How to…

Send XML Data to BW

BUSINESS INFORMATION WAREHOUSE

ASAP “How to…” Paper

Applicable Releases: BW 3.0B, BW 3.5
Version 1.01

September 2004

1 Business Scenario

This document describes the solution developed for BW 3.0A Patch 3 that you can send to BW with the data in XML format by using the Internet log HTTP. You do this in order to integrate the solution with the known staging methods into the consolidated data.

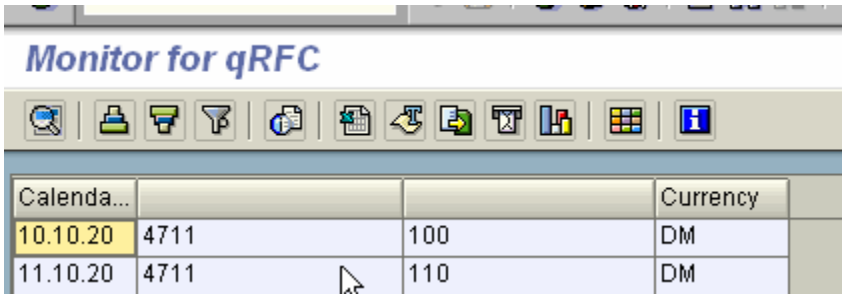
You need to send the following data records to BW in an XML document:

Calendar Day: 10.10.2001
Material: 4711
Amount: 100
Currency: DM

Calendar Day: 11.10.2001
Material: 4711
Amount: 110
Currency: DM

2 The Result

The XML data is stored in the delta queue and, consequently, integrated into the BW staging process.



3 The Step-By-Step Solution

1. Preliminary remarks

It can be assumed that, in this way, only limited amounts of data (document characters) can be transferred with each call. Therefore, the data is collected after being transferred into the delta queue of a suitably generated DataSource.

The implementation is based on the service for transferring XML data that was provided by SAP technology for Release 6.10. This XML data is sufficient for the SOAP log (Simple Object Access Protocol) for RFC-capable function modules in the ABAP environment.

The XML data has to be assigned according to an XML schema definition, which is derived from the definition of a DataSource.

2. An XML 3.0 parser has to be installed.

   Link:
   http://msdn.microsoft.com/downloads/

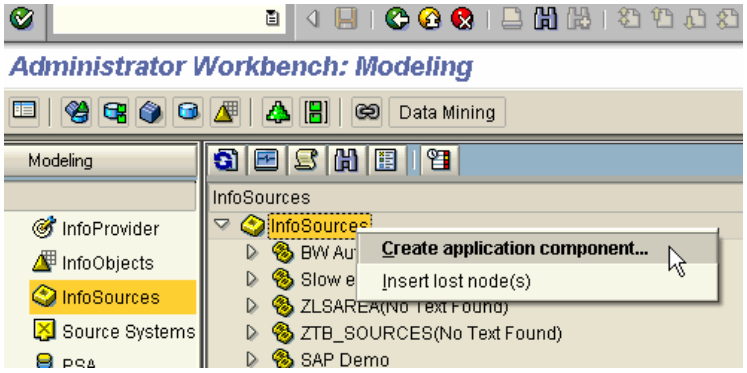
   → Download page
   → search for the actual driver/program

   Afterwards, you have to run the *Xmlinst.exe Replace Mode Tool.*

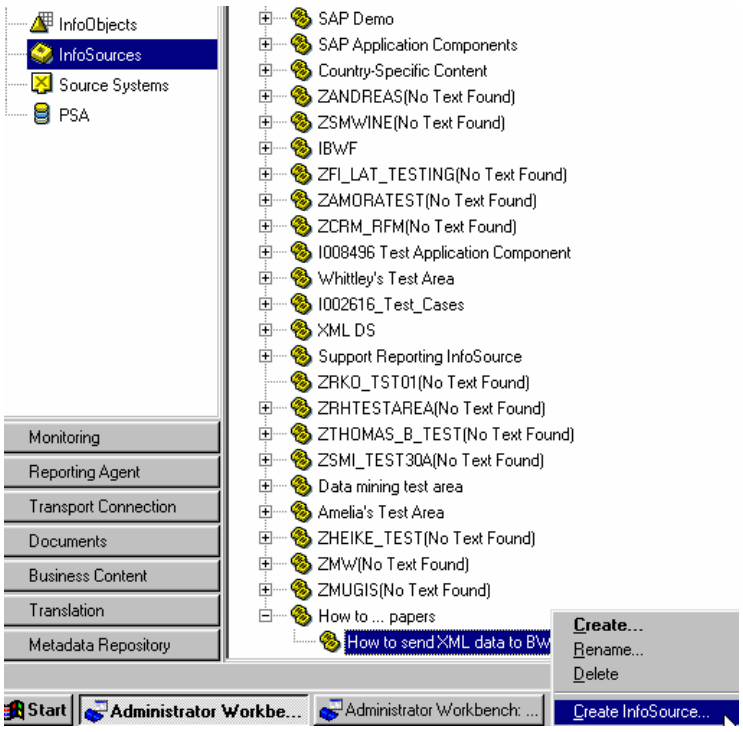
   Link:
   http://msdn.microsoft.com/downloads/

   → Download page
   → search for the actual driver/program

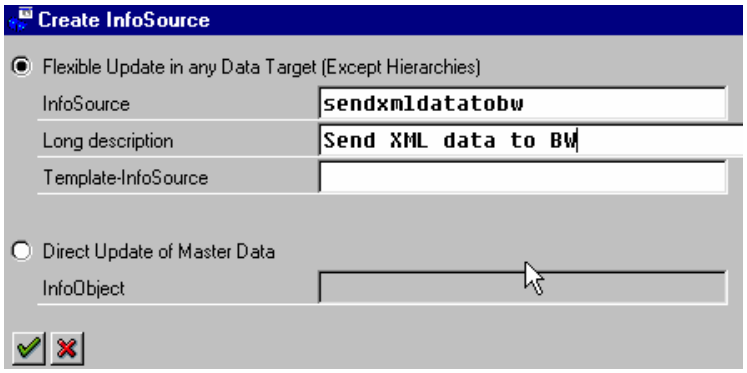
3. Create an Application Component
   Right-click the mouse on *Application Component*



4. Create an InfoSource. Right-click the mouse on *Application Component* → *Create InfoSource*

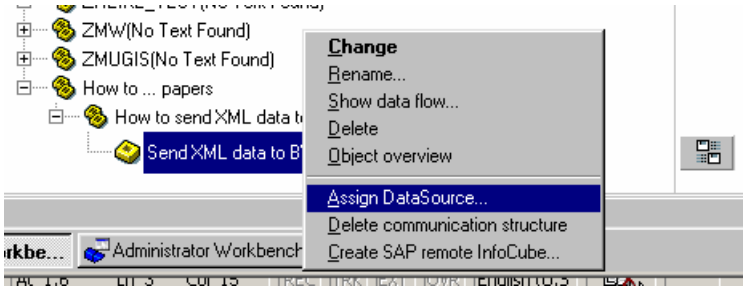


5. Enter the technical name and the description of the InfoSource.

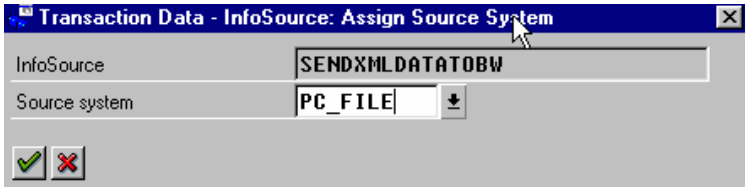


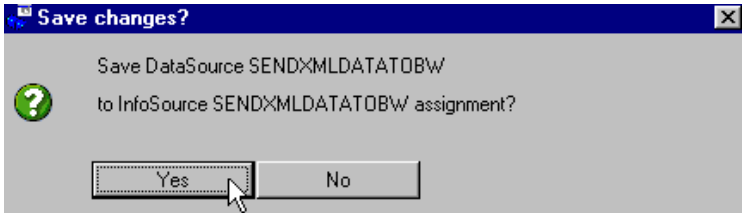
6. Create a DataSource transaction data file.

   Right-click the mouse on *InfoSource* → *Assign DataSource*



7. Assign the source system file and save the changes.

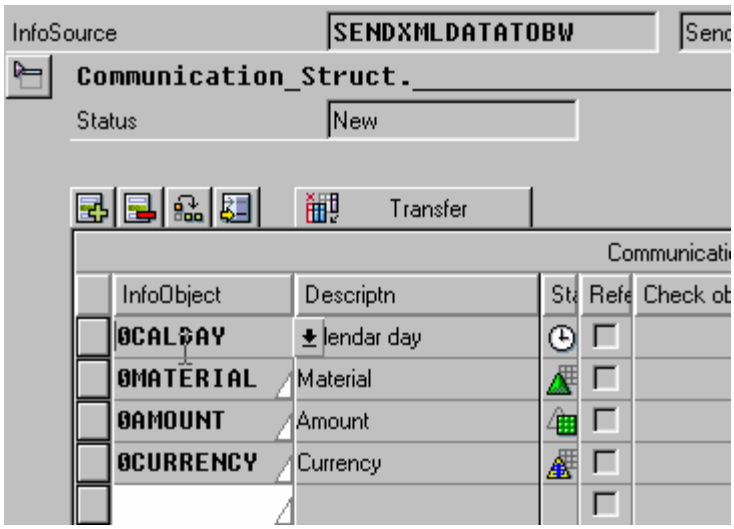




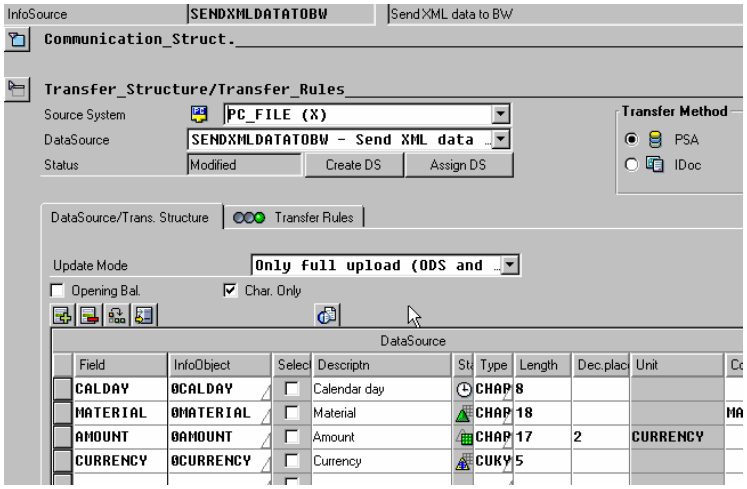
8. Assign the technical names of the InfoObjects to the business names.

Business Name	InfoObject Technical Name
Calendar Day	0CALDAY
Material	0MATERIAL
Amount	0AMOUNT
Currency	0CURRENCY

9. Create the communication structure.



10. Create the transfer structure.

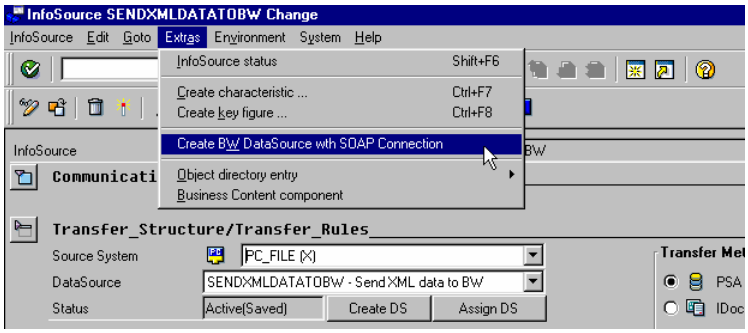


11. Save

and activate.

12. Call up the generation of an XML interface.

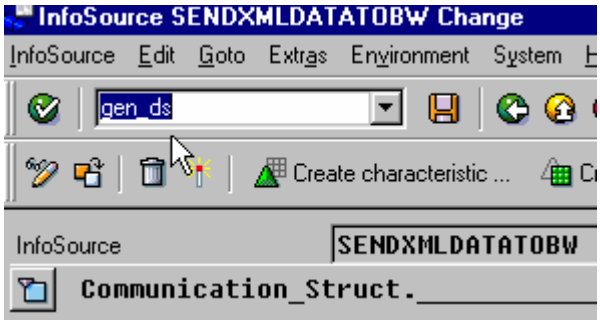
    Extras → Create BW DataSource with SOAP Connection



    or

    Function code / OK-Code: GEN_DS

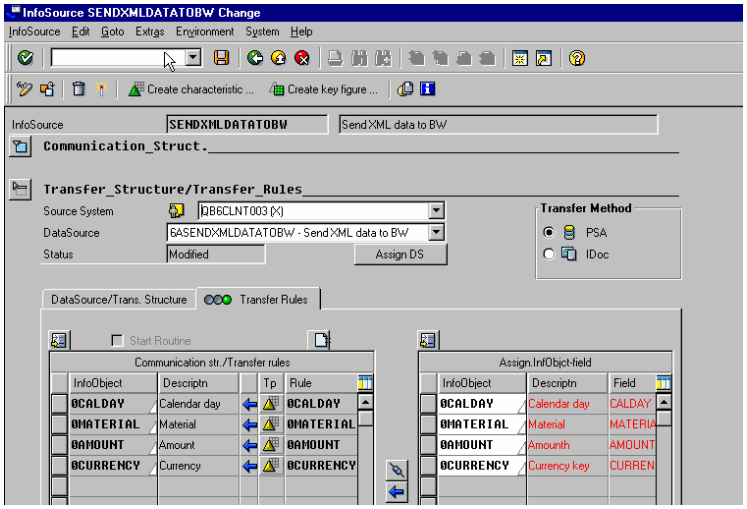
or



13. Answer all dialog boxes with *Yes*.

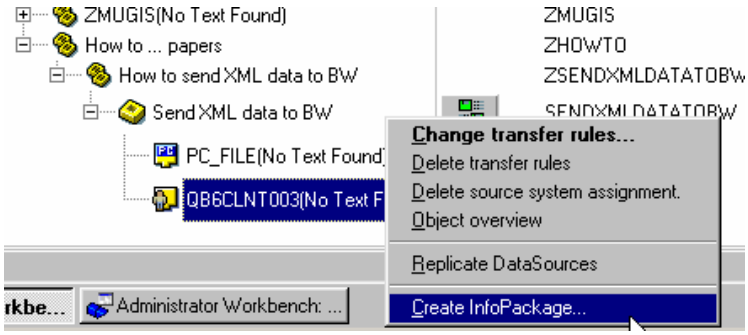
14. After the successful generation, the DataSource is connected to the Self Source System.

    The name of the generated DataSource is :
    6ADatasource

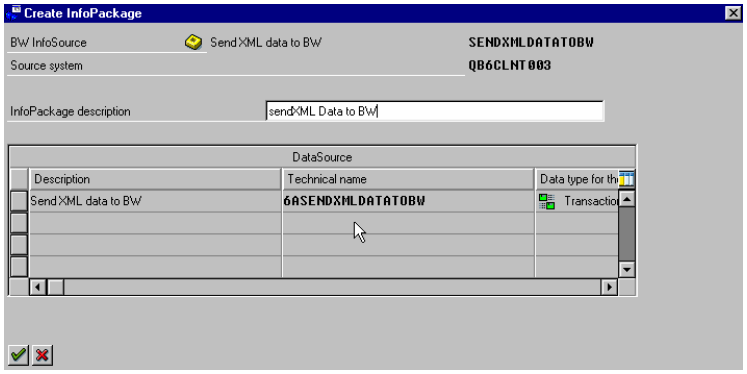


15. Save and activate.

16. Create an InfoPackage for the Self Source System.

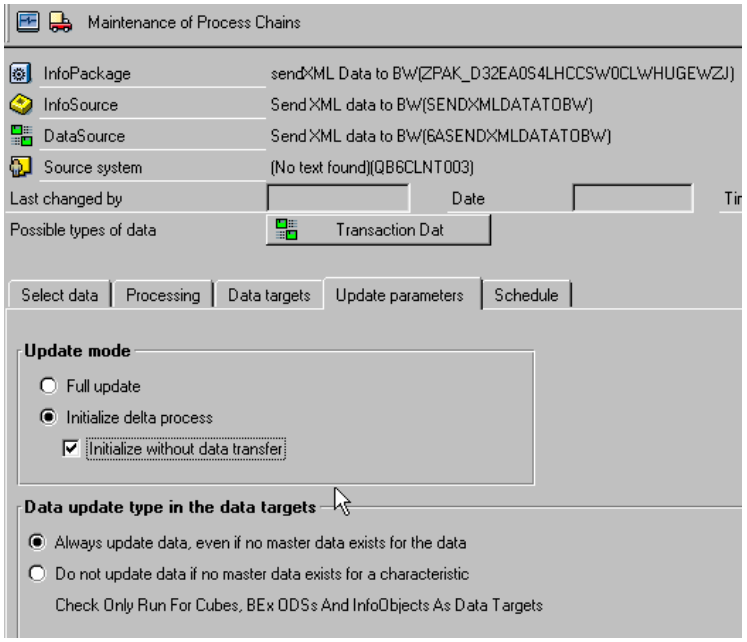


17. Enter the InfoPackage description and confirm with *Enter*.



18. The initial upload has to be simulated for the installation of the delta queue. To do this, you change the update mode in the update parameters by selecting *Initialize delta process* and *Initialize without data transfer*.

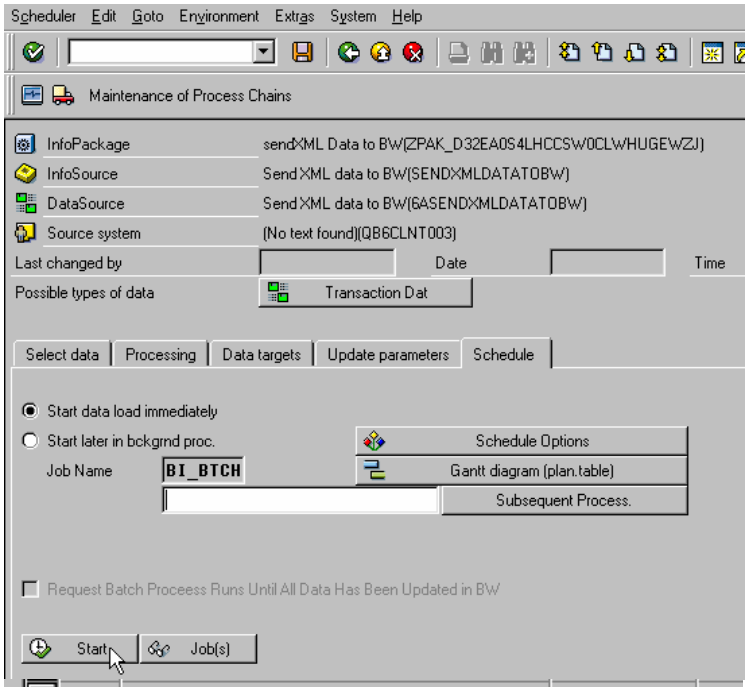
    The InfoPackage is saved.



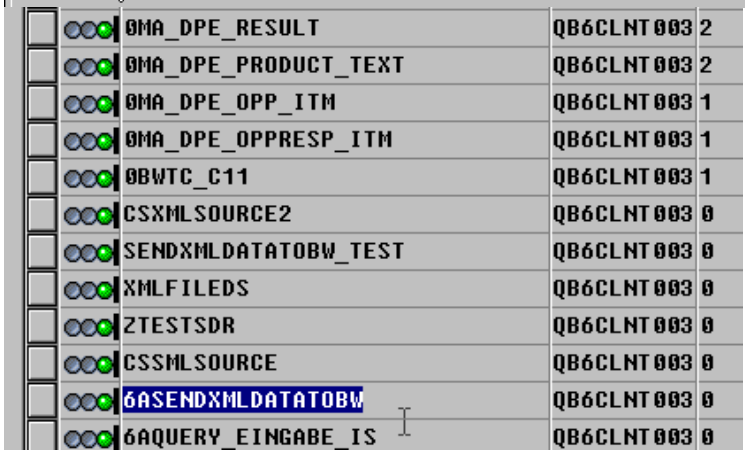
19. You install the delta queue by starting the upload in the InfoPackage *Schedule*.

    The initialization can be checked by using transaction RSA7. There has to be an entry there with the technical name of the DataSource.

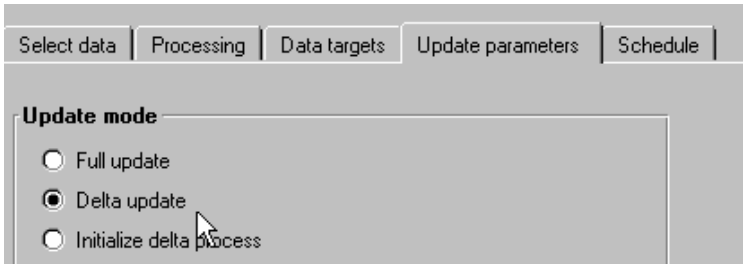
    There is a red light on the monitor that needs to be set to green.



20. Example of an entry in transaction RSA7.



21. In order to load the XML data from the delta queue, you have to create a new InfoPackage with the update parameter *Delta update*. The InfoPackage is only created and not started.



22. BW Release 3.0:

For BW Release 3.5, please jump to point 26.

In this example, the XML data is stored in a file.

The XML file compares with an XML stream, which is sent from Java applications.

Explanation of the XML document:

Lines 1- 4
– The XML header is always the same.

<?xml version="1.0" ?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">

Line 5
- Opening the SOAP body
<SOAP:Body>

Lines 6-7
- Remote function call
– This call is always determined

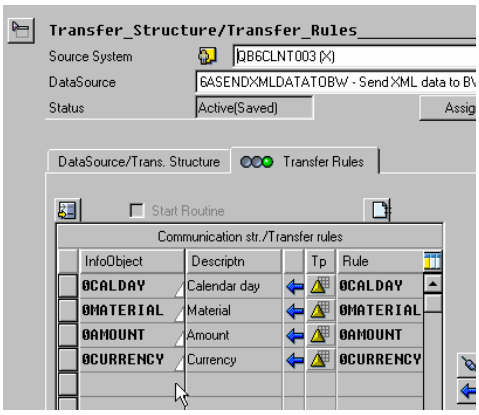
```
<?xml version="1.0" ?>
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelop
e/">
<SOAP:Body>
<rfc:_-BI0_-QI6ASENDXMLDATATOBW_RFC
xmlns:rfc="urn:sap-com:document:sap:rfc:functions">
<DATASOURCE>6ASENDXMLDATATOBW</DATASOURCE>
<DATA>
       <item>
              <CALDAY>20011010</CALDAY>
              <MATERIAL>4711</MATERIAL>
              <AMOUNT>100</AMOUNT>
              <CURRENCY>DM</CURRENCY>
       </item>
       <item>
              <CALDAY>20011011</CALDAY>
              <MATERIAL>4711</MATERIAL>
              <AMOUNT>110</AMOUNT>
              <CURRENCY>DM</CURRENCY>
       </item>
</DATA>
</rfc:_-BI0_-QI6ASENDXMLDATATOBW_RFC>
</SOAP:Body>
</SOAP:Envelope>
```

by:
<*rfc:_-BIC_-QI6A*InfoSource*_RFC xmlns:rfc="urn:sap-com:document:sap:rfc:functions*">
- The InfoSource is the technical name of the InfoSource, in this example.
6ASENDXMLDATATOBW

Line 8
– The technical name of the DataSource in this date, in this example:
6ASENDXMLDATATOBW

<*DATASOURCE*>*6AInfoSource*</*DATASOURCE*>

Lines 9-24
- The data is found here in flat form in an XML wrapper.
– The data package is closed with <*DATA*> and opened with </*DATA*>. There is only one data package in which lines are included.
– The different lines are opened with <*ITEM*> and closed with </*ITEM*>.
– The field names are the technical field names for the DataSource.



- In this example
<*CALDAY*>*20011010*</*CALDAY*>
<*MATERIAL*>*4711*</*MATERIAL*>
<*AMOUNT*>*100*</*AMOUNT*>

<CURRENCY>DM</CURRENCY>

- Lines 25 - 27
Since all open HTML-Tag's have to be closed again in XML, the function call, body, and envelope part is closed.

</rfc:_-BIC_-QI6AInfoSource_RFC>
</SOAP:Body>
</SOAP:Envelope>

23. After the delta process is installed and the XML document is correctly created, you can send the document to BW by using an HTTP call. This is implemented in this example by using a queue of different Javascript functions based on Internet Explorer 5.1. Of course, it is possible to write which functions the XML document generates and sends to BW for a small Java application. It is important to note that everything is created according to the SOAP standard. Some class libraries that support such calls exist in the Java environment.

24. The HTML code is listed here for uploading an XML file and sending this to the BW Web application server by using HTTP. This source code can be copied from here and can then be opened as an HTML site with Internet Explorer 5.1.

    The file name is marked in red. This points out the XML document (see above).

    The address of the Web application server is marked in blue. The `/sap/bc/soap/rfc` is always determined for this address: `http://ds0056.wdf.sap-ag.de:1080/sap/bc/soap/rfc` This means that you have to change `"ds0056.wdf.sap-ag.de:1080"`.

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1252">
<meta name="GENERATOR" content="Microsoft FrontPage
4.0">
<meta name="ProgId"
content="FrontPage.Editor.Document">
<title>New Page 1</title>

<SCRIPT ID=clientEventHandlersJS
LANGUAGE=javascript>
<!--

function SendRS_onclick() {

// works with InternetExplorer 5.5 SP1

      var xmlstream = new
ActiveXObject("ADODB.Stream");


      var mypath = document.myform.filename.value;
      var myport = document.myform.portname.value;

      xmlstream.Mode = 3;


                  // 1=read  3=read/write
      xmlstream.Open();
      xmlstream.Type = 1;


                  // 1=adTypeBinary  2=adTypeText
      xmlstream.LoadFromFile(mypath);
//    xmlstream.Write("cdtest");
//    xmlstream.SetEOS;
//    xmlstream.Position = 0;

      divOutputRequest.innerText = xmlstream.Size;



      var xmlhttp = new
ActiveXObject("Msxml2.XMLHTTP");
//    xmlhttp.Open("POST","http://ds0056.wdf.sap-
ag.de:1080/sap/bc/soap/rfc",false);
      xmlhttp.Open("POST",myport,false);
      xmlhttp.setRequestHeader("Content-
Length",xmlstream.Size);
                        //set the length of the
content
      xmlhttp.setRequestHeader("Content-
Type","text/xml");

      xmlhttp.send(xmlstream.Read(xmlstream.Size))
;
                              //Send the stream

      divOutputResponse.innerText =
xmlhttp.responseText;
//    alert(xmlhttp.responseText);
}

//-->
</SCRIPT>
```

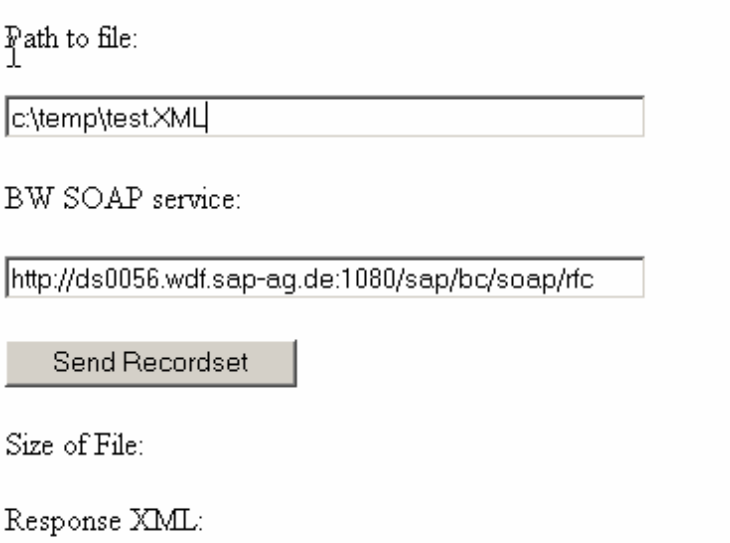
```
</head>

<body>

<FORM name="myform">
<P>Path to file: </P>
<input type="text" name="filename" size=50
maxlength=80 value="c:\temp\">
<P>BW SOAP service: </P>
<input type="text" name="portname" size=50
maxlength=80 value="http://ds0056.wdf.sap-
ag.de:1080/sap/bc/soap/rfc">
<p></p>
<INPUT type="button" value="Send Recordset"
id=SendRS name=SendRS LANGUAGE=javascript
onclick="return SendRS_onclick()">
<P>Size of File:</P>
<DIV id=divOutputRequest></DIV>
<P>Response XML:</P>
<DIV id=divOutputResponse></DIV>
</FORM>
</body>
</html>
```

25. By clicking on the *Send Recordset* button, you send the XML document to BW and write it into the delta queue.
You can check this by using transaction RSA7.

NOTE:
Please check the link http://support.microsoft.com/default.aspx?scid=kb;en-us;870669 for more details. (see Comments)

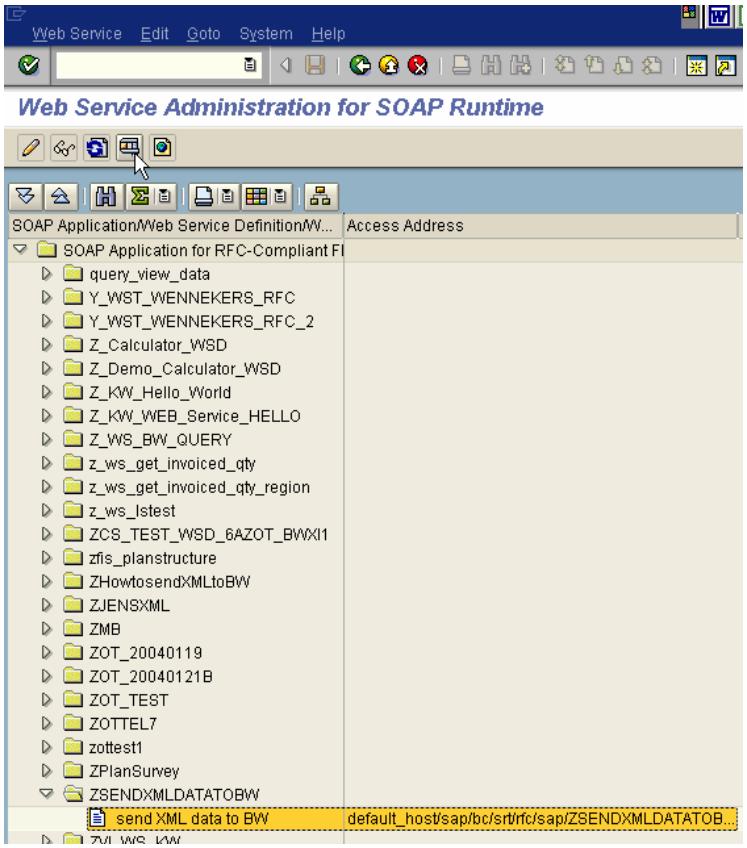


26. BW Release 3.5:
Since BW release 3.5 an WebService on top of the function module can be gernerated. This WebService can be used to send data to the delta queue.

Transaction SE37
(in this case function module "/BI0/QI6ASENDXMLDATATOBW_RFC")

→ Display → Utilities → More Utilities → Create Web Service → From the function module
(Please use for the "virtual Interface" and the " Web Service" the customer name range.)

Please use the Transaction WSADMIN to test the WebService.



27. If the data is in the delta queue once, it can be loaded by using the normal DeltaLoad Staging Process with the InfoPackage, which is described in step 19.

4 Comments

- Namespace: You can only use file DataSources that are not in the SAP namespace (digit as first character). The name of the DataSource that was first created currently has the same name as the InfoSource used when connecting to a file source system and cannot be changed. Because of this, an unnecessary DataSource has to be created as a starting point with content InfoSources before you can choose the name freely (without important digits) for a second DataSource.
- Up to Support Package 1, the menu function is hidden. However, you can execute it by using the OK-Code: `GEN_DS`.
- RFC function modules are still not deleted automatically when deleting the myself DataSource.

- Up until now, a file DataSource did not support the option of identifying data records as deletion records for a delta process. This restriction is valid for the queue connection as well. One of the DataSources delivered in the framework of SAP content and having the same pattern can, however, also arrange the process „After-images with deletion indicator delta queue (*AIMD)“* if it is established that the *record mode* that belongs to it is set correctly.
- At the moment, myself DataSources and function modules are created locally and privately. They are also not scheduled for the transport connection.
- With a delivered Internet Explorer Patch the ActiveX function ADOBD.Stream is disabled. It can be enabled by changing the registry entry. Please check the link http://support.microsoft.com/default.aspx?scid=kb;en-us;870669 for more details. It is recommended to check the way how to post the XML data from outside to BW with your company security policy. Since BW Release 3.5 (NetWeaver ’04) it is possible to use a WebService to send the data to the delta queue.

5 Summary

The use of the delta queue from the Service API is the basis of a solution as an inbound queue for the XML document data that was parsed in the SOAP service and converted in ABAP fields.

Special DataSources are generated for this within the BW system and have the following properties: They have the BW system as a source system, they are exclusively intended for uploading delta records, and they have an interface for supplying the delta queue.

The interface, that was released for developing extractors within the framework of SAPI for uploading the delta queue, is wrapped for this by using a DataSource-specific, RFC-capable function module that is generated for the DataSource. Based on the RFC-capability, this function module can be automatically identified by using the assigned HTTP handler from Basis for supporting the SOAP log.

The starting point for generating such a special DataSource is a file DataSource. You can use the file DataSource for supplying BW with large amounts of data that do not need to be run by using the XML interface. It serves to characterize the XML data in BW. Actually, the entire solution consists of two DataSources that logically belong together. One of these is for a file source system, and the other is generated within the BW system.

This connection is deliberately not assigned for DataSources from other sources. The assumption should not be made that copies of these DataSources would create the desired XML procedures. It is up to the person responsible for the application to decide whether they want to deliver their own solution based on the generic solution described here.