    </description>
  </init-param>

  <init-param>
    <param-name>idm.kdc</param-name>
    <param-value></param-value>
    <description>
        The KDC against which secondary credentials must be validated
        This can be used for BASIC fallback or credential delegation.
        By default the KDC will be discovered automatically and this
        parameter must only be used if automatic discovery fails, or
        if a different KDC to the one discovered must automatically be used.
    </description>
  </init-param>

  <init-param>
    <param-name>idm.keytab</param-name>
    <param-value>C:/winnt/BOE120SIAVMBOESRVR.keytab</param-value>
    <description>
        The file containing the keytab that Kerberos will use for
        user-to-service authentication. If unspecified, SSO will default
        to using an in-memory keytab with a password specified in the
        com.wedgetail.idm.sso.password environment variable.
    </description>
  </init-param>
```

Note:
The `idm.keytab` value refers to a filepath on the BI platform server. Values for `idm.realm` and `idm.prince` may be viewed from the Central Management Console. On the **Authentication** tab In the CMC, double-click **Windows AD**. The value for `idm.realm` is set with the "Default AD Domain" parameter, under "AD Configuration Summary". The value for `idm.prince` is set with the "Service principal name" parameter, under "Authentication Options".

4. Restart the WACS service so that the changes made to `web.xml` are recognized.
5. Use a client machien to verify that an AD SSO login token may be retrieved using the RESTful Web Services API, (for example, `http://<boe host>:6405/biprws/logon/adsso`).
6. Test the token by using a `GET` query including `X-SAP-LogonToken` in the header and using the `/infostore` API.

5.4 To configure Methods and Headers command line parameters

As an administrator, you can restrict what methods and headers may be used by RESTful web services, by adding the appropriate options to "Command Line Parameters" in the properties of your Web Application Container Service (WACS). Changes to the parameters require restarting the WACS service.

1. Log on to the Central Management Console as an administrator user.
2. Click **Servers**, and then click **Servers List**.
3. Right-click on your Web Application Container Server (WACS); for example, `MySIA.WebApplicationContainerServer`, and click **Properties**.

   The **Properties** tab for the WACS server appears.
4. In the "Command Line Parameters" area, enter the methods and headers that will be allowed.

Each option group is enclosed by double quotes. Use Methods other than `GET`, `HEAD` and `POST`. Use commas to separate the option values such as `PUT` and `DELETE` as shown in the following example.

```
"-Dcom.sap.bip.rs.cors.extra.methods= PUT, DELETE"
"-Dcom.sap.bip.rs.cors.extra.headers= X-SAP-LogonToken, X-SAP-PVL, WWW-Authenticate"
```

Note:
The default value to allow all methods and headers is `*` (asterisk). Omitting the command line parameters entirely, has the same effect.

5. Click **Save and Close**.
6. Restart the service by right-clicking on the WACS server name, for example `MySIA.WebApplicationContainerServer` and click **Restart Server**.

5.5 To configure the base URL for RESTful web services

If your BI platform deployment uses a proxy server or contains more than one instance of the Web Application Container Server (WACS), you may need to configure the base URL for use with RESTful web services. Before you configure the base URL, you must know the server name and port number that listens to RESTful web service requests.

The base URL is used as part of every RESTful web service request. Developers programmatically discover the base URL and use it to direct RESTful web service requests to the correct server and port. The base URL is also used in RESTful web service responses to define hyperlinks to other RESTful resources.

Note:

In default installations of the BI platform, the base URL is defined as `http://<server name>:6405/biprws`. Replace `<servername>` with the name of the server that hosts RESTful web services.

1. Log on to the Central Management Console (CMC) as an administrator.
2. In the CMC, click **Applications**.

   A list of applications is displayed.
3. Right-click **RESTful Web Service** > **Properties**.

   The "Properties" dialog box appears.
4. In the **Access URL** text box, type the name of the base URL for RESTful web services.

   For example, type `http://<servername>:<portnumber>/biprws`. Replace `<servername>` and `<portnumber>` with the name of the server and the port that listens to RESTful web service requests.
5. Click **Save and Close**.

5.6 To enable the error message stack

As an administrator, you can configure the error messages returned by RESTful web services to include the error stack. The error stack provides extra debugging information that can be used to discover where errors have occurred.

Note:
You may not want to enable the error stack in production scenarios, because it could provide information about the BI platform that you do not want to reveal to end users. It is recommended to enable the error stack in production scenarios as required for debugging, and to turn it off when it is no longer needed.

1. Log on to the Central Management Console as administrator user.
2. Click **Servers**, and then click **Servers List**.
3. Right-click on your Web Application Container Server (WACS); for example, right-click on `MySIA.WebApplicationContainerServer`, and click **Properties**.

   The **Properties** tab for the WACS server appears.
4. In the **RESTful Web Service** area, select **Show Error Stack**.
5. Click **Save and Close**.

Error stack information is included in RESTful web service error messages.

5.7 To set the default number of entries displayed on each page

When a RESTful web service response contains a feed with a large number of entries, the response can be divided into pages. You can configure the default number of entries that are displayed on each page. When developers make RESTful web service requests, they can specify the number of entries to display on each page. However, if they do not specify this value then the default page size is used.

1. Log on to the Central Management Console as an administrator.
2. Click **Servers**, and then click **Servers List**.
3. Right-click on your Web Application Container Server (WACS); for example, right click on `MySIA.WebApplicationContainerServer`, and click **Properties**.

   The **Properties** tab for the WACS server appears.
4. In the **RESTful Web Service** area, type the default page size in the **Default Number of Objects on One Page** text area.
5. Click **Save and Close**.

5.8 To set the timeout value of a logon token

Logon tokens expire after they have not been used for a certain amount of time. You can set the amount of time that an unused logon token remains valid.

Note:
By default, the logon token timeout value is one hour.

1. Log on to the Central Management Console as an administrator.
2. Click **Servers**, and then click **Servers List**.
3. Right-click on your Web Application Container Server (WACS); for example, right click on `MySIA.WebApplicationContainerServer`, and click **Properties**.

   The **Properties** tab for the WACS server appears.
4. In the **RESTful Web Service** area, type the number of minutes for a logon token to be valid in the **Enterprise Session Token Timeout (minutes)** text area.
5. Click **Save and Close**.

5.9 To configure session pool settings

You can improve server performance by using a session pool. The session pool caches active RESTful web service sessions so they can be reused when a user sends another request that uses the same logon token in the HTTP request header. The session pool size defines the number of cached sessions to be stored at one time, and the session timeout value controls the amount of time that a session is cached.

You can set the session pool size and the session timeout value:

1. Log on to the Central Management Console (CMC) as an administrator.
2. Click **Servers**, and then click **Servers List**.
3. Right-click on your Web Application Container Server (WACS); for example, right-click on `MySIA.WebApplicationContainerServer`, and click **Properties**.

   The **Properties** tab for the WACS server appears.
4. Type the maximum number of sessions to cache in the **Session Pool Size** text box of the **RESTful Web Service** area.
5. Type the session pool timeout value in the **Session Pool Timeout (minutes)** text box of the **RESTful Web Service** area.
6. Click **Save and Close**.
7. Right-click on the WACS server, for example, `MySIA.WebApplicationContainerServer`, and click **Restart Server**.

5.10 To enable HTTP basic authentication

HTTP basic authentication lets users make RESTful web service requests without providing a logon token. If HTTP basic authentication is enabled, users are prompted to provide their user name and password the first time they make a RESTful web service request.

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

When you enable HTTP basic authentication, you set the default HTTP basic authentication type to SAP, Enterprise, LDAP, or WinAD. Users can override the default HTTP basic authentication type when they log on.

Logging on to the BI platform using HTTP basic authentication consumes a license. If the session pool caching is used, the request uses the license associated with its cached session. If session pool caching is not used, a license is consumed while the request is in progress and released once the request is finished.

1. Log on to the Central Management Console (CMC) as an administrator.
2. Click **Server** > **Servers List**.
3. Right-click on your Web Application Container Server (WACS); for example, right-click on `MySIA.WebApplicationContainerServer`, and click **Properties**.

   The **Properties** tab for the WACS server appears.
4. In the "RESTful Web Service" area, select **Enable HTTP Basic Authentication**.
5. (Optional) In the **Default Authentication Scheme for HTTP Basic** list, select the default type of HTTP basic authentication.
6. Click **Save and Close**.

Note:

When an end user logs on using HTTP basic authentication, they can specify the type of authentication to use. In a web browser, the user types `<authtype>\<username>` in the user name prompt, and `<password>` in the password prompt.

To log on using HTTP basic authentication programmatically, users add the `Authorization` attribute to the HTTP request header, and set the value to be `Basic <authtype>\<username>:<password>`.

Replace `<authtype>` with the authentication type, `<username>` with the user name, and `<password>` with the password. The authentication type, user name, and password must be base64-encoded as defined by RFC 2617. User names that contain the `:` character cannot be used with HTTP basic authentication.

Related Topics

• To configure session pool settings

5.11 To configure cross-origin resource sharing (CORS)

The **Cross-Origin Resource Sharing Configuration** (CORS) setting allows you to add a list of domain names to let users retrieve data from multiple sources on JavaScript-based web pages. This is necessary to get around the security policy that JavaScript and Ajax languages employ to prevent cross-domain access. To avoid compromising security, only those websites that may be accessed are added to the **Allow Origins** WACS server properties in CMC.

A **Max Age (minutes)** setting is also available to adjust the cache expiry time, which sets the maximum number of minutes that browsers can retain HTTP requests.

Note:
By default, access to any and all domains are allowed with * (asterisk).

1. Log on to the Central Management Console as an administrator.
2. Click **Server** > **Servers List**.
3. Right-click on your Web Application Container Server (WACS); for example, right-click `MySIA.WebApplicationContainerServer`, and click **Properties**.

   The **Properties** tab for the WACS server appears.
4. In the **RESTful Web Service** area, go to the **Cross-Origin Resource Sharing Configuration** text box beside **Allow Origins:** and replace the * (asterisk) with your list of domain names, each separated by a comma. For example: `http://origin1.server:8080, http://origin2.server:8080`
5. In the **Max Age (minutes):** text box, type the maximum number of minutes that you want browsers to cache HTTP requests.
6. Click **Save and Close**.

5.12 To enable and configure trusted authentication

Trusted authentication is activated and configured through the Central Management Console (CMC) in areas that include **Authentication > Enterprise**, where Trusted Authentication is enabled and a shared secret key file is generated; **Users and Groups > User List**, where an account is created for a trusted user; and **Servers > Servers List > WACS > Properties**, where the "Retrieving Method" option is selected for `/logon/trusted` API logon token requests.

1. Log on to the Central Management Console as an administrator.
2. Go to **Authentication** > **Enterprise**, and then click **Trusted Authentication is enabled**.
3. Click **New Shared Secret**, and click **Download Shared Secret**.
4. Click **Save** and place the `TrustedPrincipal.conf` file in the default location, which is `<EnterpriseDir>\<platform>`.

An example location appears as follows:

```
"C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjectsEnterprise XI 4.0\win64_x64\"
```

Note:

- You can change the default location of the `TrustedPrincipal.conf` shared secret file by adding a command line entry in the CMC at **Servers > Servers List > WACS > Properties > Command Line Parameters**, and then restarting the WACS service. For example, a command line entry using `-Dbobj.trustedauth.home=` and the folder `SharedSecrets` placed at the root of the `C:\` drive of the BI platform server would appear as follows:

  ```
  "-Dbobj.trustedauth.home=C:\SharedSecrets"
  ```

- You can leave the option **Shared Secret Validity Period (days)** at the default value of zero (0) so that it does not expire. The **Trusted logon request is timeout after N millisecond(s) (0 means no limit)** option can be left at the default value of zero (0) so that there is no time limit for trusted logon requests.

5. Click **Update** to save the change.
6. Add a new user and password, for example `bob` and `Passw0rd`, in **Users and Groups** > **User List** using **Manage > New > New user**. Uncheck **User must change password at next logon**, then click **Create & Close**.

   Note:
   You can also create a new user by clicking the Create new user icon, or by right-clicking in an open area of the window that lists user names, and select **New** > **New User**.

7. Go to **Servers** > **Core Services** > **WACS** > **Properties**, scroll down to the "Trusted Authentication Configuration" section and use the "Retrieving Method" menu to select either **HTTP_HEADER**, **QUERY_STRING** or **COOKIE**.

   Note:
   You can optionally change the "User Name Parameter" from the default label of `X-SAP-TRUSTED-USER` to any other convenient label, (for example `UserName`, `bankteller`, or `nurse`) that RESTful web services developers must use.

8. Restart the service by right-clicking on the WACS server name, for example `MySIA.WebApplicationContainerServer`, and click **Restart Server**.

   Note:
   Later changing the option under "Retrieving Method" as shown in step 7 does not require restarting WACS.

9. Verify that you are able to retrieve a logon token by using the `.../biprsw/logon/trusted/` API and sending a `GET` request with the default header label of `X-SAP-TRUSTED-USER` with the user name created in step 5.

5.13 Securing Microsoft Silverlight access to the WACS server

Microsoft Silverlight components that are hosted in external applications can access the BI platform by using the Business Intelligence platform RESTful web service SDK. As an administrator, you can enhance the security of the BI platform by restricting which domains are authorized to make Silverlight requests to applications hosted by the Web Application Container Server (WACS), including RESTful web services.

Note:
Default installations of the BI platform allow unrestricted access to the WACS server by external Silverlight components.

The Silverlight access policy is defined by the `ClientAccessPolicy.xml` file, which is shared by all instances of the WACS server in a BI platform installation. Modifying the Silverlight access policy file changes the Silverlight access restrictions for all applications that are hosted by any WACS server in a BI platform deployment. This includes RESTful web services, and may include other BI platform web applications such as the Central Management Console (CMC) and BI Launch Pad if they are hosted by the WACS server.

Note:
RESTful web services are always hosted by a WACS server and cannot be hosted by another type of servlet container.

The `ClientAccessPolicy.xml` file is located at `$ENTERPRISEDIR/warfiles/webapps/ROOT`, where `$ENTERPRISEDIR` represents the location of your BI platform installation. Modify this file to change the Silverlight policy settings. After editing this file, you must restart the WACS servers for the changes to take effect.

For more information about how to edit a Silverlight policy file, consult the Microsoft Silverlight product documentation.

API reference

The API reference lists the URLs that can be used to access the BI platform with the Business Intelligence platform RESTful web service SDK. To use these URLs, append them to the end of the base URL.

Related Topics

- Retrieving the base URL for RESTful web service requests

6.1 RESTFul Web Service URIs summary list

The following table summarizes the available RESTful Web Service URIs. The root URI for the services listed in the following table is `http://<host>:<port>/biprws`. The default port is 6405. Feed refers to Atom Feed, and Entry refers to Atom Entry.

RESTFul Web Service URIs and response

The following table lists the API, the response, a URI example, and a comment or reference to sample.

RESTful Web Service API	Response	URI Example	Comments
`/`	Service document that contains a link to the `/infostore` API.		This is the root level of an infostore resource
`/infostore`	Feed contains all the folders with `SI_PARENT_ID =4`.		
`/infostore/<id>`	Entry corresponding to the info object with `SI_ID=<id>`.	`/99`	See sample 1.
`/infostore/cuid_<cuid>`	Entry corresponding to the info object with `SI_CUID=<cuid>`.	`/cuid_ASHnC0S_Pw5LhKFbZ.iA_j4`	
`/infostore/<id>/children`	Feed contains links to all children of info objects with `SI_PARENTID=<id>`.	`/99/children`	The info object that has the `<id>` (integer value) has children.

RESTful Web Service API	Response	URI Example	Comments
/infostore/<id>/children?kind=<kind>	Feed contains links to the children of info objects with `SI_PARENTID=<id>` and `SI_KIND=<kind>`.	/99/children?kind=InfoView	See sample 2.
/infostore/<id>/children?page=<n>&pageSize=<size>	Feed contains page number `<n>` data with page size `<size>`.	/23/children?page=1&pageSize=2	See sample 3. The default number for `page` is `1` and for `pageSize` it's `1000`. A relationship feed does not support feed paging.
/infostore/<id>/relationships/<type>	Feed contains specific info object links (belongs to `<type>`) related to info object with `SI_ID=<id>`.	/12/relationships/userGroups/12/relationships/receivedAlerts	`receiveAlert` belongs to the attribute `relationships`, and its entry contains certain attributes based on `receiveAlert` model.
/infostore/<id>/relationships/<type>/<another id>	Entry contains relationship information.	/12/relationships/receivedAlerts/5432	See sample 4.
/infostore/<id>/scheduleForms	Feed contains links to all the available scheduling forms for an info object with `SI_ID=<id>`.	/4738/scheduleForms	The corresponding info object of `<id>` is schedulable.
/infostore/<id>/scheduleForms/<form>	Entry form with specific content.	/4738/scheduleForms/daily	Supports both `GET` and `POST` requests. For a `POST` request, the info object will be scheduled.
/logon/long	`GET` returns the long form for logon, which contains the user and password authentication template. `POST` returns the logon token when the authentication form is posted.		
/logon/token	The token form for logon contains only the token parameter.		

RESTful Web Service API	Response	URI Example	Comments
`/logon/adsso`			Use `GET` to get a token through AD SSO. The user has already entered their credentials through Windows Active Directory,
`/logon/trusted`			Use `GET` to get a token using the Trusted Authentication API. The trusted user has been authenticated elsewhere, for example through a Windows Active Directory logon; only name, not a password is needed.
`/logoff`	empty body		Use `POST` and leave the body empty to explicitly log off the BI platform server.

1. Entry sample: /Infostore<id>

XML format:

```
  <author>
    <name>System Account</name>
  </author>
  <id>tag:sap.com,2010:bip-rs/AWItAeqx.FpBgqTpFH8LqwE</id>
  <title type="text">Root Folder 99</title>
  <updated>2011-12-22T16:33:33.721Z</updated>
  <link href="http://commandcom-lcm:6405/biprws/infostore/99/children"
rel="http://www.sap.com/rws/bip#children" title="Children"></link>
  <link href="http://commandcom-lcm:6405/biprws/infostore/Application%20Folder" rel="up"></link>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="id" type="int32">99</attr>
      <attr name="cuid" type="string">AWItAeqx.FpBgqTpFH8LqwE</attr>
      <attr name="description" type="string" null="true"></attr>
      <attr name="name" type="string">Root Folder 99</attr>
      <attr name="type" type="string">Folder</attr>
    </attrs>
  </content>
</entry>
```

JSON format:

```
{
   "Children":
     {"__deferred":
        {"uri":"http://commandcom-lcm:6405/biprws/infostore/99/children"}
     },
   "up":
     {"__deferred":
        {"uri":"http://commandcom-lcm:6405/biprws/infostore/Application%20Folder"}
     },
   "id":99,
   "cuid":"AWItAeqx.FpBgqTpFH8LqwE",
   "name":"Root Folder 99",
   "type":"Folder"
}
```

2. Feed sample: /infostore/<id>/children?kind=<kind>

XML format:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>tag:sap.com,2010:bip-rs/AWItAeqx.FpBgqTpFH8LqwE/children</id>
  <title type="text">Children of Root Folder 99</title>
  <updated>2011-12-22T22:25:38.729Z</updated>
  <link href="http://commandcom-lcm:6405/biprws/infostore/99/children?page=1&amp;pageSize=2" rel="self"></link>

  <link href="http://commandcom-lcm:6405/biprws/infostore/99/children?page=1&amp;pageSize=2"
rel="first"></link>
  <link href="http://commandcom-lcm:6405/biprws/infostore/99/children?page=2&amp;pageSize=2" rel="next"></link>

  <link href="http://commandcom-lcm:6405/biprws/infostore/99/children?page=14&amp;pageSize=2"
rel="last"></link>
  <entry>
    <title type="text">Alerting</title>
    <id>tag:sap.com,2010:bip-rs/AQvlGy105VlDlLws55dmm.0</id>
    <author>
      <name>System Account</name>
    </author>
    <link href="http://commandcom-lcm:6405/biprws/infostore/479" rel="alternate"></link>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id" type="int32">479</attr>
        <attr name="cuid" type="string">AQvlGy105VlDlLws55dmm.0</attr>
        <attr name="description" type="string" null="true"></attr>
        <attr name="name" type="string">Alerting</attr>
        <attr name="type" type="string">AlertingApp</attr>
      </attrs>
    </content>
  </entry>
  <entry>
    <title type="text">Analysis edition for OLAP</title>
    <id>tag:sap.com,2010:bip-rs/AVH4dKB.OpJBoK6b.fAUmjA</id>
    <author>
      <name>System Account</name>
    </author>
    <link href="http://commandcom-lcm:6405/biprws/infostore/3902" rel="alternate"></link>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id" type="int32">3902</attr>
        <attr name="cuid" type="string">AVH4dKB.OpJBoK6b.fAUmjA</attr>
        <attr name="description" type="string" null="true"></attr>
        <attr name="name" type="string">Analysis edition for OLAP</attr>
        <attr name="type" type="string">Pioneer</attr>
      </attrs>
    </content>
  </entry>
</feed>
```

JSON format:

```
{
   "__metadata":
     {"uri":"http://commandcom-lcm:6405/biprws/infostore/99/children?page=1&amp;pageSize=2"},

   "first":
     {"__deferred":
        {"uri":"http://commandcom-lcm:6405/biprws/infostore/99/children?page=1&amp;pageSize=2"}
     },
   "next":
     {"__deferred":
        {"uri":"http://commandcom-lcm:6405/biprws/infostore/99/children?page=2&amp;pageSize=2"}
     },
   "last":
     {"__deferred":
        {"uri":"http://commandcom-lcm:6405/biprws/infostore/99/children?page=14&amp;pageSize=2"}
     },
   "entries":
   [
      {"__metadata":
  {"uri":"http://commandcom-lcm:6405/biprws/infostore/479"},
  "id":479,
  "cuid":"AQvlGy105VlDlLws55dmm.0",
  "name":"Alerting",
  "type":"AlertingApp"
```

```
       },

      {"__metadata":
         {"uri":"http://commandcom-lcm:6405/biprws/infostore/3902"},
         "id":3902,
         "cuid":"AVH4dKB.OpJBoK6b.fAUmjA",
         "name":"Analysis edition for OLAP",
         "type":"Pioneer"
      }
   ]
}
```

3. Feed sample with filter: type=InfoView

XML format:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>tag:sap.com,2010:bip-rs/AWItAeqx.FpBgqTpFH8LqwE/children</id>
  <title type="text">Children of Root Folder 99</title>
  <updated>2011-12-22T21:04:16.684Z</updated>
  <link href="http://commandcom-lcm:6405/biprws/infostore/99/children?page=1&amp;pageSize=50"
rel="self"></link>
  <link href="http://commandcom-lcm:6405/biprws/infostore/99/children?page=1&amp;pageSize=50"
rel="first"></link>
  <link href="http://commandcom-lcm:6405/biprws/infostore/99/children?page=1&amp;pageSize=50"
rel="last"></link>
  <entry>
    <title type="text">BI launch pad</title>
    <id>tag:sap.com,2010:bip-rs/Ac7UIwmYafpFuhiiw6FRXLQ</id>
    <author>
      <name>System Account</name>
    </author>
    <link href="http://commandcom-lcm:6405/biprws/infostore/476" rel="alternate"></link>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id" type="int32">476</attr>
        <attr name="cuid" type="string">Ac7UIwmYafpFuhiiw6FRXLQ</attr>
        <attr name="description" type="string" null="true"></attr>
        <attr name="name" type="string">BI launch pad</attr>
        <attr name="type" type="string">InfoView</attr>
      </attrs>
    </content>
  </entry>
</feed>
```

JSON format:

```
{
   "__metadata":
     {"uri":"http://commandcom-lcm:6405/biprws/infostore/99/children?page=1&amp;pageSize=50"},
   "first":
      {"__deferred":
         {"uri":"http://commandcom-lcm:6405/biprws/infostore/99/children?page=1&amp;pageSize=50"}
      },
   "last":
      {"__deferred":
         {"uri":"http://commandcom-lcm:6405/biprws/infostore/99/children?page=1&amp;pageSize=50"}
      },
   "entries":
   [
      {"__metadata":
         {"uri":"http://commandcom-lcm:6405/biprws/infostore/476"},
         "id":476,
         "cuid":"Ac7UIwmYafpFuhiiw6FRXLQ",
         "name":"BI launch pad",
         "type":"InfoView"
      }"
   ]
}
```

4. Attributed relationship entry sample: /12/relationships/receivedAlerts/5432

XML format:

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <title type="text">5432</title>
  <id>tag:sap.com,2010:bip-rs/AfRWaT5_131NlLLf5bRMLKY/relationships/receivedAlerts/5432</id>
  <author>
    <name>Administrator</name>
    <uri>http://commandcom-lcm:6405/biprws/infostore/12</uri>
  </author>
  <link href="http://commandcom-lcm:6405/biprws/infostore/5432" rel="related"></link>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="id" type="int32">5432</attr>
      <attr name="markedAsRead" type="bool">false</attr>
    </attrs>
  </content>
</entry>
```

JSON format:

```
{
   "related":
   {
      "__deferred":
        {"uri":"http://commandcom-lcm:6405/biprws/infostore/5432"}
   },
   "id":5432,
   "markedAsRead":false
}
```

6.2 /logon/long

Log on to the BI platform with a username and password by making requests to the `/logon/long` URL.

- Use the `GET` method to retrieve an XML template for the request body.
- Use the `POST` method to log on to the BI platform and obtain a logon token.

GET http://<baseURL>/logon/long

Make a `GET` request to `/logon/long` to receive a template that can be used in the request body of a `POST` request to the same URL.

Request:

- Method: GET
- URL: `http://<baseURL>/logon/long`

  Replace `<baseURL>` with the base URL for RESTful web service requests.
- Header:

Name	Value
`Accept`	`application/xml`

- Body: none

Response:

- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Content-Length	Length of content in the response body.

- Body: An XML template that can be used to populate the request body of the `POST` request.

```
<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>
```

POST http://<baseURL>/logon/long

To receive a logon token, make a `POST` request to `/logon/long`, providing your user name and password.

Request:

- Method: `POST`
- URL: `http://<baseURL>/logon/long`

  Replace `<baseURL>` with the base URL for RESTful web service requests.
- Header:

Name	Value
`Content-Type`	`application/xml`
`Accept`	`application/xml`

- Body:

```
<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>
```

- Use `<attr name="userName" type="string"></attr>` to define the user name.
- Use `<attr name="password" type="string"></attr>` to define the password.
- Use `<attr name="auth" type="string"></attr>` to define the type of authentication. Use one of `secEnterprise`, `secLDAP`, `secWinAD`, or `secSAPR3`.

Response:

- Header:

Attribute	Value
Status Code	HTTP response code.
Server	Type of server.
X-SAP-LogonToken	A logon token.
Date	Date and time of the response
Content-Type	Type of content in the response body.
Content-Length	Length of content in the response body.

  The `X-SAP-LogonToken` attribute contains the logon token.

```
X-SAP-LogonToken:"COMMANDCOM-LCM:6400@{3&2=5595,U3&p=40674.9596531551,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=40674.9596531551,Y7&4F=12,U3&63=secEnterprise,0P&66=60,03&68=secEnter
prise:Administrator,0P&qe=100,U3&vz=SFY6agrLPxpfQBK1ZKYCwoBZKCbfsQm7VgWZFiH.RhM,UP
```

- Body:

  The response body contains a copy of the logon token in the `<attr name="logonToken" type="string">` element. The logon token must be converted from its XML-encoded format to its original format before it can be used. For example, replace the `&amp;` character sequence with the `&` character.

```
<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=5604,U3&amp;p=40623.9456463889,Y7&amp;4F=12,U3&amp;63=secEnterprise,0P&amp;68=secEnter
prise:Administrator,0P&amp;qe=100,U3&amp;vz=g5KUU8cAA.d_ARmSDnBy6T7jJVNyFCTso4s0q3dI.4k,UP}</attr>
    </attrs>
  </content>
</entry>
```

This example shows the returned logon token in the response body:

```
COMMANDCOM-LCM:6400@{3&amp;2=5604,U3&amp;&amp;p=40623.9456463889,Y7&amp;&amp;4F=12,U3&amp;&amp;63=secEn
terprise,0P&amp;&amp;68=secEnterprise:Administrator,0P&amp;&amp;qe=100,U3&amp;&amp;vz=g5KUU8cAA.d_ARmSDn
By6T7jJVNyFCTso4s0q3dI.4k,UP}
```

To use this logon token, convert it to its original format:

```
COMMANDCOM-LCM:6400@{3&2=5604,U3&p=40623.9456463889,Y7&4F=12,U3&63=secEnterprise,0P&68=secEnterprise:Ad
ministrator,0P&qe=100,U3&vz=g5KUU8cAA.d_ARmSDnBy6T7jJVNyFCTso4s0q3dI.4k,UP}
```

Related Topics

- Converting a logon token from XML-encoded text
- To get a logon token from a user name and password

6.3 /logon/token

Log on to the BI platform with a serialized session or session token obtained from an existing serialized session by making requests to the `/logon/token` URL.

- Use the `GET` method to retrieve an XML template for the request body.
- Use the `POST` method to log on to the BI platform and obtain a logon token.

GET http://<baseURL>/logon/token

Make a `GET` request to `/logon/token` to receive a template that can be used in the request body of a `POST` request to the same URL.

Request:

- Method: GET
- URL: `http://<baseURL>/logon/token`

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

- Header:

Name	Value
`Accept`	`application/xml`

- Body: none

Response:

- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Content-Length	Length of content in the response body.

- Body: An XML template that can be used to populate the request body of the `POST` request.

```
<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>
```

POST http://<baseURL>/logon/token

To receive a logon token, make a `POST` request to `/logon/token`, providing a serialized session or session token obtained from another SDK.

Request:

- Method: `POST`
- URL: `http://<baseURL>/logon/token`.

  Replace `<baseURL>` with the base URL for RESTful web service requests.
- Header:

Name	Value
`Content-Type`	`application/xml`
`Accept`	`application/xml`

- Body:

```
<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_131NlLLf5bRM
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>
```

  - Use `<attr name="tokenType" type="string" possibilities="token, serial izedSession">` to define the type of token.

    Use `token` if you are providing a session token. Use `serializedSession` if you are providing a serialized session.
  - Use `<attr name="logonToken" type="string">` to define the serialized session or session token value.

Note:
The serialized session or session token value must be XML-encoded to remove illegal XML characters. For example, replace the `&` character with `&amp;`.

Response:

- Header:

Attribute	Value
Status code	HTTP response code.
Server	Type of server.
X-SAP-LogonToken	A logon token.
Content-Type	Type of content in the response body.
Content-Length	Length of content in the response body.

  The `X-SAP-LogonToken` attribute contains the logon token. The logon token is contained between the quotation marks.

```
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"
```

- Body:

  The response body contains an XML-encoded copy of the logon token in the `<attr>` element. The logon token must be converted from its XML-encoded format to its original format. For example, replace the `&amp;` character sequence with the `&` character.

```
<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>
```

  This example shows the returned logon token in the response body:

```
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
```

  To use this logon token, convert it to its original format:

```
COMMANDCOM-LCM:6400@{3&2=5319,U3&p=40722.7462034491,Y7&4F=12,U3&63=secEnterprise,0P&66=60,03&68=secEnter
prise:Administrator,0P&qe=100,U3&vz=KeDu7064jWSptBT_m5BkBJ5Q_NaxyvE_WStqXmigYrg,UP
```

Related Topics

- Converting a logon token from XML-encoded text

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

6.4 /logon/adsso

The `/logon/adsso` (Active Directory Single Sign On - ADSSO) is used to acquire tokens from Active Directory user accounts. The BOE server must have `web.xml` configured for ADSSO and users' Windows Active Directory login name must match their BOE account name.

- Use the `GET` method to retrieve the logon token.

GET http://<baseURL>/logon/adsso

Make a `GET` request to `/logon/adsso` to receive a logon token.

Request:

- Method: GET
- URL: `http://<baseURL>/logon/adsso`

  Replace `<baseURL>` with the base URL for RESTful web service requests.
- Header:

Name	Value	Note
`Accept`	`application/xml`	Used by default, so use of this header is not necessary.

- Request Body: None
- Response Header

Name	Value	Example
Status Code	HTTP response code.	200 OK
Server	Type of server.	Apache-Coyote/1.1
X-SAP-LogonToken	Returned encoded token.	COMMANDCOM-LCM:6400@{3&2=588...9QO0XnE,UP}
Date	Date and time of response.	Fri, 16 Dec 2011 22:00:57 GMT
Content-Type	Type of content in the response body.	application/xml
Content-Length	Length of content in the response body.	6919

Response Body:

```
<?xml version="1.0" ?>
<entry xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>
      @VMBOESRVR.ADDOM.COM
    </name>
  </author>
  <id>
    tag:sap.com,2010:bip-rs/logon/adsso
  </id>
  <title type="text">
    Logon Result
  </title>
  <updated>
    2011-11-21T22:15:51.340Z
  </updated>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="logonToken" type="string">
        VMBOESRVR.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%3D2K8ADDOMAIN
        %2CDC%3DCOM,0P&qe=100,U3&vz=kOox8TDqAiFsfs8T3GefI3sWXIyKymc9qvytAjihC7w,UP}
      </attr>
    </attrs>
  </content>
</entry>
```

Note:
Internet Explorer can be used to retrieve an Active Directory single sign on `logonToken` by entering `http://<baseURL>/logon/adsso`. However, the returned value includes <name>, <id> and <updated> strings, data that is not part of a valid `logonToken`. The following text clipping shows irrelevant data that is prefixed to a `logonToken` request obtained with Internet Explorer.

```
@VMBOESRVR.ADDOM.COMtag:sap.com,2010:bip-rs/logon/adsso2011-11-21T19:02:00.761Z
```

The following text clipping shows a valid `logonToken` without extraneous data.

```
VMBOESRVR.AD
DOM.COM:6400@{3&2=4584,U3&p=40868.9276775116,Y7&4F=4331,U3&63=secWinAD,0P&66=60,03&68=secWinAD:CN%3DADUs
er1%2CCN%3DUsers%2CDC%3D2K8ADDOMAIN%2CDC%3DCOM,0P&qe=100,U3&vz=kOox8TDqAiFsfs8T3GefI3sWXIyKymc9qvytAjihC7w,UP}
```

Related Topics

- Converting a logon token from XML-encoded text
- To configure web.xml to enable WinAD SSO

6.5 /logon/trusted

GET http://<baseURL>/logon/trusted

Used to retrieve a logon token using a trusted authenticated user name by sending a `GET` request to `/logon/trusted` by one of three retrieval methods: an HTTP header request, an HTTP-encoded URL query, or a cookie.

The following retrieving methods are available in CMC: **Servers > Servers List > WACS > "Trusted Authentication Configuration"**.

Retrieving Method
`HTTP_HEADER`
`QUERY_STRING`
`COOKIE`

The "User Name" parameter can be changed in CMC: **Servers > Core Services > WACS > Users Name Parameter**.

User Name Paramater	String value restrictions
`X-SAP-TRUSTED-USER`	Default setting. Cannot contain spaces or a colon (`:`)

Note:
The WACS service does not need a restart between changes to the **Retrieving Method** or **User Name Parameter**. Query String URLs must be HTTP encoded. In generally, characters such as spaces and colons must not be used within values or name parameters.

1. Request using `HTTP_HEADER`:

- Method: `GET`
- URL: `http://<baseURL>/logon/trusted`

  Replace `<baseURL>` with the base URL for RESTful web service requests.
- Header: none

2. Request using `QUERY_STRING`:

- Method: `GET`
- URL: `http://<baseURL>/logon/trusted?<X-SAP-TRUSTED-USER>=<trustedUserName>`

  Replace `<baseURL>` with the base URL for RESTful web service requests. The default label is `<X-SAP-TRUSTED-USER>`. This label can be changed to another value in CMC: **Servers > WACS**, "Trusted Authentication Configuration", **User Name Parameter**. Replace `<trustedUserName>` with the name of a trusted user account name as defined in **CMC > Users and Groups**.

- Header: none
- Restriction: URL must be HTTP encoded

3. Request using `COOKIE`:

- Method: `GET`
- URL: `http://<baseURL>/logon/trusted`

  Replace `<baseURL>` with the base URL for RESTful web service requests.
- Header: none
- Cookie values:

Cookie catego-ry	Example value	Note
`Domain`	www.sap.com	This value is used as the server address in `<server address>`
Name	`X-SAP-TRUSTED-USER`	This is the default label. It may be changed in **CMC > Servers > WACS**, "Trusted Authentication Configuration", **User Name Parameter**
Value	`bob`	The name of the trusted user as defined in **CMC > Users and Groups**
Path	/	Path local to the <server address>. Usually this is a forward slash (/).

- Response Header:

Name	Value	Example
Status Code	HTTP response code.	200 OK
Server	Type of server.	Apache-Coyote/1.1
X-SAP-Lo-gonToken	Returned encoded token.	COMMANDCOM-LCM:6400@{3&amp;2=5613,U3&amp;p=40884.81...RrzfQ,UP}
Date	Date and time of response.	Wed, 07 Dec 2011 19:29:49 GMT
Content-Type	Type of content in the re-sponse body.	text/html
Content-Length	Length of content in the re-sponse body.	577

Response Body example:

```
<?xml version="1.0" ?>
<entry xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>
      @COMMANDCOM-LCM:6400
    </name>
  </author>
  <id>
    tag:sap.com,2010:bip-rs/logon/trusted
  </id>
  <title type="text">
    Logon Result
  </title>
  <updated>
    2011-12-07T21:46:57.091Z
  </updated>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="logonToken" type="string">
        COMMANDCOM-LCM:6400@{3&2=5652,U3&p=40884.90760...GwVVaCm.xJ.OtXrTB6n9TuzNfE,UP}
      </attr>
    </attrs>
  </content>
</entry>
```

Related Topics

- Converting a logon token from XML-encoded text
- To get a logon token from a serialized session or session token

6.6 /logoff

POST http://<baseURL>/logoff

Make a `POST` request to `/logoff` to invalidate the logon token and log off the BI platform.

Request:

- Method: POST
- URL: `http://<baseURL>/logoff`

  Replace `<baseURL>` with the base URL for RESTful web service requests.
- Header:

Name	Value
`Accept`	`application/xml`
`X-SAP-LogonToken`	The logon token value, in quotation marks.

- Body: none

Response:

- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Content-Length	Length of content in the response body.

- Body: none

Related Topics

- To log off the BI platform

6.7 /infostore

GET http://<baseURL>/infostore

Make a `GET` request to `/infostore` to retrieve the contents of the top level of the BI platform repository.

Request:

- Method: `GET`
- URL: `http://<baseURL>/infostore`

  Replace `<baseURL>` with the base URL for RESTful web service requests.
- Header:

Name	Value
`Accept`	`application/xml`
`X-SAP-LogonToken`	The logon token value, in quotation marks.

- Body: none

Response:

- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Transfer-Encoding	Type of transfer encoding used to transmit the response.

- Body:

  The response is a feed that contains entries for each top-level resource in the BI platform repository.

```
<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>2011-03-31T23:55:10.852Z</updated>
  <link href="http://localhost:6405/biprws/infostore/4/children?page=1&amp;pageSize=50" rel="self"></link>

  <link href="http://localhost:6405/biprws/infostore/4/children?page=1&amp;pageSize=50" rel="first"></link>

  <link href="http://localhost:6405/biprws/infostore/4/children?page=1&amp;pageSize=50" rel="last"></link>

  <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"></link>
    <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" null="true"></attr>
        <attr name="type" type="string">Folder</attr>
      </attrs>
    </content>
  </entry>
  <entry>
    <title type="text">Application Folder</title>
    <id>tag:sap.com,2010:bip-rs/AdoctK9h1sBHp3I6uG0Sh7M</id>
    <author><name>System Account</name></author>
    <link href="http://localhost:6405/biprws/infostore/Application%20Folder" rel="alternate"></link>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id" type="int32">43</attr>
        <attr name="cuid" type="string">AdoctK9h1sBHp3I6uG0Sh7M</attr>
        <attr name="description" type="string"></attr>
        <attr name="type" type="string">Folder</attr>
      </attrs>
    </content>
  </entry>
...
</feed>
```

Related Topics

- To view the top level of the BI platform repository

6.8 /infostore/<id>

You can retrieve a resource from the BI platform repository by looking it up by its ID. The ID corresponds to the `SI_ID` property of the object.

Request:

- Method: `GET`
- URL: `http://<baseURL>/infostore/<id>`

  Replace `<baseURL>` with the base URL for RESTful web service requests, and replace `<id>` with the ID of the resource.
- Header:

Name	Value
`Accept`	`application/xml`
`X-SAP-LogonToken`	The logon token value, in quotation marks.

- Body: none

Response:

- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Content-Location	A link to the resource.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Content-Length	Length of content in the response body.

- Body: This example shows the response for the resource with ID = `99`, using the request`http://<baseURL>/infostore/99`.

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <author><name>System Account</name></author>
  <id>tag:sap.com,2010:bip-rs/AWItAeqx.FpBgqTpFH8LqwE</id>
  <title type="text">Root Folder 99</title>
  <updated>2011-04-14T10:27:50.969Z</updated>
  <link href="http://localhost:6405/biprws/infostore/99/children" rel="http://www.sap.com/rws/bip#chil
dren"></link>
  <link href="http://localhost:6405/biprws/infostore/Application%20Folder" rel="up"></link>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="id" type="int32">99</attr>
      <attr name="description" type="string" null="true"></attr>
     <attr name="cuid" type="string">AWItAeqx.FpBgqTpFH8LqwE</attr><attr name="description" type="string"
 null="true"></attr>
      <attr name="name" type="string">Root Folder 99</attr>
      <attr name="type" type="string">Folder</attr>
    </attrs>
  </content>
</entry>
```

Related Topics

- To retrieve an object by ID

6.9 /infostore/cuid_<cuid>

You can retrieve a resource from the BI platform repository by looking it up by its CUID. The CUID corresponds to the `SI_CUID` property of the object.

Request:

- Method: `GET`
- URL: `http://<baseURL>/infostore/cuid_<cuid>`

  Replace `<baseURL>` with the base URL for RESTful web service requests, and replace `<cuid>` with the CUID of the resource.
- Header:

Name	Value
`Accept`	`application/xml`
`X-SAP-LogonToken`	The logon token value, in quotation marks.

- Body: none

Response:

- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Content-Location	An alternate link to the resource using its ID.
Content-Length	Length of content in the response body.

- Body:

  This example shows the response for querying for the object with CUID = `AWItAeqx.FpBgqTpFH8LqwE`, using the request `http://<baseURL>/infostore/cuid_AWItAeqx.FpBgqTpFH8LqwE`.

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <author><name>System Account</name></author>
  <id>tag:sap.com,2010:bip-rs/AWItAeqx.FpBgqTpFH8LqwE</id>
  <title type="text">Root Folder 99</title>
  <updated>2011-04-14T10:27:50.969Z</updated>
  <link href="http://localhost:6405/biprws/infostore/99/children" rel="http://www.sap.com/rws/bip#chil
dren"></link>
  <link href="http://localhost:6405/biprws/infostore/Application%20Folder" rel="up"></link>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="id" type="int32">99</attr>
      <attr name="cuid" type="string">AWItAeqx.FpBgqTpFH8LqwE</attr>
      <attr name="description" type="string" null="true"></attr>
      <attr name="name" type="string">Root Folder 99</attr>
      <attr name="type" type="string">Folder</attr>
    </attrs>
  </content>
</entry>
```

Related Topics

- To retrieve an object by CUID

6.10 /infostore/<id>/children

You can retrieve the children of a resource by appending `/children` to the end of the RESTful web service request for the resource.

Request:

- Method: `GET`
- URL: `http://<baseURL>/infostore/<id>/children`

  Replace `<baseURL>` with the base URL for RESTful web service requests. Replace `<id>` with the ID or CUID of the resource.

- Header:

Name	Value
`Accept`	`application/xml`
`X-SAP-LogonToken`	The logon token value, in quotation marks.

- Body: none

Response:

- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Transfer-Encoding	Type of transfer encoding used to transmit the response.

- Body:

  This example shows the response for retrieving the children of the folder named Logical Groups (ID = 4079), using the request `http://localhost:6405/biprws/infostore/4079/children`.

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>tag:sap.com,2010:bip-rs/AeqYRbv26opLrzd92.uxtEE/children</id>
  <title type="text">Children of Logical Groups</title>
  <updated>2011-04-20T21:25:16.847Z</updated>
  <link href="http://localhost:6405/biprws/infostore/Logical%20Groups/children?page=1&amp;pageSize=50"
rel="self"></link>
  <link href="http://localhost:6405/biprws/infostore/Logical%20Groups/children?page=1&amp;pageSize=50"
rel="first"></link>
  <link href="http://localhost:6405/biprws/infostore/Logical%20Groups/children?page=1&amp;pageSize=50"
rel="last"></link>
  <entry>
    <title type="text">Applications</title>
    <id>tag:sap.com,2010:bip-rs/AWZ0V.pxFA5DmK5Wk0eZp5s</id>
    <author><name>System Account</name></author>
      <link href="http://localhost:6405/biprws/infostore/3976" rel="alternate"></link>
      <content type="application/xml">
        <attrs xmlns="http://www.sap.com/rws/bip">
          <attr name="id" type="int32">3976</attr>
          <attr name="cuid" type="string">AWZ0V.pxFA5DmK5Wk0eZp5s</attr>
          <attr name="description" type="string">Applications</attr>
          <attr name="type" type="string">LogicalGroup</attr>
        </attrs>
      </content>
  </entry>
...
  <entry><title type="text">Universe</title>
    <id>tag:sap.com,2010:bip-rs/Ac5u3HEfLbxMu70IL3UMsbY</id>
    <author><name>System Account</name></author>
    <link href="http://localhost:6405/biprws/infostore/3959" rel="alternate"></link>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id" type="int32">3959</attr>
        <attr name="cuid" type="string">Ac5u3HEfLbxMu70IL3UMsbY</attr>
        <attr name="description" type="string">Universes</attr>
        <attr name="type" type="string">LogicalGroup</attr>
      </attrs>
    </content>
  </entry>
  <entry>
    <title type="text">UserAndUserGroup</title><id>tag:sap.com,2010:bip-rs/AQFxiDd1HltNne32Pigt9qQ</id>
    <author><name>System Account</name></author>
    <link href="http://localhost:6405/biprws/infostore/4019" rel="alternate"></link>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id" type="int32">4019</attr>
        <attr name="cuid" type="string">AQFxiDd1HltNne32Pigt9qQ</attr>
        <attr name="description" type="string">Users and User Groups</attr>
        <attr name="type" type="string">LogicalGroup</attr>
      </attrs>
    </content>
  </entry>
</feed>
```

Related Topics

- To access child objects

6.11 /infostore/<id>/relationships

You can retrieve the relationships of a resource by appending `/relationships` to the end of the RESTful web service request for the resource.

Request:

- Method: `GET`
- URL: `http://<baseURL>/infostore/<id>/relationships/<type>/<another id>`

  Replace `<baseURL>` with the base URL for RESTful web service requests. Replace `<id>` and `<another id>` with the ID or CUID of the resource.
- Header:

Name	Value
`Accept`	`application/xml`
`X-SAP-LogonToken`	The logon token value, in quotation marks.

- Body: none

Response:

- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Transfer-Encoding	Type of transfer encoding used to transmit the response.

- Body:

  This XML example shows the response for retrieving the relationships of the folder named Logical Groups (ID = 5432), using the request `http://localhost:6405/biprws/infostore/12/relationships/receivedAlerts/5432`.

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <title type="text">5432</title>
  <id>tag:sap.com,2010:bip-rs/AfRWaT5_131N1LLf5bRMLKY/relationships/receivedAlerts/5432</id>
  <author>
    <name>Administrator</name>
    <uri>http://commandcom-lcm:6405/biprws/infostore/12</uri>
  </author>
  <link href="http://commandcom-lcm:6405/biprws/infostore/5432" rel="related"></link>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="id" type="int32">5432</attr>
      <attr name="markedAsRead" type="bool">false</attr>
    </attrs>
  </content>
</entry>
```

  JSON format:

```
{
   "related":{
      "__deferred":{
         "uri":"http://commandcom-lcm:6405/biprws/infostore/5432"
      }
   },
   "id":5432,
   "markedAsRead":false
}
```

Related Topics

- To access child objects

6.12 /scheduleForms

GET http://<baseURL>/infostore/<id>/scheduleForms

Get a list of URLs that can be used to schedule a resource by sending a request to `/scheduleForms` using the `GET` method.

Note:
If the resource is not schedulable, an error is returned.

Request:
- Method: GET
- URL: `http://<baseURL>/infostore/<id>/scheduleForms`

  Replace `<baseURL>` with the base URL for RESTful web service requests. Replace `<id>` with the ID or CUID of a schedulable resource.
- Header:

Name	Value
`Accept`	`application/xml`
`X-SAP-LogonToken`	The logon token value, in quotation marks.

- Body: none

Response:
- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Content-Length	Length of content in the response body.

- Body:

  An XML feed of scheduling URLs. This example shows the scheduling URLs for the resource with ID=5177.

```
<feed xmlns="http://www.w3.org/2005/Atom">
<author>
  <name>Administrator</name>
  <uri>http://localhost:6405/biprws/infostore/12</uri>
</author>
<id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/scheduleForms</id>
<title type="text">Schedule Drilldown</title><updated>2011-05-18T10:31:30.092Z</updated>
  <entry>
    <title type="text">now</title><id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/now</id>
    <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/now" rel="alternate"></link>
  </entry>
  <entry>
    <title type="text">once</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/once</id>
    <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/once" rel="alternate"></link>
  </entry>
  <entry>
    <title type="text">hourly</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/hourly</id>
   <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/hourly" rel="alternate"></link>

  </entry>
  <entry>
    <title type="text">daily</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/daily</id>
   <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/daily" rel="alternate"></link>

  </entry>
  <entry>
    <title type="text">weekly</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/weekly</id>
   <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/weekly" rel="alternate"></link>

  </entry>
  <entry>
    <title type="text">monthly</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/monthly</id>
   <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/monthly" rel="alternate"></link>

  </entry>
  <entry><title type="text">NthDayOfMonth</title>
    <id>tag:sap.com,2010:bip-rs/ASb6ObslHktFnk3uF8.g3tw/NthDayOfMonth</id>
    <link href="http://localhost:6405/biprws/infostore/5177/scheduleForms/NthDayOfMonth" rel="alter
nate"></link>
  </entry>
</feed>
```

Related Topics

• To discover the scheduling URLs for an object

6.13 /scheduleForms/<form>

Schedule a resource by making requests to the `/scheduleForms/<form>` URL. Replace `<form>` with the type of scheduling, for example, now, daily, weekly, monthly, or once.

- Use the `GET` method to retrieve an XML template for the request body.
- Use the `POST` method to schedule a resource.

GET http://<baseURL>/infostore/<id>/scheduleForms/<form>

Make a `GET` request to `/scheduleForms/<form>` to receive a template that can be used in the request body of a `POST` request to the same URL.

Request:

- Method: GET
- URL: `http://<baseURL>/infostore/<id>/scheduleForms/<form>`

  Replace `<baseURL>` with the base URL for RESTful web service requests. Replace `<id>` with the ID or CUID of a schedulable resource. Replace `<form>` with the scheduling frequency, for example now, daily, weekly, or once.
- Header:

Name	Value
`Accept`	`application/xml`
`X-SAP-LogonToken`	The logon token value, in quotation marks.

- Body: none

Response:

- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Content-Length	Length of content in the response body.

- Body: An XML template that can be used to populate the request body of the `POST` request.

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Administrator</name>
    <uri>http://localhost:6405/biprws/infostore/12</uri>
  </author>
  <id>tag:sap.com,2010:bip-rs/ASb6Obs1HktFnk3uF8.g3tw/scheduleForms/now</id>
  <title type="text">Schedule Drilldown now</title>
  <updated>2011-05-18T10:31:30.092Z</updated>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="retriesAllowed" type="int32">0</attr>
      <attr name="retryIntervalInSeconds" type="int32">1800</attr>
    </attrs>
  </content>
</entry>
```

POST http://<baseURL>/infostore/<id>/scheduleForms/<form>

Make a `POST` request to `/scheduleForms/<form>` to schedule a resource

Request:

- Method: `POST`
- URL: `http://<baseURL>/infostore/<id>/scheduleForms/<form>`.

  Replace `<baseURL>` with the base URL for RESTful web service requests. Replace `<id>` with the ID or CUID of a schedulable resource. Replace `<form>` with the scheduling method, for example, now, daily, weekly, or once.
- Header:

Name	Value
`Content-Type`	`application/xml`
`Accept`	`application/xml`
`X-SAP-LogonToken`	The logon token value, in quotation marks.

- Body:

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Administrator</name>
    <uri>http://localhost:6405/biprws/infostore/12</uri>
  </author>
  <id>tag:sap.com,2010:bip-rs/ASb6Obs1HktFnk3uF8.g3tw/scheduleForms/now</id>
  <title type="text">Schedule Drilldown now</title>
  <updated>2011-05-18T10:31:30.092Z</updated>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="retriesAllowed" type="int32">3</attr>
      <attr name="retryIntervalInSeconds" type="int32">1800</attr>
    </attrs>
  </content>
</entry>
```

  Fill in each `<attr>` element with an appropriate value for your scheduling request. For example, to allow three attempts to schedule the resource, set the `retriesAllowed` property of the `<attr>` element to `3`.

```
<attr name="everyNDays" type="int32"></attr>
```

Note:
The `<attr>` elements have different properties depending on the type of scheduling that has been chosen.

Response:

- Header:

Attribute	Value
Status Code	HTTP response code.
Server	Type of server.
Location	A URL that contains a link to the location of the scheduled instance.
Date	The date and time of the response.
Content-Type	Type of content in the response body.
Content-Length	Length of content in the response body.

- Body: none

Related Topics

- To schedule a resource

6.14 ?page=<n>&pageSize=<m>

You can limit the number of entries returned by a request by setting the page size and requesting a certain page. The response contains only the entries for that page, and includes links to URLs that can be used to retrieve other pages of information.

Request:

- Method: `GET`
- URL: `http://<baseURL>/infostore/<id>?page=<n>&pageSize=<m>` .

  Replace `<baseURL>` with the base URL for RESTful web service requests, replace `<id>` with the ID, or CUID of the resource you want to retrieve. Replace `<n>` with the number of the page you want to view, and replace `<m>` with the number of entries to return on each page.
- Header:

Name	Value
`Accept`	`application/xml`
`X-SAP-LogonToken`	The logon token value, in quotation marks.

- Body: none

Response:

- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Content-Length	Length of content in the response body.

- Body:

  This example shows the response for requesting the children of the resource with ID = 23. It requests the second page of entries, where there are three entries per page. The request is`http://<baseURL>/infostore/23/children?page=2&pageSize=3`. The response includes links to other pages of results.

```
<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>2011-04-07T23:50:17.983Z</updated>
<link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=2&amp;pageSize=3"
```

```
rel="self"></link>
<link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=1&amp;pageSize=3"
rel="first"></link>
<link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=3&amp;pageSize=3"
rel="next"></link>
<link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=1&amp;pageSize=3"
rel="previous"></link>
<link href="http://localhost:6405/biprws/infostore/Root%20Folder/children?page=3&amp;pageSize=3"
rel="last"></link>
<entry>
  <title type="text">Platform Search Scheduling</title>
  <id>tag:sap.com,2010:bip-rs/AfbVaQ1CdrNDkKlzAKEK3aI</id>
  <author><name>System Account</name></author>
  <link href="http://localhost:6405/biprws/infostore/4320" rel="alternate"></link>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="id" type="int32">4320</attr>
      <attr name="cuid" type="string">AfbVaQ1CdrNDkKlzAKEK3aI</attr>
      <attr name="description" type="string" null="true"></attr>
      <attr name="type" type="string">Folder</attr></attrs>
  </content>
</entry>
<entry>
  <title type="text">Probes</title>
  <id>tag:sap.com,2010:bip-rs/AYtU9ijcgpxFsbgLW0om5_U</id>
  <author><name>System Account</name></author>
  <link href="http://localhost:6405/biprws/infostore/4001" rel="alternate"></link>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="id" type="int32">4001</attr>
      <attr name="cuid" type="string">AYtU9ijcgpxFsbgLW0om5_U</attr>
      <attr name="description" type="string" null="true"></attr>
      <attr name="type" type="string">Folder</attr>
    </attrs>
  </content>
</entry>
<entry>
  <title type="text">Report Conversion Tool</title>
  <id>tag:sap.com,2010:bip-rs/AY9zJ8BgaF9OucZ2h2slcJM</id>
  <author><name>Administrator</name>
  <uri>http://localhost:6405/biprws/infostore/12</uri></author>
  <link href="http://localhost:6405/biprws/infostore/4082" rel="alternate"></link>
  <content type="application/xml">
    <attrs xmlns="http://www.sap.com/rws/bip">
      <attr name="id" type="int32">4082</attr>
      <attr name="cuid" type="string">AY9zJ8BgaF9OucZ2h2slcJM</attr>
      <attr name="description" type="string"></attr>
      <attr name="type" type="string">Folder</attr>
    </attrs>
  </content>
  </entry>
</feed>
```

Related Topics

- To use pagination with results

6.15 ?type=<type>

You can limit the type of results that are returned from a RESTful web service request by appending `?type=<type>` to the end of the RESTful web service request. The `<type>` attribute corresponds to the `SI_KIND` property of the object.

Request:

- Method: `GET`

- URL: `http://<baseURL>/infostore/<id>?type=<type>`

  Replace `<baseURL>` with the base URL for RESTful web service requests, replace `<id>` with the ID or CUID of the resource, and replace `<type>` with the type of entry to return.
- Header:

Name	Value
`Accept`	`application/xml`
`X-SAP-LogonToken`	The logon token value, in quotation marks.

- Body: none

Response:
- Header:

Name	Value
Status Code	HTTP response code.
Server	Type of server.
Date	Date and time of response.
Content-Type	Type of content in the response body.
Content-Length	Length of content in the response body.

- Body:

  This example shows the response for requesting the children of the resource with ID = 99 that are of type 'InfoView'. The request is`http://<baseURL>/infostore/99/children?type=InfoView`. The response includes links to other pages of results.

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>tag:sap.com,2010:bip-rs/AWItAeqx.FpBgqTpFH8LqwE/children</id>
  <title type="text">Children of Root Folder 99</title>
  <updated>2011-06-06T23:40:10.209Z</updated>
  <link href="http://localhost:6405/biprws/infostore/99/children?page=1&pageSize=50" rel="self"></link>
  <link href="http://localhost:6405/biprws/infostore/99/children?page=1&pageSize=50" rel="first"></link>

  <link href="http://localhost:6405/biprws/infostore/99/children?page=1&pageSize=50" rel="last"></link>
  <entry>
    <title type="text">BI launch pad</title>
    <id>tag:sap.com,2010:bip-rs/Ac7UIwmYafpFuhiiw6FRXLQ</id>
    <author><name>System Account</name></author>
    <link href="http://localhost:6405/biprws/infostore/474" rel="alternate"></link>
    <content type="application/xml">
      <attrs xmlns="http://www.sap.com/rws/bip">
        <attr name="id" type="string">474</attr>
        <attr name="cuid" type="string">Ac7UIwmYafpFuhiiw6FRXLQ</attr>
        <attr name="description" type="string" null="true"></attr>
        <attr name="type" type="string">InfoView</attr>
      </attrs>
    </content>
  </entry>
</feed>
```

Related Topics

- To filter results by type

Appendix

7.1 Appendix A - RWS error messages summary, categorized

The following table lists RESTful Web Services error codes. Bracketed items indicated as `{ * insert resource name here * }`, in this table are replaced by the relevant resource name or value in the error message.

For more information about RWS and any other BI platform error messages, see the BusinessObjects XI *Error Messages Explained* guide.

The following table summarizes the errors organized by category.

Category	<message>
WebApplicationMapper	
`RWS 00002`	`General server error.`
`RWS 00003`	`Client input error.`
NoAccessException	
`RWS 00004`	`Forbidden.`
RSPluginException	
`RWS 00006`	`Unable to create service. See server logs for details.`
`RWS 00007`	`Unknown error occurred while invoking service. See server logs for details.`
`RWS 00010`	`Resource not supported for the requested object.`
InvalidEntSessionException	
`RWS 00008`	`The HTTP header does not contain the X-SAP-LogonToken attribute.`
`RWS 00011`	`Invalid session token timeout value: { * insert resource name here * }.`
`RWS 00016`	`The server session is not available from the PJS service bean.`

Category	<message>
RWS 00076	Logon may not proceed because a session is already asso ciated with this request.
RWS 00079	Please validate your input.
NotFoundException	
RWS 00005	Not Found.
RWS 00009	Resource not found: { * insert resource name here * }.
RWS 00012	Info object with ID { * insert resource name here * } not found.
RWS 00015	No relationship named { * insert resource name here * }.
DuplicateException	
RWS 00013	Duplicate Object.
RWS 00051	A duplicate { * insert resource name here * } instance was created.
CodecException	
RWS 00017	Encode failure.
RWS 00018	{ * insert resource name here * } is NULL.
RWS 00019	Illegal Argument: { * insert resource name here * }.
RWS 00020	Cannot serialize value of type { * insert resource name here * }.
RWS 00021	Unterminated string.
RWS 00022	Malformed date: { * insert resource name here * }.
RWS 00023	Malformed time: { * insert resource name here * }.
RWS 00024	Malformed datetime: { * insert resource name here * }.
RWS 00025	Cannot deserialize value of type { * insert resource name here * }.
RWS 00026	Cannot get the attribute name. The name is either null or empty.
<reserved>	
RWS 00001	<reserved>
RWS 00014	<reserved>
RWS 00027	<reserved>
RWS 00028	<reserved>

Category	<message>
RWS 00029	<reserved>
RWS 00030	<reserved>
ModelException	
RWS 00031	Model error.
RWS 00032	No setter.
RWS 00033	Parameters must not be used with this getter command: { * insert resource name here * }.
RWS 00034	Setter must have exactly one parameter: { * insert re source name here * }.
RWS 00035	Setter { * insert resource name 1 here * } is not of the same type as getter { * insert resource name 2 here * }.
RWS 00036	Source: { * insert resource name 1 here * } + destina tion: { * insert resource name 2 here * }.
RWS 00037	Reference equality is not implemented.
RWS 00038	This use in hash-based collections is not implemented.
RWS 00039	Class { * insert resource name here * } is not a model class.
RWS 00040	Class { * insert resource name here * } is not a model class.
RWS 00041	Attribute ''{ * insert resource name 1 here * }'' cannot bind to two get (or set) methods: { * insert resource name 2 here * }, and { * insert resource name 3 here * }.)
RWS 00042	Model contains at least 1 write-only attribute. name: { * insert resource name 1 here * }, method: { * insert resource name 2 here * }.)
RWS 00043	No accessible constructor without parameters for class { * insert resource name here * }.)
RWS 00044	{ * insert resource name 1 here * } object is null for composition property { * insert resource name 2 here * }.
RWS 00045	Couldn't inject property ''{ * insert resource name 1 here * }'' to field { * insert resource name2 here * } of type { * insert resource name 3 here * }.

Category	<message>
RWS 00046	Property name already exists: { * insert resource name here * }.
RWS 00047	GUID must not contain the path separator '/'.
RWS 00048	No type for class { * insert resource name here * }.
RWS 00049	Empty filter.
RWS 00050	Filter may not use ''{ * insert resource name here * }'' in conjunction with any other filter strings.
RWS 00080	Cannot bind unknown attribute "{ * insert resource name 1 here * }'' to method ''{ * insert resource name 2 here * }".
WebApplicationException-Mapper	
RWS 00052	Bad request. (RWS00052) Corresponds with HTTP Response Code 400. This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616. Consult section 10.4 of RFC 2616 for more guidance on resolving this error. Applies to RWS 00052 to RWS 00075.
RWS 00053	Unauthorized. (RWS00053) Corresponds with HTTP Response Code 401.
RWS 00054	Payment required. (RWS00054) Corresponds with HTTP Response Code 402.
RWS 00055	Forbidden. (RWS00055) Corresponds with HTTP Response Code 403.
RWS 00056	Not found. (RWS00056) Corresponds with HTTP Response Code 404.
RWS 00057	Method not allowed. (RWS00057) Corresponds with HTTP Response Code 405.
RWS 00058	Not acceptable. (RWS00058) Corresponds with HTTP Response Code 406.
RWS 00059	Proxy authentication required. (RWS00059) Corresponds with HTTP Response Code 407.
RWS 00060	Request timeout. (RWS00060) Corresponds with HTTP Response Code 408.
RWS 00061	Conflict. (RWS00061) Corresponds with HTTP Response Code 409.
RWS 00062	Gone. (RWS00062) Corresponds with HTTP Response Code 410.
RWS 00063	Length required. (RWS00063) Corresponds with HTTP Response Code 411.

Category	<message>
`RWS 00064`	`Precondition failed. (RWS00064)` Corresponds with HTTP Response Code 412.
`RWS 00065`	`Request entity too large. (RWS00065)` Corresponds with HTTP Response Code 413.
`RWS 00066`	`Request-URI too long. (RWS00066)` Corresponds with HTTP Response Code 414.
`RWS 00067`	`Unsupported media type. (RWS00067)` Corresponds with HTTP Response Code 415.
`RWS 00068`	`Requested range not satisfiable. (RWS00068)` Corresponds with HTTP Response Code 416.
`RWS 00069`	`Expectation failed. (RWS00069)` Corresponds with HTTP Response Code 417.
`RWS 00070`	`Internal server error. (RWS00070)` Corresponds with HTTP Response Code 500.
`RWS 00071`	`Not implemented. (RWS00071)` Corresponds with HTTP Response Code 501.
`RWS 00072`	`Bad gateway. (RWS00072)` Corresponds with HTTP Response Code 502.
`RWS 00073`	`Service unavailable. (RWS00073)` Corresponds with HTTP Response Code 503.
`RWS 00074`	`Gateway timeout. (RWS00074)` Corresponds with HTTP Response Code 504.
`RWS 00075`	`HTTP version not supported. (RWS00075)` Corresponds with HTTP Response Code 505.
CredentialException	
`RWS 00077`	`The authentication scheme you have chosen is currently not supported.`
`RWS 00078`	`The credentials could not be decoded.`

7.2 RESTful Web Services (RWS) Error Messages

RESTful Web Services error messages include the following:

Range	Category
RWS 00002 - RWS 00010	RESTful Web Services
RWS 000011 - RWS 000026	RESTful Web Services
RWS 000031 - RWS 000051	RESTful Web Services
RWS 00052 - RWS 00075	RESTful Web Services
RWS 000076 - RWS 000079	RESTful Web Services

7.2.1 RWS 00002 - RWS 00010

General server error. (RWS 00002)

Cause

An unknown error occurred in the BIP RESTful Web Service.

Action

Please check the server logs for more details.

Client input error. (RWS 00003)

Cause

There is an unknown error in the input of the client provided to the BIP RESTful Web Service.

Action

Please consult the documentation for the resource you're trying to call to determine if your input was indeed valid.

Forbidden (RWS 00004)

Cause

This resource may not be accessed.

Action

Verify you have the right permissions to access the resource.

Not Found (RWS 00005)

Cause

The specific resource could not be found. Either the resource does not exist or you do not have the permissions to view it.

Action

Verify that the URL you used was correct. If you're trying to view an InfoObject, use the Central Management Console (CMC) to verify that you have the right to view that object.

Unable to create service. See server logs for details. (RWS 00006)

Cause

The BIP RESTful Web Service was unable to create the requested service.

Action

Examine the JavaDoc for Constructor.newInstance. Cross check the cause of this exception with the exceptions thrown by Constructor.newInstance.

Unknown error occurred while invoking service. See server logs for details. (RWS 00007)

Cause

The BIP RESTful Web Service encountered an unknown error while invoking the service.

Action

Check the log of the Web Application Server containing the BIP RESTful Web Service to see more details.

The HTTP header does not contain the X-SAP-LogonToken attribute. (RWS 00008)

Cause

Access to the requested resources requires you to have been authenticated.

Action

Please pass in the X-SAP-LogonToken in the request's header. You may generate one using the logon resource.

Resource not found: {0} (RWS 00009)

Cause

The specific resource could not be found. Either the resource does not exist or you do not have the permissions to view it.

Action

Verify that the URL you used was correct. If you're trying to view an InfoObject, use the Central Management Console (CMC) to verify that you have the right to view that object.

Resource not supported for the requested object. (RWS 00010)

Cause

You attempted to access a resource for an InfoObject which was not supported. For example, this exception would be thrown when you try to access the Crystal Reports service for a Folder.

Action

Don't call this method on unsupported objects. Only visit links that are valid.

7.2.2 RWS 00011 - RWS 00026

Invalid session token timeout value: {0}. (RWS 000011)

Cause

A logon token could not be created because of an invalid setting in the BIP RESTful Web Service.

Action

Please contact your administrator to set an appropriate session token timeout value for the BIP RESTful Web Service in the Central Management Console (CMC).

Info object with ID {0} not found. (RWS 000012)

Cause

The InfoObject could not be found. If it's suppose to exist, have you verified that you have the permissions to view it?

Action

Use the Central Management Console (CMC) to verify that the InfoObject exists and that you have the right to view it.

Duplicate Object (RWS 000013)

Cause

A duplicate object was detected.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

No relationship named {0}. (RWS 000015)

Cause

The relationship could not be found on the InfoObject.

Action

Verify that the URL used was one genereated by the BIP RESTful WebService by visiting the root object. If the URL is indeed valid, have you checked your permissions to verify that you have the appropriate rights to view the relationship?

The server session is not available from the PJS service bean. (RWS 000016)

Cause

The Adaptive Processing Server has not passed a server session to the BIP RESTful Web Service.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

Encode failure. (RWS 000017)

Cause

The BIP RESTful Web Service uses a codec to encode objects into a user-readable format (e.g., XML). Unfortunately, it looks an encoding error occurred.

Action

Please check the logs for more details about the parameter that caused this error. If the problem remains unclear, please contact SAP BusinessObjects support for assistance.

`{0} is NULL. (RWS 000018)`

Cause

The BIP RESTful Web Service uses a codec to encode objects into a user-readable format (e.g., XML). Unfortunately, during its execution, it couldn't reference to a value.

Action

Please check the logs for more details about the parameter that caused this error. If the problem remains unclear, please contact SAP BusinessObjects support for assistance.

`Illegal Argument: {0} (RWS 000019)`

Cause

The BIP RESTful Web Service uses a codec to encode objects into a user-readable format (e.g., XML). Unfortunately, during its execution, it detected an illegal argument.

Action

Please check the logs for more details about the parameter that caused this error. If the problem remains unclear, please contact SAP BusinessObjects support for assistance.

`Cannot serialize value of type {0}. (RWS 000020)`

Cause

The BIP RESTful Web Service uses a codec to encode objects into a user-readable format (e.g., XML). We were unable to serialize a value.

Action

Please check the logs for more details about the parameter that caused this error. If the problem remains unclear, please contact SAP BusinessObjects support for assistance.

`Unterminated string. (RWS 000021)`

Cause

The BIP RESTful Web Service uses a codec to encode objects into a user-readable format (e.g., XML). It encountered an unterminated string.

Action

Please check the logs for more details about the parameter that caused this error. If the problem remains unclear, please contact SAP BusinessObjects support for assistance.

Malformed date: {0}. (RWS 000022)

Cause

The BIP RESTful Web Service was unable to encode/decode the date passed into it.

Action

Please check the logs for more details about the parameter that caused this error. If the problem remains unclear, please contact SAP BusinessObjects support for assistance.

Malformed time: {0}. (RWS 000023)

Cause

The BIP RESTful Web Service was unable to encode/decode the time passed into it.

Action

Please check the logs for more details about the parameter that caused this error. If the problem remains unclear, please contact SAP BusinessObjects support for assistance.

Malformed datetime: {0}. (RWS 000024)

Cause

The BIP RESTful Web Service was unable to encode/decode the date time passed into it.

Action

Please make sure the date time is in a format recognized by the ATOM standard (RFC 4287). Check the logs for more details about the parameter that caused this error. If the problem remains unclear, please contact SAP BusinessObjects support for assistance.

Cannot deserialize value of type {0}. (RWS 000025)

Cause

The BIP RESTful Web Service uses a codec to encode objects into a user-readable format (e.g., XML). Unfortunately, it looks a decoding error occurred.

Action

Please check the logs for more details about the parameter that caused this error. If the problem remains unclear, please contact SAP BusinessObjects support for assistance.

`Cannot get the attribute name. The name is either null or empty. (RWS 000026)`

Cause

The BIP RESTful Web Service uses a codec to encode objects into a user-readable format (e.g., XML). While reading/writing the user-readable format, a parser error occurred.

Action

Please check the logs for more details about the parameter that caused this error. If the problem remains unclear, please contact SAP BusinessObjects support for assistance.

7.2.3 RWS 00031 - RWS 00051

`Model error. (RWS 000031)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`No setter. (RWS 000032)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`Getter must not have parameters: {0}. (RWS 000033)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`Setter must have exactly one parameter: {0}. (RWS 000034)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`Setter {0} is not of the same type as getter {1}. (RWS 000035)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`source: {0} + destination: {1}. (RWS 000036)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

Reference equality is not implemented. (RWS 000037)

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

This use in hash-based collections is not implemented. (RWS 000038)

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

Class {0} is not a model class. (RWS 000039)

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

property '{0}' cannot bind to two fields: {1}, and {2}. (RWS 000040)

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

Attribute '{0}' cannot bind to two get (or set) methods: {1}, and {2}. (RWS 000041)

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

Model contains at least 1 write-only attribute. name: {0}, method: {1}. (RWS 000042)

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

No accessible constructor without parameters for class {0}. (RWS 000043)

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

{0} object is null for composition property {1}. (RWS 000044)

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`Couldn't inject property '{0}' to field {1} of type {2}. (RWS 000045)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`Property name already exists: {0} (RWS 000046)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`GUID must not contain the path separator '/' (RWS 000047)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`No type for class {0} (RWS 000048)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`Empty filter. (RWS 000049)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`Filter may not use '{0}' in conjunction with any other filter strings. (RWS 000050)`

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

`A duplicate {0} instance was created. (RWS 000051)`

Cause

The BIP RESTful Web Service code has singleton objects to manage its daily operations. Strangely, a duplicate of a singleton object was created.

Action

This error should not be thrown in a customer environment. If you have verified that your installation is correct and hasn't been corrupted, please contact SAP BusinessObjects support for help resolving this issue.

7.2.4 RWS 00052 - RWS 00075

Bad request. (RWS 00052)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Unauthorized (RWS 00053)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Payment required. (RWS 00054)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Forbidden (RWS 00055)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Not found. (RWS 00056)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Method not allowed (RWS 00057)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Not acceptable, (RWS 00058)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Proxy authentication required. (RWS 00059)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Request timeout. (RWS 00060)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Conflict (RWS 00061)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Gone (RWS 00062)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Length required. (RWS 00063)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

`Length required. (RWS 00063)`

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

`Precondition failed. (RWS 00064)`

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

`Request entity too large. (RWS 00065)`

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

`Request-URI too long. (RWS 00066)`

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Unsupported media type. (RWS 00067)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Requested range not satisfiable. (RWS 00068)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Expectation failed. (RWS 00069)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.4 of RFC 2616 for more guidance on resolving this error.

Internal server error. (RWS 00070)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.5 of RFC 2616 for more guidance on resolving this error.

Not implemented. (RWS 00071)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.5 of RFC 2616 for more guidance on resolving this error.

Bad gateway. (RWS 00072)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.5 of RFC 2616 for more guidance on resolving this error.

Service unavailable. (RWS 00073)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.5 of RFC 2616 for more guidance on resolving this error.

Gateway timeout. (RWS 00074)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.5 of RFC 2616 for more guidance on resolving this error.

HTTP version not supported. (RWS 00075)

Cause

This is a generic error message thrown by the BIP RESTful Web Service under the circumstances dictated by RFC 2616.

Action

Please consult section 10.5 of RFC 2616 for more guidance on resolving this error.

7.2.5 RWS 00076 - RWS 00080

Logon may not proceed because a session is already associated with this request. (RWS 000076)

Cause

You attempted to log onto the BIP RESTful Web Service while a session has already been associated with the request.

Action

Don't pass in a session to the BIP RESTful Web Service when you use the Logon resource.

The authentication scheme you have chosen is currently not supported. (RWS 000077)

Cause

The selected authentication scheme you have chosen is not supported by the BI Platform RESTful Web Service.

Action

Either pass in the credentials using the X-SAP-LogonToken mechanism or use HTTP BASIC authentication (see RFC 2617).

The credentials could not be decoded. (RWS 000078)

Cause

The credentials passed into the BI Platform RESTful Web Service could not be decoded.

Action

Make sure credentials are encoded correctly before using them. If you're using HTTP BAISC authentication, make sure they're encoded in the format specified by RFC 2617.

Please validate your input. (RWS 000079)

Cause

Please make sure the content of your request is formatted correctly and contains all the necessary fields.

Action

Re-send the request after you've verified that that content of your request is formatted correctly. Typically, you may use GET to determine what format the request should be in. You may also check the documentation for this information as well.

Cannot bind unknown attribute {0} to method {1}. (RWS00080)

Cause

The BIP RESTful Web Service contains invalid data in its binaries.

Action

Because this error message is rare in a fully installed environment, it may indicate a faulty or corrupted installation. If you have verified that your installation is correct and hasn't been corrupted, contact SAP BusinessObjects support for help resolving this issue.

More Information

Information Resource	Location
SAP product information	http://www.sap.com
SAP Help Portal	http://help.sap.com/analytics Access the most up-to-date English documentation covering all SAP Analytics products at the SAP Help Portal: • http://help.sap.com/bobi (BusinessObjects Business Intelligence) • http://help.sap.com/boepm (Enterprise Performance Management) • http://help.sap.com/boeim (Enterprise Information Management) Certain guides linked to from the SAP Help Portal are stored on 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. To find a comprehensive list of product documentation in all supported languages, visit:http://help.sap.com/boall.
SAP Support Portal	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.
Developer resources	http://www.sdn.sap.com/irj/sdn/bi-sdk-dev https://www.sdn.sap.com/irj/sdn/businessobjects-sdklibrary (BI SDK Developer Library)
Articles and eLearning on the SAP Community Network	http://scn.sap.com/docs/DOC-19311 These articles were formerly known as technical papers.

Information Resource	Location
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.
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