How to…
Expose A BW Query As Web Service In A UDDI Registry

Version 1.00 – January 2005

Applicable Releases:
SAP NetWeaver ‘04
(SAP BW 3.5)
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.
1 Business Scenario

You would like to expose information from a BW Query to the outside world. The consumers of this information should not have to have any knowledge about BW, e.g. how to run a query in a BW System. In our example the query returns the invoiced quantity for a given country and region.

2 Introduction

Web Services can be used to provide simple scenarios that do not require any special knowledge about a component system. In SAP NetWeaver'04 all functionality is available to base a Web Service on a function module and expose it to a UDDI Registry. We deploy the OLAP BAPI in our own function module in order to run a given query with pre-defined variable parameters (country and region). The OLAP BAPI uses the MDX (Multi-Dimensional Expressions) standard for running queries in BW.
3 The Step By Step Solution

3.1 Implement Your Own Function Module Which Runs a Fixed Query

1. At first you may want to get an overview of the OLAP BAPI. In transaction BAPI you find the OLAP BAPI with its methods and documentation.

2. Create your own function module in transaction SE37. Use import parameters you need for specifying filter values for the query.

3. Specify export parameters for the key figure value(s) and a return parameter for checking successful execution.

4. In the source code you have to create the MDX statement and run it with the OLAP BAPI. Please check the appendix for the complete sample code.

```plaintext
CONCATENATE i_country i_region
INTO l_name SEPARATED BY space.
CONCATENATE '"' l_name '"'
INTO l_name.
append_mdx 'SELECT [Measures].MEMBERS ON COLUMNS,'.
append_mdx 'FILTER ( [REGION].MEMBERS,
[REGION].CURRENTMEMBER.NAME = '.
append_mdx l_name.
append_mdx ') ON ROWS'.
append_mdx 'FROM FABCUBE5/KWS'.
```
3.2 Create a Web Service Using the Function Module

There are several ways to create a Web Service. You can start the Web Service Wizard directly (transaction WS_WZD_START). Alternatively, you can create Web Service in SE80 or directly start the Wizard from the Function Builder (SE37). Either way the result will be the same.

5. When editing the function module, select Utilities → More Utilities → Create Web Service → From the Function Module in transaction SE37.

6. The wizard starts and gives some introductory information. Press “Continue”.

7. The virtual interface is created. Specify a name, short description, specify the Endpoint Type (already defaulted to “Function Module” if you start from transaction SE37). Press “Continue”.

---

- 3 -
8. Specify the function module (defaulted correctly if you start from transaction SE37). Press "Continue".

9. Now the Web Service itself is created. Specify name, short description and choose an authorization profile. Press "Continue".

10. Check all settings and press "Complete". All objects will then be created.
3.3 Publish the Web Service to a UDDI Registry

The Web Service is now available and ready for use. You don’t have to do anything in addition. Publishing the Web Service in a UDDI Registry makes the Web Service known to people who have access to that registry. Let’s assume that we have a UDDI Registry available. Please check the appendix if the UDDI Registry is not known or not available.

11. In transaction SE80 locate and double click on your Web Service, choose the UDDI tab and press "Publish".

12. Choose the registry you want to use for publishing and specify user and password of the UDDI Registry if necessary.

13. A line is added to the list specifying the UDDI Registry and a generated UUID. The Web Service is now available and can be found in the UDDI Registry.
14. If the UDDI Registry of a SAP J2EE Engine is used you can use the link http://<servername>:<port>/uddiclient. Choose “Search Registry” and press “Next”.

15. Choose the registry. Mind that the names of the (links to the) registries differ between the one defined before in the transaction and the one defined in the J2EE Engine itself. Nevertheless they can point to the same physical registry. Specify your search criteria using “%” as a wildcard and press “Search” at the bottom.

16. You will find your Web Service in the registry.
4 Appendix

4.1 Sample Code for BW Query Function Module

FUNCTION z_ws_get_invoiced_qty_region.

**Local Interface:**
**IMPORTING**
  VALUE(I_COUNTRY) TYPE /BI0/OICOUNTRY
  VALUE(I_REGION) TYPE /BI0/OIREGION
**EXPORTING**
  VALUE(ES_RETURN) TYPE BAPIRET2
  VALUE(E_QUANTITY) TYPE BAPI6111CD-VALUE

CONSTANTS:
  c_max_cells   TYPE bapi6111cd-cell_ordinal VALUE 10000.

DATA:
  l_name        TYPE bapi6111mdx-line,
  l_datasetid   TYPE bapi6111gen-obj_handle,
  ls_mdx        TYPE bapi6111mdx,
  lt_mdx        TYPE TABLE OF bapi6111mdx WITH DEFAULT KEY,
  ls_cell_data  TYPE bapi6111cd,
  lt_cell_data  TYPE TABLE OF bapi6111cd WITH DEFAULT KEY.

DEFINE append_mdx.
  ls_mdx = &1.
  append ls_mdx to lt_mdx.
END-OF-DEFINITION.

* Compose MDX statement
CONCATENATE i_country i_region INTO l_name SEPARATED BY space.
CONCATENATE "" l_name "" INTO l_name.

  append_mdx 'SELECT'.
  append_mdx '[Measures].MEMBERS ON COLUMNS,'.
  append_mdx 'FILTER ( [0REGION].MEMBERS, [0REGION].CURRENTMEMBER.NAME ='.
  append_mdx l_name.
  append_mdx ') ON ROWS'.
  append_mdx 'FROM FABCUBE5/KWWS'.

CLEAR es_return.

* Create MDX data set
CALL FUNCTION 'BAPI_MDDATASET_CREATE_OBJECT'
IMPORTING
  datasetid    = l_datasetid
  return       = es_return
TABLES
  command_text = lt_mdx.

CHECK es_return IS INITIAL.

* Run MDX data selection
CALL FUNCTION 'BAPI_MDDATASET_SELECT_DATA'
EXPORTING
  datasetid = l_datasetid
IMPORTING
  return    = es_return.

CHECK es_return IS INITIAL.
* Get MDX result set
CALL FUNCTION 'BAPI_MDDATASET_GET_CELL_DATA'
   EXPORTING
       datasetid = l_datasetid
       start_cell = 0
       end_cell   = c_max_cells
   IMPORTING
       return     = es_return
   TABLES
       cell_data  = lt_cell_data.
   CHECK es_return IS INITIAL.

* Get return value
CLEAR e_quantity.

   READ TABLE lt_cell_data INTO ls_cell_data INDEX 1.
   IF sy-subrc = 0.
       e_quantity = ls_cell_data-value.
   ENDIF.

ENDFUNCTION.

4.2 Activating the UDDI Server Built Into J2EE Engine

In SAP NetWeaver'04 documentation navigate to NetWeaver ➔ Application Platform ➔ Java Technology in SAP Web Application Server ➔ Administration Manual ➔ Server Administration ➔ SAP J2EE Engine Administration ➔ Web Services Container Service ➔ Managing the UDDI Server. Follow the steps described in order to activate the UDDI Server functionality. The UDDI user you create is the one you need for step 12.

4.3 Creating a Link to the UDDI Registry

For publishing a Web Service (e.g. from transaction SE80) a link to the UDDI Registry must be defined.

1. In transaction SM59 create new RFC destinations for publishing to inquiring from the UDDI Registry.
2. Specify name, description, and connection type “G”. Fill in the server of the J2EE Engine (if the UDDI Registry of the J2EE Engine is used) and as Service No. specify the port of the J2EE Engine. The default path prefix for the J2EE UDDI Registry is displayed in the screen shot.

3. Create a similar entry for inquiring the Registry.

4. In transaction SUDDIREG switch to change mode and create a new entry.

5. Specify the RFC destinations created before in the Inquire and Publish API Destinations. You can specify a user and a password (the one created for the UDDI Server). Save your new entry.
# 5 Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAPI</td>
<td>Business Application Programming Interface</td>
</tr>
<tr>
<td>J2EE</td>
<td>Java 2 Platform, Enterprise Edition</td>
</tr>
<tr>
<td>MDX</td>
<td>MultiDimensional eXpressions</td>
</tr>
<tr>
<td>OLAP</td>
<td>Online Analytical Processing</td>
</tr>
<tr>
<td>RFC</td>
<td>Remote Function Call</td>
</tr>
<tr>
<td>UDDI</td>
<td>Universal Description, Discovery, and Integration</td>
</tr>
</tbody>
</table>