DOE's Attachment Capability - with a Sample Application

Applies to:

SAP NetWeaver Mobile 7.1 SP03 and above, DOE (Data Orchestration Engine)

Summary

This document illustrates a sample application that uses the attachment (BLOB) handling capability of the DOE.

Author: Rohit Bhatnagar

Company: SAP labs India Pvt Ltd.

Created on: 26 October 2009

Author Bio

I work in the development team for SAP NetWeaver Mobile, and have been associated with this product for the last 2.2 years. My areas of expertise includes DOE Work-bench, Design time, and Backend Integration runtime.

Table of Contents

1. Introduction

Application to display Employee details which includes BLOB data (Image) on a Laptop client using DOE middleware.

Definitions you need to know:

Binary **L**arge **Ob**ject: A database field that holds any digitized information, including text, images, audio or video. Also known simply as a "large object" or LOB, a BLOB may have a huge storage capacity.

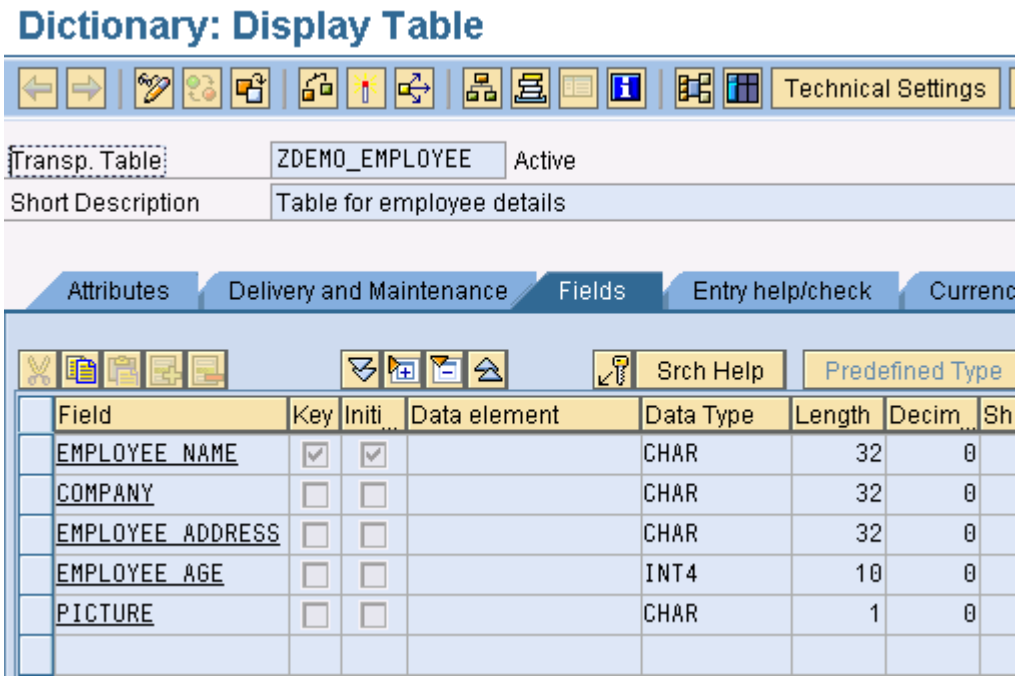
Character **L**arge **Ob**ject: A database field that holds a large amount of text (character data). Also known as a "memo field" in some database programs. A DBCLOB is a "Double Byte" CLOB that supports two-byte Unicode characters

2. Backend Related implementation

2.1 Backend schema & Backend Application

To demonstrate an example lets consider only one table as Employee. For employee details we have employee name, company, address, age and image which we want to distribute to devices. Images will be distributed as attachment

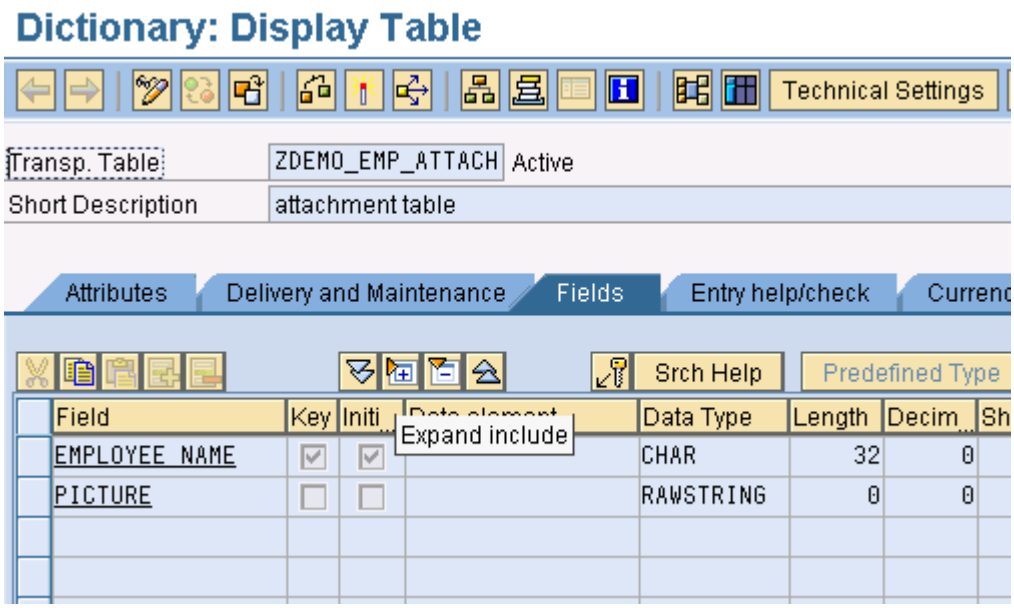
First create one table as shown below:



Here Picture is declared as CHAR1 because this field will indicates weather there is an attachment or not for that employee record based on 'X' or '' (Space) value.

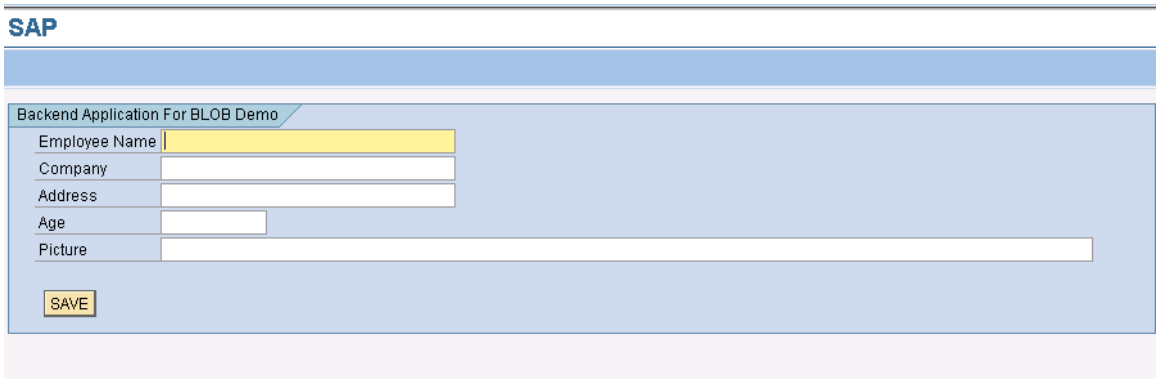
Since there are two RFC function calls(GetDetail & GetAttachment) for every key, GetAttachment function call can be avoided for those keys by making the attachment fields(Char1 field) SPACE.

Now create one more table with only key of above table and BLOB (image) data.



Picture is declared as RAWSTRING.

For loading data create a simple backend application,



For persisting picture we need to convert Binary object to Xstring PAI Function module of application screen may look like this:

```
REPORT  ZDEMO_EMPLOYEE_BLOB.
tables:zdemo_employee.

data: file_path_pic type string.
data: wa_employee type zdemo_employee.
data : wa_emp_attach type zdemo_emp_attach.
DATA  file_table type filetable.
data wa_filetable like line of file_table.

MODULE USER_COMMAND_1000 INPUT.
  data:gfcode like sy-UCOMM.
  gfcode = sy-ucomm.
  CASE gfcode.
    WHEN 'BACK'.
      set screen 0.
    WHEN 'SAVE'.
      "save employee profile data
      perform save_data.
    WHEN OTHERS.
```

```
  ENDCASE.
ENDMODULE.                 " USER_COMMAND_1000  INPUT
*&---------------------------------------------------------------------*
*&      Form  SAVE_DATA
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
FORM SAVE_DATA .

  select single * from ZDEMO_EMPLOYEE into wa_employee where employee_name = ZDEMO
_EMPLOYEE-EMPLOYEE_NAME.
  if sy-subrc <> 0.
    wa_employee-EMPLOYEE_NAME = ZDEMO_EMPLOYEE-EMPLOYEE_NAME.
    wa_employee-COMPANY = ZDEMO_EMPLOYEE-company.
    wa_employee-EMPLOYEE_ADDRESS  = ZDEMO_EMPLOYEE-EMPLOYEE_address.
    WA_EMPLOYEE-EMPLOYEE_AGE = ZDEMO_EMPLOYEE-EMPLOYEE_age.
    if file_path_pic is not initial.
      wa_employee-PICTURE = 'X'.
    endif.
    insert zdemo_employee from wa_employee.
    " now to save attachments
    perform save_attachments.
    message 'Record saved!' type 'S'.
  else.
    message 'Username Profile Already exists' type 'S' display like 'E'.
  endif.
ENDFORM.                    " SAVE_DATA
*&---------------------------------------------------------------------*
*&      Form  SAVE_ATTACHMENTS
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
FORM SAVE_ATTACHMENTS .
  types: begin of ty_itab,
       raw(255) type x,
     end of ty_itab.
  data itab type table of ty_itab.
  data length type i.
  if wa_employee-PICTURE = 'X'.

    CALL FUNCTION 'GUI_UPLOAD'
      EXPORTING
        FILENAME   = file_path_pic
        FILETYPE   = 'BIN'
      IMPORTING
        FILELENGTH = length
      TABLES
        DATA_TAB   = itab.

    wa_emp_attach-employee_name = zdemo_employee-employee_name.
```

```
    CALL FUNCTION 'SCMS_BINARY_TO_XSTRING'
      EXPORTING
        INPUT_LENGTH = length
      IMPORTING
        BUFFER       = wa_emp_attach-picture
      TABLES
        BINARY_TAB   = itab.

    insert Zdemo_emp_attach from wa_emp_attach.
  endif.
ENDFORM.                    " SAVE_ATTACHMENTS
```

2.2. BAPI Wrappers

Now create GETLIST, GETDETAILS BAPIWRAPPERS and attachment Function module

For download attachment function module the importing parameter will be the key fields of the node

And exporting parameter will be the BLOB/CLOB field.

```
FUNCTION ZDEMO_EMPLOYEE_GETLIST.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  TABLES
*"      RETURN STRUCTURE  BAPIRET2
*"      IT_ZDEMO_EMPLOYEE STRUCTURE  ZDEMO_EMPLOYEE
*"----------------------------------------------------------------------
select * from ZDEMO_EMPLOYEE into table IT_ZDEMO_EMPLOYEE.
ENDFUNCTION.
FUNCTION ZDEMO_EMPLOYEE_GETDETAILS.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(EMPLOYEE_NAME) TYPE  ZDEMO_EMPLOYEE-EMPLOYEE_NAME
*"  EXPORTING
*"     VALUE(WA_EMPLOYEE_NAME) TYPE  ZDEMO_EMPLOYEE
*"  TABLES
*"      RETURN STRUCTURE  BAPIRET2
*"----------------------------------------------------------------------
select single * from ZDEMO_EMPLOYEE into wa_employee_name where
employee_name = employee_name.
ENDFUNCTION.
FUNCTION ZDEMO_ATTACHMENT_PIC.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(EMPLOYEE_NAME) TYPE  ZDEMO_EMPLOYEE-EMPLOYEE_NAME
*"  EXPORTING
*"     VALUE(EX_PICTURE_ATTCH) TYPE  ZDEMO_EMP_ATTACH-PICTURE
*"  TABLES
*"      RETURN STRUCTURE  BAPIRET2
*"----------------------------------------------------------------------
data wa_attach type ZDEMO_EMP_ATTACH.

select single * from ZDEMO_EMP_ATTACH into wa_attach where EMPLOYEE_NAME = EMPLOYE
E_NAME.
```

```
ex_picture_attch = wa_attach-PICTURE.
ENDFUNCTION.
```

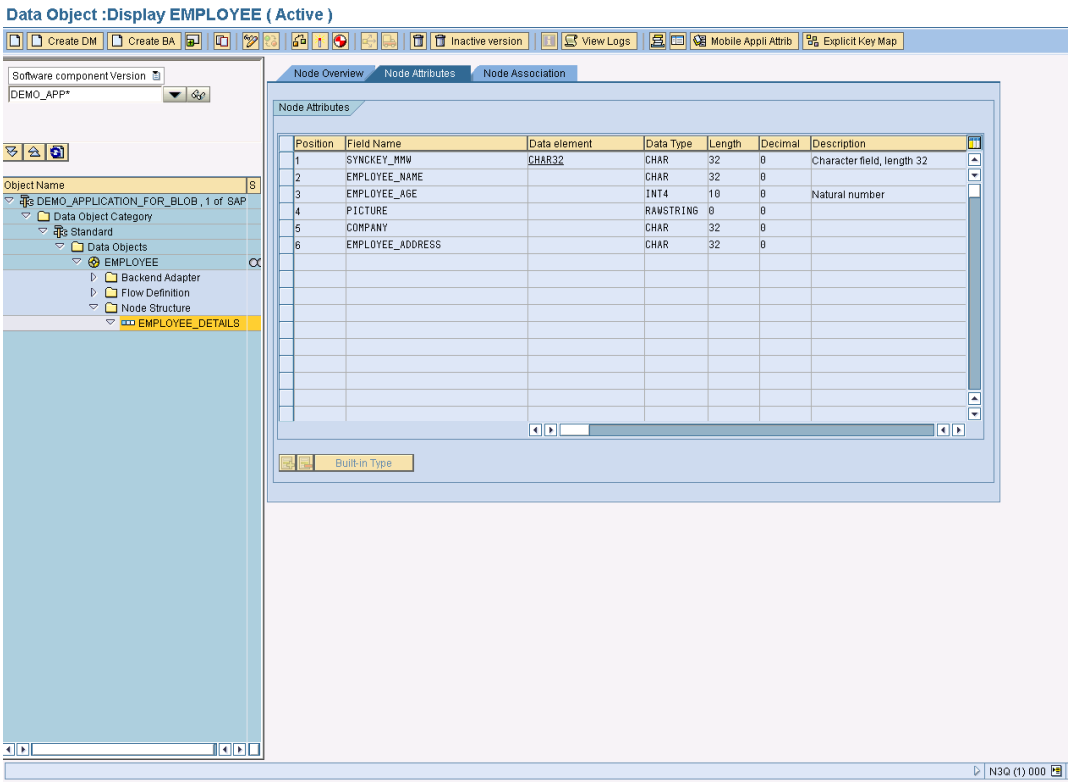
3. Data Object Modeling in DOE workbench

i. Go to SDOE_WB Transaction

ii. Create an SWCV and a data object having only one node (root node) to contain employee profile as well as image.

iii. For the field corresponding to BLOB, we need to select the check box corresponding to Binary memo. Only if this field is marked as "binary memo" will DOE consider it for attachment relevance.

Employee_details node:



iv. Now activate the data object and create Backend Adapter for it.

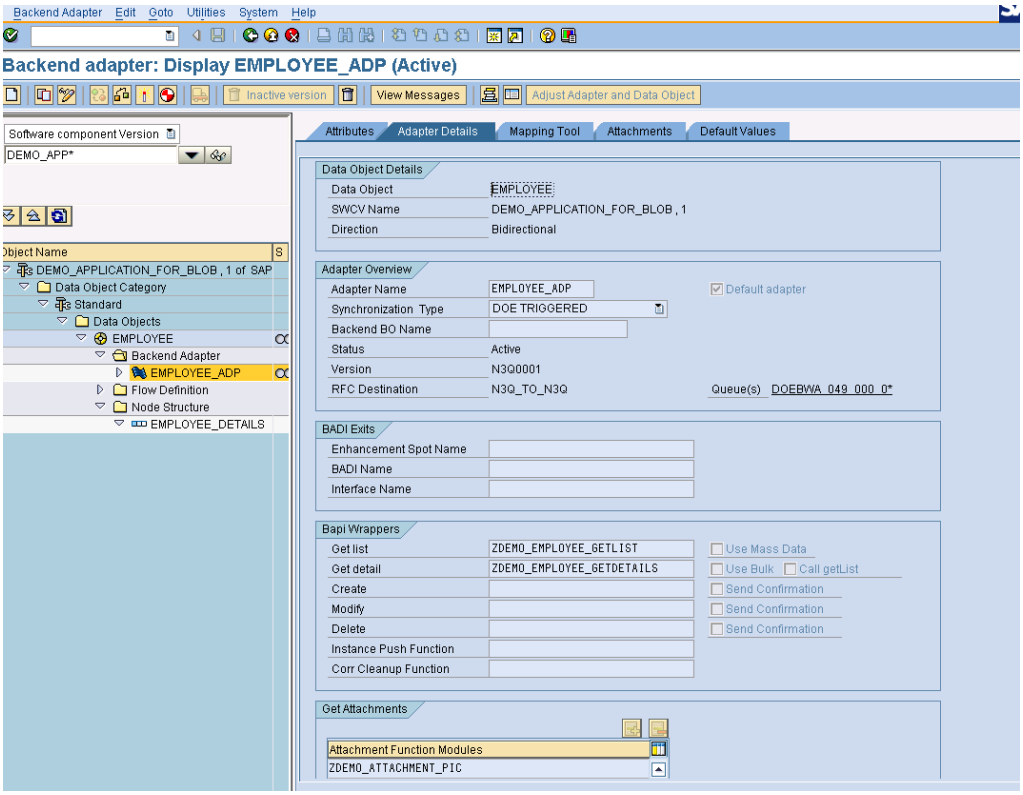
v. Here the activation of data object will create a different table for BLOB similar to CDS table.

vi. The name of the table will be of following pattern SDOE*_BM (and SDOE*_TM in case of text memo)

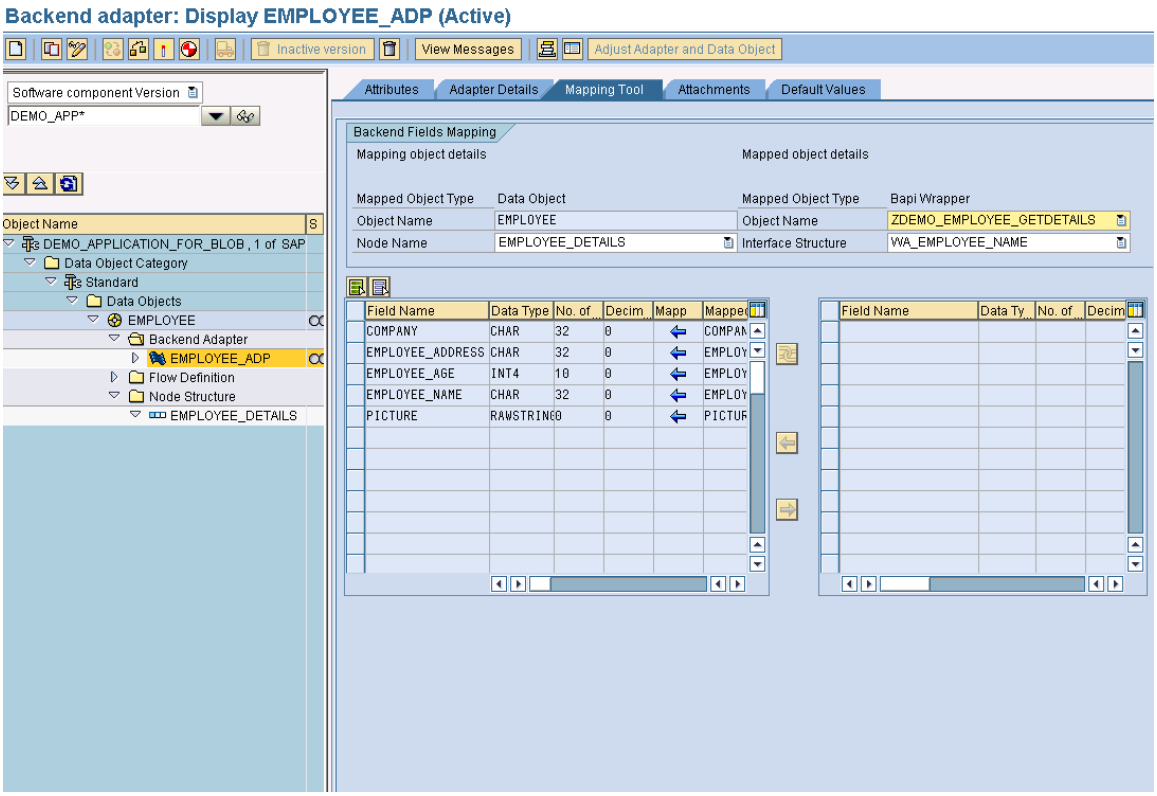
4. Backend Adapter:

i. Right click on active Data Object and create Backend adapter.

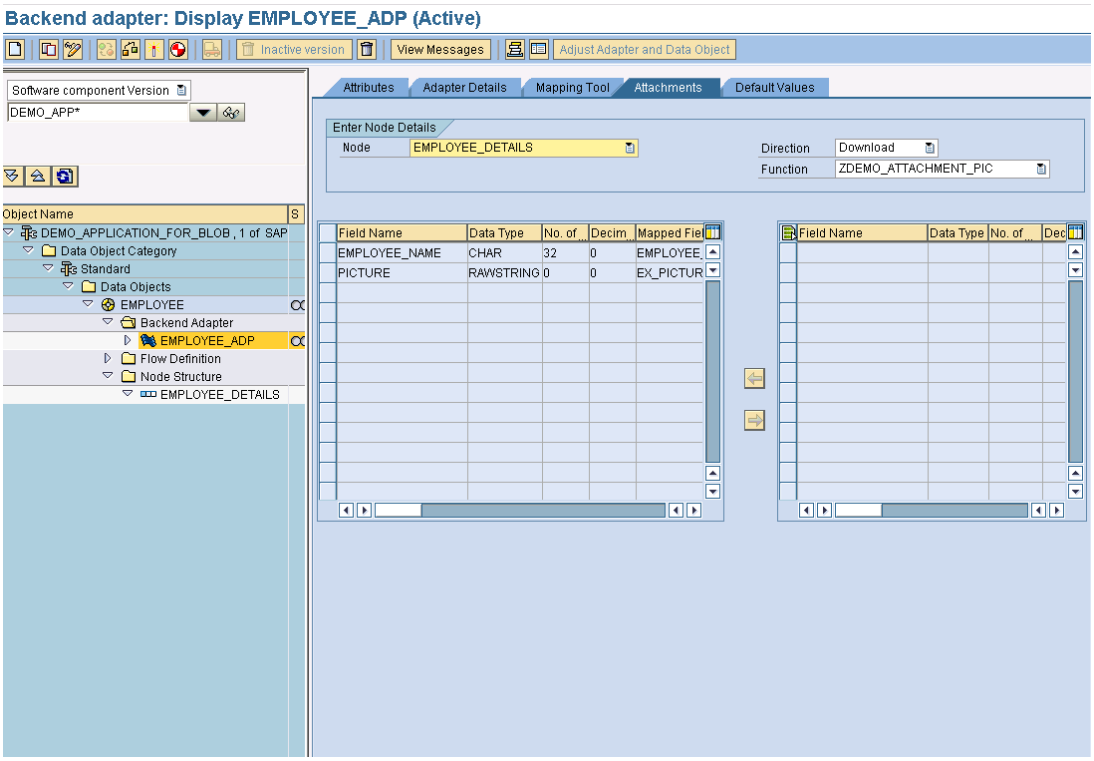
ii. Give the required details viz : name, synchronization type, bapiwrappers and Attachment function module.



iii. Now click on mapping tool tab and do the mapping. Here the field "picture" will not be mapped with automatic mapping because field "picture" in getdetails bapiwrapper is of type CHAR1.Therefore we will have to map it explicitly using left arrow button. The below picture represent this:



iv. Go to attachment tab and do the mapping for keys as well as for BLOB field with download attachment function module.



v. Activate the adapter.

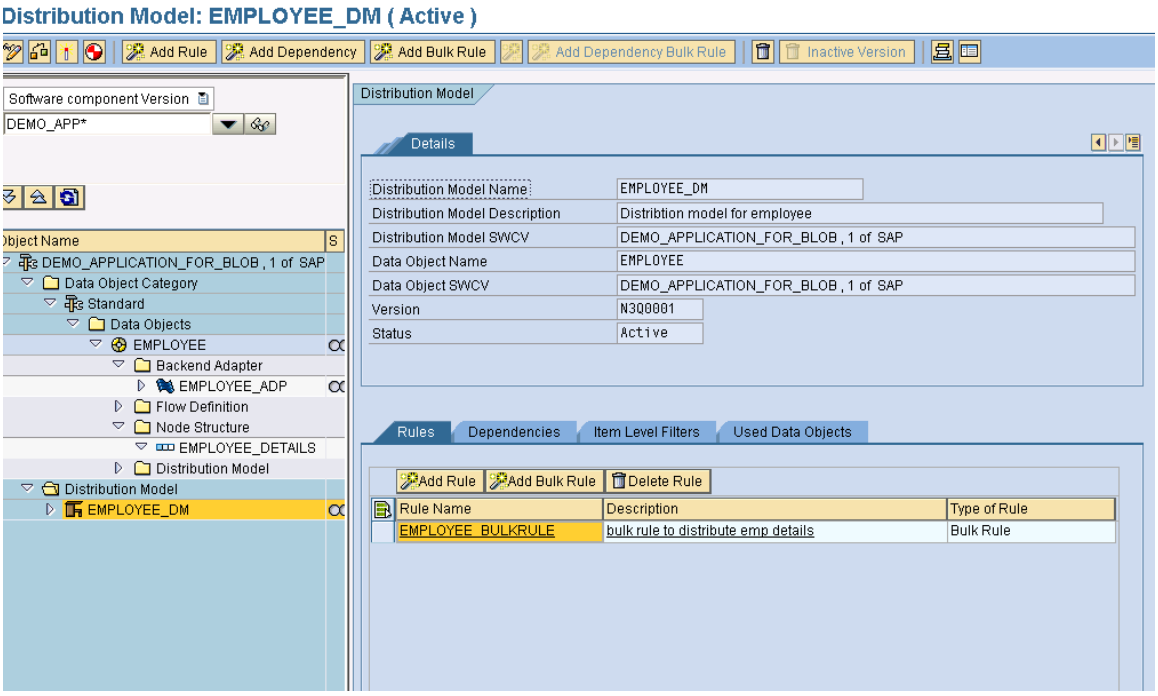
5. Distribution Model:

For my simple application I have defined a simple DM on the data object with a bulk rule to distribute the content to all the subscribing devices.

1- Create a Distribution model by right clicking on data object. Create a bulk rule to distribute it to all the devices.

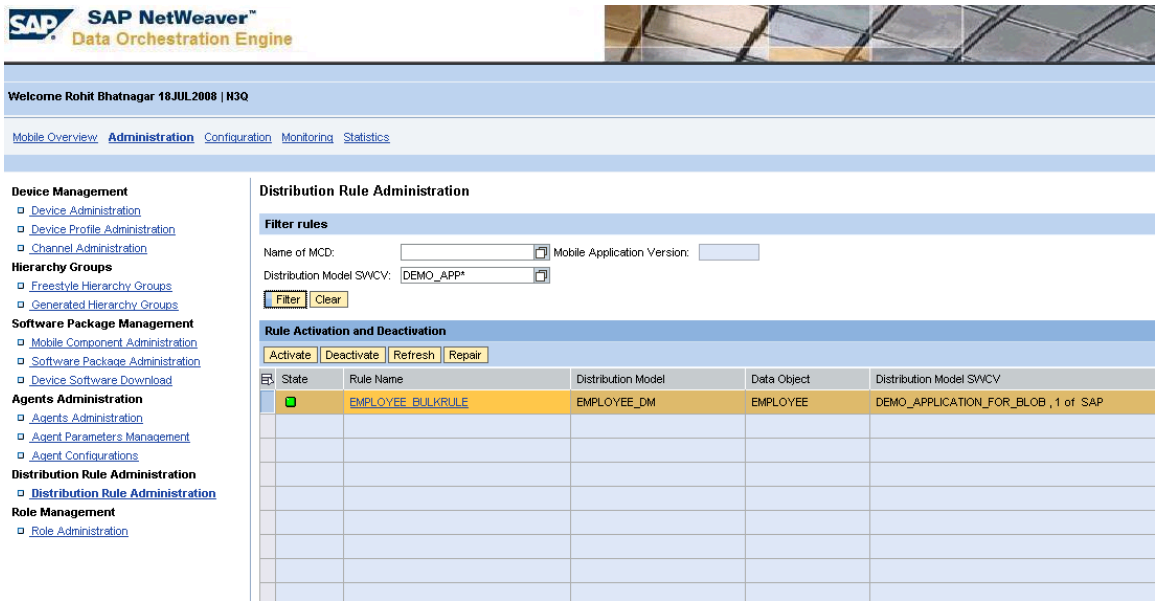
2- Activate the DM

3- The DM looks as following:



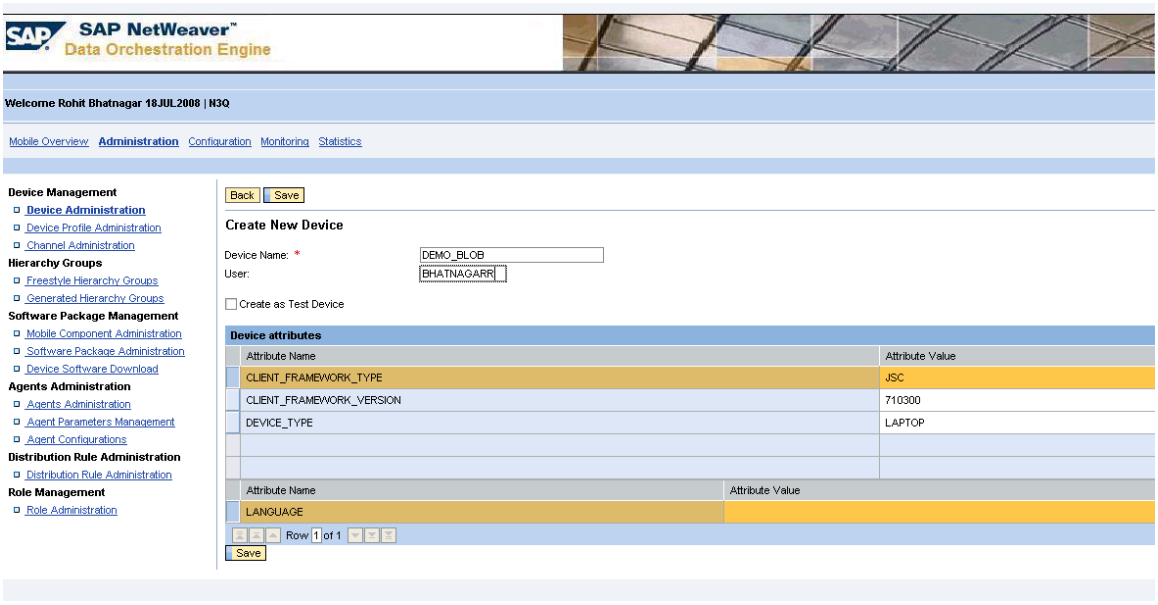
Activation of Rule:

1. Go to Admin portal
2. Click on Administration
3. Click on Distribution Rule Administration
4. Specify SWCV in Distribution Model SWCV
5. Click on filter
6. select the rule and activate it

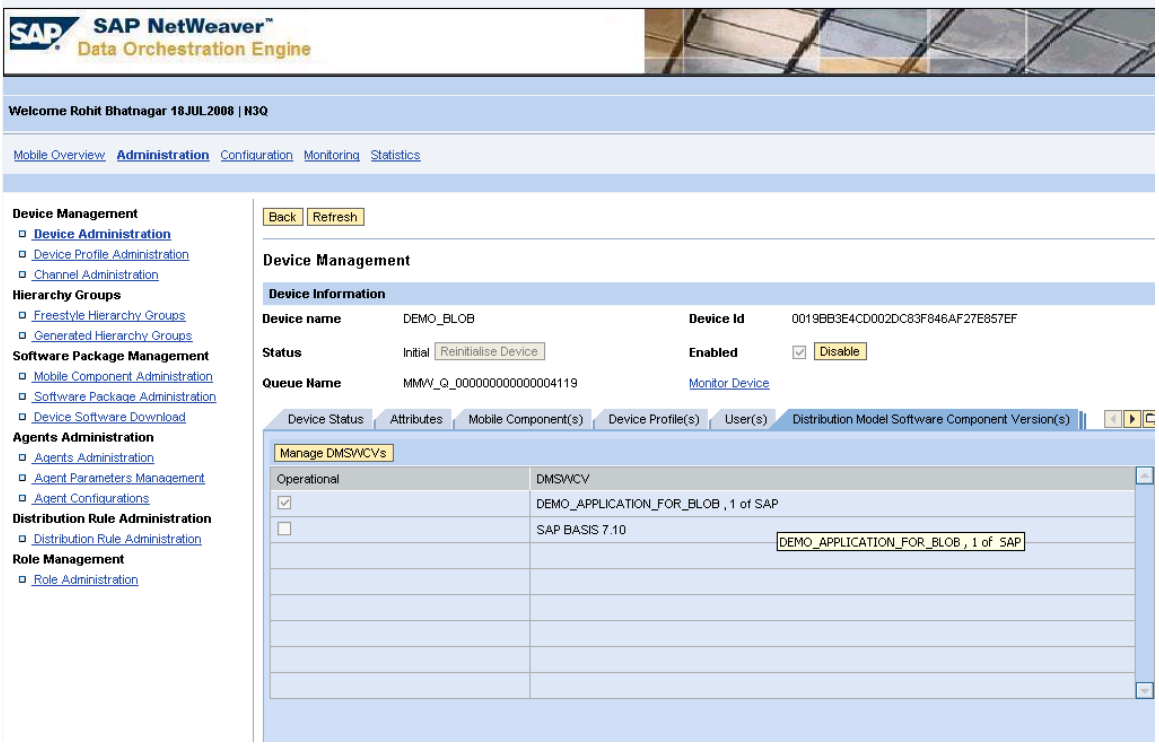


6. Device Creation

1. Click on device administration
2. Click on create device
3. specify the name and user name and save it



4. Click on manage device
5. Select distribution model software component version tab
6. Click on manage DMSWCV
7. Click on assign more DM SWCV
8. Select SWCV and save it
9. Click on enable to enable the device



10. Go to device status and trigger pending extract

11. In the device out bound queue, check the content of corresponding field of the respective message. Being binary the content may not be readable.

7. Performance optimization (Compare memo configuration)

This configuration parameter is used to compare memo fields explicitly even if backend says there is an update on memo fields this will be useful because download of full blob will not be needed in case of false update.

1. Go to configuration
2. click on backend configuration
3. select COMPARE_MEMO field
4. Insert SWCV and Data object name.
5. Click on save.

Related Content

For more information, visit the Mobile homepage.

Copyright