BIS

Web Services

Usage and Administration

S4IT - BIS Web Services Usage and Administration
V1.1 - English
Dec 2009

Published by Solutions for Informations Technologies

Contents

Introduction

Goal

The BIS Web Services exposes an open and standard interface to the BIS ROLAP engine. This way any application that acts as a Web Service Client can access the authorized BIS data layer.

In this manual will be explained:

1) How an Executive Report can become a Web Service and how the security is implemented;

2) How users with Navigator or Higher authorization level can run Ad-Hoc Web Service queries;

3) Usage of the BIS Web Services Interface by a Sample Web Service Client.

Advantages

The main advantage is that the developers are now allowed to create their own user interface layer maintaining the access to the already predefined star schemas and reports. The BIS Web Services allows external access to the BIS star schema maintaining the security rules defined at BIS metadata level.

There are two distinct ways of getting data through the BIS Web Services interface:

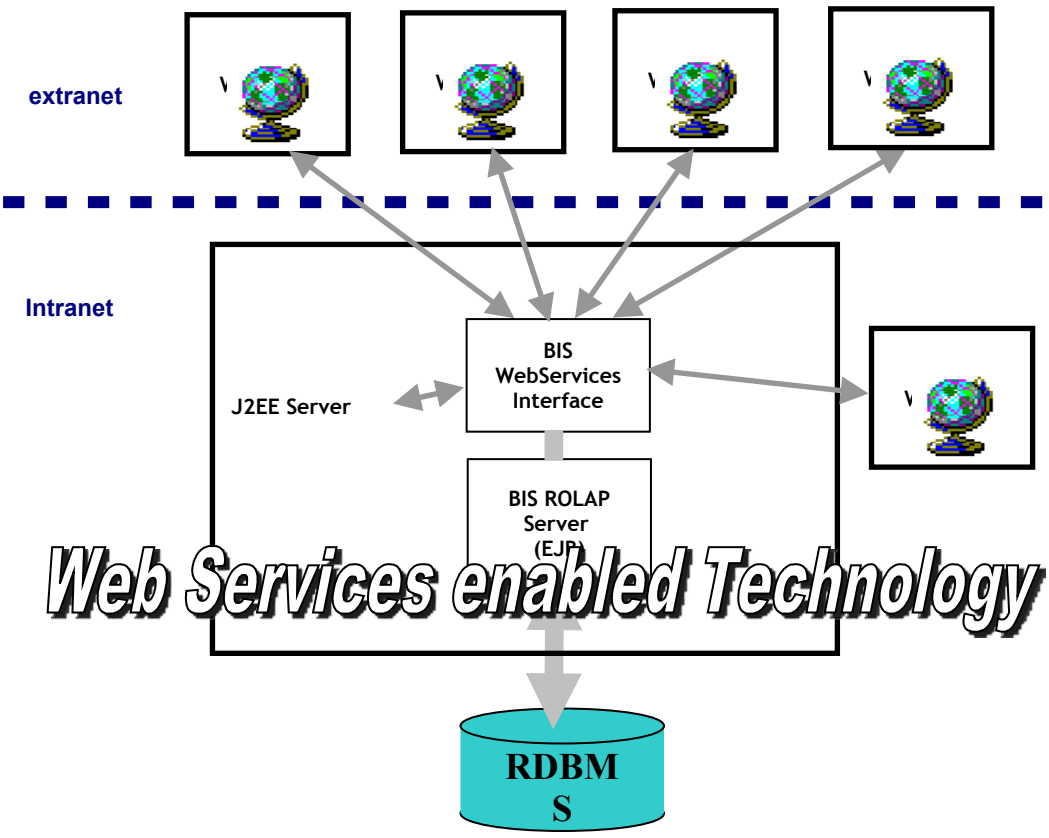
- Using the Pre-defined reports
- Executing Ad-Hoc queries, using the BIS ROLAP QUERY SYNTAX. This functionality is limited to BIS-Navigators only.

The security is enforced using a UserNameToken-based authentication between a web service client and a web service provider. This allows the federation of services. This way both the Web Services provider and the consumer may run in distinct domains.

Architecture

The BIS Web Services are implemented using the Java EE Platform. The Java EE platform uses a distributed multi-tiered application model for enterprise applications. Application logic is divided into components according to function, and the various application components that make up a Java EE application are installed on different machines depending on the tier in the multi-tiered Java EE environment to which the application component belongs. To expose Web Services it was used the JAX-WS API. JAX-WS stands for Java API for XML Web Services. JAX-WS is a technology for building web services and clients that communicate using XML. JAX-WS allows developers to write message-oriented as well as RPC-oriented web services.

With JAX-WS, clients and web services have a big advantage: the platform independence of the Java programming language. In addition, JAX-WS is not restrictive: a JAX-WS client can access a web service that is not running on the Java platform, and vice versa. This flexibility is possible because JAX-WS uses technologies defined by the World Wide Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.



As the picture above shows the BIS Web Services (WS) exposes an interface that can be consumed by Web Services Clients running in the same J2EE Server; in another J2EE Server or .NET server in the same hardware platform or even by WS Client applications running on other servers in an intranet or extranet environment. The BIS engine accesses the RDBMS through a JDBC connection pool, defined at J2EE server level.

The first version of BIS Web Services is available in a pre-deployed SUN Glassfish v2 J2EE Application Server.

Deployment

The access to BIS data through Web Services is an easy task. The process is to have an administrator, through the standard BIS-Admin tool, locating a Report in the Executive Reports tree and indicate that it is accessible through a Web Service. By default the name of the Web Service is the same of the report but it can be changed.

Steps to turn an Executive Report exposed as a Web Service

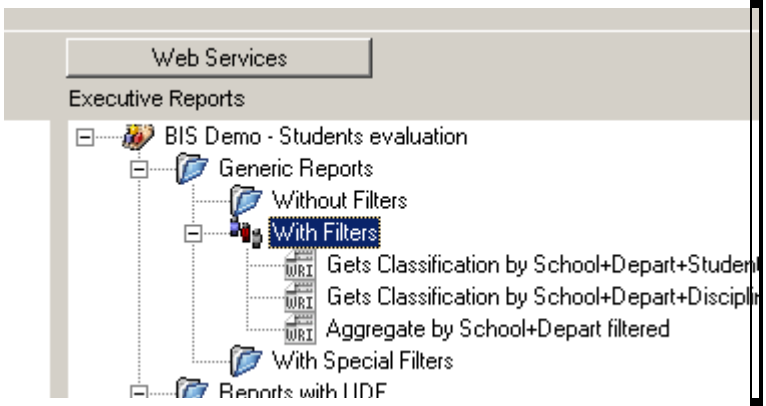
Define a New Web Service

1. Login in the BIS-Admin tool

2. Navigate to the Executive Reports

3. Choose your report to expose as a Web Service.

   As an example we will use the report "**Aggregate by School+Depart filtered**" in the folder "With Filters".



4. Click with the right button of the mouse in the chosen report, in order to get a pop-up menu containing several options.

5. Click the **Web Service** option on the pop-up menu.

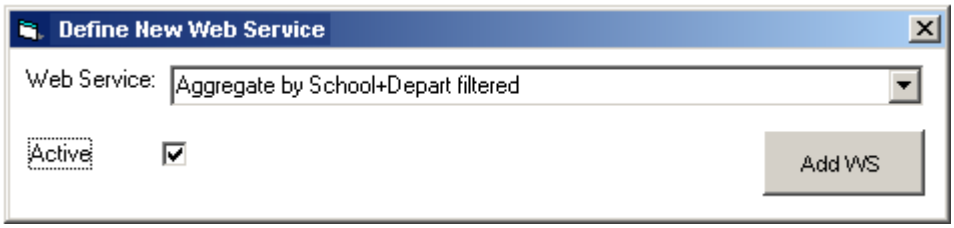
6. A form will be presented where we define:

a. The name of the Web Service (by default is the same as the the report). Some changes may be required. If the Web Service name will be used as an Argument passed through a http url the plus sign must be dropped from the name, due to is special http meaning.

b. In order for the report be accessible through the web service interface is necessary to check the **Active flag**.

c. Press the button **"Add WS"**

At this point the Web Service is available through the BIS Web Services interface !



The reports exposed as Web Services are recognized by the colors of the report name in the Executive tree:

a) **Blue** name: Report is exposed as an **Active Web Service**

b) **Red** name**:** Report is exposed as a **Web Service,** but is **inactive.**

Update the Web Service

It is possible to change the name and the active status of a previously defined web service.

In order to change these attributes execute the following steps:

1. Choose your report that is exposed as a Web Service in the Executive Reports tree.

   As an example we will use the report "**Aggregate by School+Depart filtered**" in the folder "With Filters".

2. Click with the right button of the mouse in the chosen report, in order to get a pop-up menu containing several options.

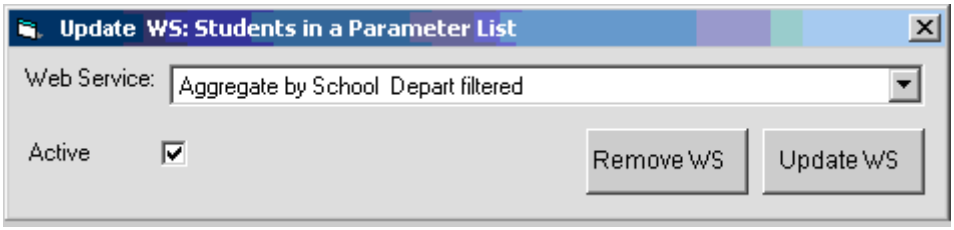
3. Click the **Web Service** option on the pop-up menu.

   The Update/Remove Web Service form will pop-up.

   It is also possible to access this from from the list of Web Services, by double clicking on the web service client row.

4. Change Name and/or the Active Status

5. Press the **Update WS** button to confirm the changes.



Remove a Web Service

To remove a web service definitions just press the Remove WS button from the above form.

List of Web Services

To list the defined Web Services just press the button  on the top of the Executive Reports tree. To switch again to the Executive Reports tree view, just press again the same button.

The list of Web Services, shows the Web Service name, the Active status and the name of the Executive Report associated with the Web Service.

Web Services

Web Service	Active	Report
Aggregate by School_Depart filtered	No	Aggregate by School+Depart filtered
Aggregate Query by School_Deparment	Yes	Aggregate Query by School+Deparment
Classification with a Filter Relative to the Schema Date	Yes	Classification with a Filter Relative to the Schema Date

You can update

Security of the BIS Web Services

The security in terms of who can access a specific web service is the same as the one implemented to the Executive Reports plus the Active flag control of the Web Service.

So, if a report is accessible to the users that belong to the groups with permissions to run the report, than the Web Service that exposes a report will also return data to the same group of users only.

It is the responsibility of the Web Service Client to pass to the BIS Web Service provider the correct user's credentials.

Only a trusted Web Service Client that provides a specific token and encrypted security code is authorized to access the BIS Web Service interface. The token and security code are provided to the Web Service Client developers by the BIS Web Services Administrator.

Sample Web Service Client

In this section we will present a sample jsp page developed for a SUN Glassfish Application Server that uses a Web Service Client using the BIS Web Service WSDL, but that can be implemented with minor changes or no changes at all in other Application Servers like IBM WebSphere or Oracle Web Logic, by a developer with experience in implementing a Web Service Client based on the JAX-WS 2.1 API specification.

The WSDL of the BIS Web Service definition is accessible by default through the URL **http://**s4wsHostName**:8484/s4WSsrv/S4ServerWSService?wsdl**

A similar sample JSP is implemented in the pre-deployed Web Service Client with the root Context "**S4ITWSCLI**"

Below are 3 samples on how to invoke the Client Web Service to execute pre-defined reports (**s4Report**) or an ad-hoc query (**s4SQL**).

The arguments are not posicional and have the following purposes:

s4User: The BIS User Name. This matches the login name of BIS user with Navigator or Executive privilegies.

s4SQL: A S4 Ad-Hoc Select statement. It is only allowed for Navigator users.

s4Report: Name of a Web Service mapped to an executive report.

Par: A list of parameters to be passed to the question filters implemented in the report. The number of parameters must exactly match the number and data type of the question filters of the report.

1) http:// s4wsHostName:8484/S4ITWSCLI/?**s4Report**=Relative Filter Fact Date&**s4User**=executive_user_name1

This report doesn't have any question filter.

2) http:// s4wsHostName:8484/S4ITWSCLI/?**s4Report**=Relative to System Date and SEX Stud Num&**Par**='M'&**Par**=24&**Par**=1 AND 100&**s4User**=executive_user_name2

This report has 3 question filters. The first is a string. The second is a number; The third is a between number values condition.

3) http:// s4wsHostName:8484/S4ITWSCLI/?s4User=navigator_user_name&**s4SQL=SELECT DIMENSIONS** [Disciplines Dimension].[teacher_name], [Schools Dimension].[ZipCode], **FACTS** [BIS Demo - Students evaluation].[AVG_classification], **WHERE** [Time Dimension].[last_day_of_sem] <= TO_DATE('2009-06-30','YYYY-MM-DD') AND question filter.

The query must have the following syntax:

SELECT DIMENSIONS List of Dimensions (min 1; separated by coma)

FACTS List of Facts (min 1; separated by coma)

[optional *WHERE* conditions of the query]

The results of the reports or ad-hoc queries are automatically grouped and sorted by the dimensions' list.

Web Service functions exposed in the WSDL of the BIS Web Service:

port.report(s4WSSecurityString, s4WSToken, BisUserName, OlapWSReport, RepParams): A function to execute a Predefined report.
This function returns a string containing the sessionid that is necessary to maintain session persistence.

Function Arguments:

- **s4WSSecurityString:** This property was defined in the text box **WS Phrase** of **Web Services Security Properties** of s4WSProperties tool and saved to the property **X500USERNAME** in **s4Properties** file.
- **s4WSToken:** This property was defined in the text box **Pass Phrase** of **Web Services Security Properties** of s4WSProperties tool**.**
- **BisUserName:** The login name of an Executive, Navigator, Designer or Administrator user**.**
- **OlapWSReport:** the name of web service of an executive report to be executed. The **BisUserName** must belong to a group with authorization to execute this Web Service Report.
- **Reparams:** An optional list of arguments that must correspond to the number and types of answers required by the question filters of the report.

port.adHocQuery(s4WSSecurityString, s4WSToken, BisUserName, OlapSQL): A function to execute an Ad-Hoc query. Function not allowed for Executives. This function returns a string containing the sessionid that is necessary to maintain session persistence.

Function Arguments:

- **s4WSSecurityString:** This property was defined in the text box **WS Phrase** of **Web Services Security Properties** of s4WSProperties tool and saved to the property **X500USERNAME** in **s4Properties** file.
- **s4WSToken:** This property was defined in the text box **Pass Phrase** of **Web Services Security Properties** of s4WSProperties tool**.**
- **BisUserName:** The login name of a Navigator, Designer or Administrator user**.**
- **OlapSQL:** am Ad-Hoc query using the below syntax. The **BisUserName** must have permissions to the facts of the query.

The query must have the following syntax:

SELECT DIMENSIONS List of Dimensions (min 1; separated by coma)

FACTS List of Facts (min 1; separated by coma)

[*WHERE* conditions of the query]

List of Dimensions: [Dimension Name].[Attribute Name]

List of Facts: [Fact Name].[Metric Name]

port.errorCode(mysessionid): Returns the Error Code of the Report / Query execution. Zero (0) means OK.

port.errorMsg(mysessionid): Returns a string with the Error Message of a Report / Query execution.

port.getRows(mysessionid): Returns the number of Rows in the result set.

port.getColumns(mysessionid): Returns the number of Columns in the result set.

port.getColumnHeader(mysessionid, ColNumber):): Returns the name of a specific Column of the result set.

port.getColumnType(mysessionid, column): Returns the type of a specific Column of the result set.

port.getValue(mysessionid, row, column): Returns the Value of a cell located in a specific Row+Column of the result set.

port.closeSession(mysessionid): Should be executed by the end of retrieving all the cells of the result set, in order to release the resources of the Application server allocated for the execution of the web service session.

The sample JSP part of a Web Service Client:

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP S4 WebService Sample Page</title>
   </head>
   <body>
      <h1>S4 WS Client JSP Sample (GlassFish)</h1>
      <%-- start web service invocation --%><hr/>
      <%
      ws.S4ServerWSService service = null;
      ws.S4ServerWS port = null;
      java.lang.String OlapSQL = “”;
      java.lang.String mysessionid = "";
      try {
         service = new ws.S4ServerWSService();
         port = service.getS4ServerWSPort();
         java.text.DateFormat df = new java.text.SimpleDateFormat("yyyy-MM-dd hh:mm:ss.SSS");
         java.util.Date today = new java.util.Date();
         out.println("getData(begin) :" + df.format(today) + "<br>");
         java.lang.String s4WSSecurityString = "s4WS";
         java.lang.String s4WSToken =
"<x500Token>0C12157737161175311013713512157737161175311013713512157737161175 31</x500Token>";
         java.lang.String BisUserName = request.getParameter("s4User");
         java.lang.String SQLParam = request.getParameter("s4SQL");
         if (SQLParam == null) {
            SQLParam = request.getParameter("s4Report");
            OlapSQL = SQLParam;
            java.util.List<java.lang.String> RepParams = null;
            if (request.getParameter("s4User") == null) {
```

```
            if (request.getParameterMap().size() > 1) {
                RepParams = java.util.Arrays.asList(request.getParameterValues("Par"));
            }
        } else {
            if (request.getParameterMap().size() > 2) {
                RepParams = java.util.Arrays.asList(request.getParameterValues("Par"));
            }
        }
        mysessionid = port.report(s4WSSecurityString, s4WSToken, BisUserName, OlapSQL, RepParams);

    } else {
        OlapSQL = SQLParam;
        mysessionid = port.adHocQuery(s4WSSecurityString, s4WSToken, BisUserName, OlapSQL);
    }
    today = new java.util.Date();
    out.println("getData(after) :" + df.format(today) + "<br>");
    if (port.errorCode(mysessionid) != 0) {
            java.lang.String ErrMsg = "S4SERVER error:" + String.valueOf(port.errorCode(mysessionid)) + " Msg:" +
port.errorMsg(mysessionid);
        out.println("<font face='Arial' size='4' color='#ff0000'>Error: " + ErrMsg + "</font>");
    } else {
        out.println(port.getQryInfo(mysessionid) + "<br>");
        int Rows = port.getRows(mysessionid);
        out.println("#Rows = " + Rows);
        int Columns = port.getColumns(mysessionid);
        out.println("#Columns = " + Columns);
            out.println("<table class=MsoTableGrid border=1 cellspacing=0 cellpadding=0 style='border-
collapse:collapse;border:none;mso-border-alt:solid windowtext .5pt;mso-yfti-tbllook:191;mso-padding-alt:0cm 5.4pt 0cm 5.4pt;mso-
border-insideh:.5pt solid windowtext;mso-border-insidev:.5pt solid windowtext'>");
        out.println("<th WIDTH=50>Row #</th>");
        for (int j = 0; j < Columns; j++) {
            out.println("<th WIDTH=" + (j < 1 ? String.valueOf((j + 1) * 100) : 100) + ">" + port.getColumnHeader(mysessionid, j)
+ "</th>");
        }
        java.text.DecimalFormat nf = new java.text.DecimalFormat("#,##0.000");
        for (int row = 0; row < Rows; row++) {
            out.println("<tr>");
            out.println("<td align=right style=border-width:10;>" + (row + 1) + "</td>");
            for (int column = 0; column < Columns; column++) {
                String ColType = (String) (port.getColumnType(mysessionid, column));
```

```
                if (ColType.compareTo("double") == 0) {
                    out.println("<td align=right>");
                    double theFact = Double.valueOf(port.getValue(mysessionid, row, column));
                    out.println(nf.format(theFact));
                } else {
                    String theValue = port.getValue(mysessionid, row, column);
                    out.println("<td width=" + String.valueOf(theValue.length() * 7) + ">" + theValue);
                }
                out.println("</td>");
            }
            out.println("</tr>");
        }
        out.println("</table>");
        today = new java.util.Date();
        out.println("<font face='Arial' size='4' color='#00ff00'>getData(Fetch End):" + df.format(today) + "</font>");
    }
} catch (Exception ex) {
    out.println("</table>");
    out.println("<font face='Arial' size='4' color='#ff0000'>Error: " + ex.toString() + "</font>");
    out.println("<hr/>");
    out.println("s4SQL: " + OlapSQL);
} finally {
    if (port != null) {
        port.closeSession(mysessionid);
        port = null;
    }
    if (service != null) {
        service = null;
    }
}
%>
<%-- end web service invocation --%><hr/>
<hr/>
</body>
</html>
```