Extracting SAP ECC Compounding Hierarchies into SAP BI using Custom Datasource

Applies to:

SAP ECC 6.0, SAP BI 7.0. For more information, visit the EDW homepage

Summary

Standard Business Content extractors support main hierarchies (e.g., cost center, general ledger [G/L] account, material, plant, work center), but they do not support all hierarchies in the SAP ECC source system, such as Vendor Hierarchy (MKH1). In this case, SAP does not provide a standard Datasource to link ECC hierarchies to BW. This document guides you to extract compounding hierarchy such as Vendor compounded with Purchasing Organization.

Author: Srinivasa Raju B

Company: Deloitte Consulting India Private Ltd.

Created on: 18 July 2011

Author Bio

Srinivasa Raju B is currently working in Deloitte Consulting India Private Ltd. He is experienced in SAP BI, BO and ABAP.

© 2011 SAP AG

Table of Contents

Overview 3
Scenario 3
Configure the Extractor in ECC 4
Step 1: Enter transaction GS01 to create an empty set 4
Step 2: Use transaction BW07 to generate the Datasource for Customer Reference Set hierarchies 4
Step 3: Add ABAP code in the user exit EXIT_SAPLRSAP_004 5
Configure the Extractor in BW 5
Step 1: Prepare the Info Object to accept data from the Datasource 5
Step 2: Replicate the Datasource in SAP BI 6
Step 3: Assign the Datasource to the Infosource 6
Step 4: Create the Info Package and load the hierarchy data. 7
Function Module Code 8
Related Content 13
Disclaimer and Liability Notice 14

© 2011 SAP AG

Overview

While standard business content extractors support main hierarchies (e.g., cost center, general ledger [G/L] account, material, plant, work center), there are several other hierarchies in the ECC source system, such as vendor hierarchy (MKH1), that these extractors do not support. In this case, SAP does not provide a standard Datasource to link ECC hierarchies to the BW. This becomes even more complicated when there is a compounding to the hierarchy, (e.g., if vendor compounded with purchasing organization)

Scenario

The vendor hierarchy is maintained with transaction LFMH in ECC. This vendor hierarchy is based on the combination of vendor and purchasing organization. This example demonstrates the idea of compounding. Below, a small vendor hierarchy in transaction MKH1 is shown with a category called C for the sample data used in this article (Figure 1).

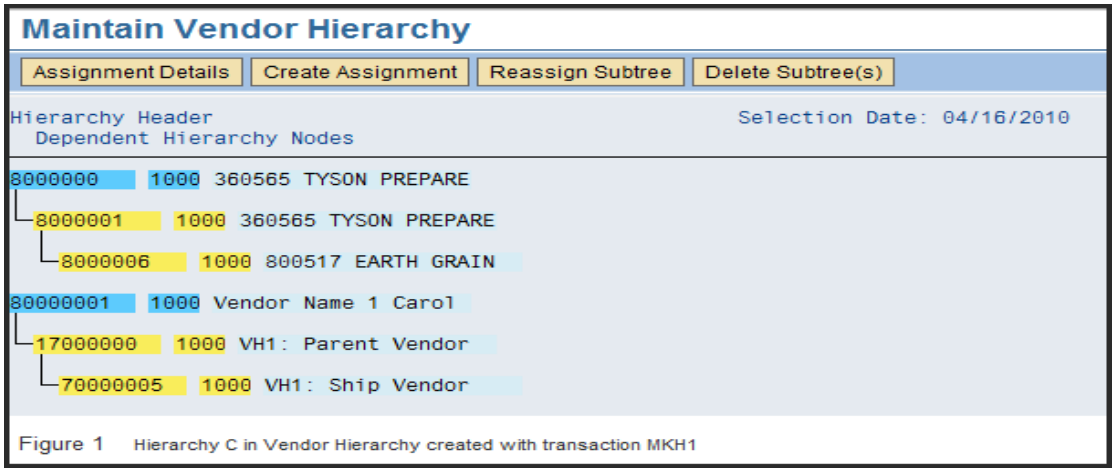
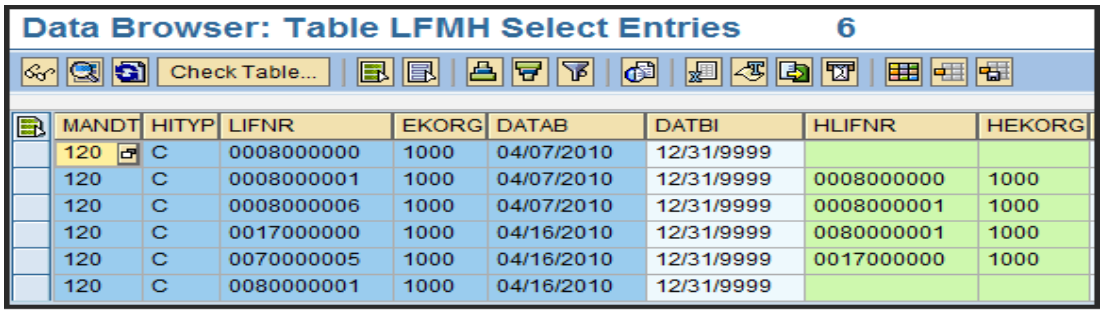


Figure 1 Hierarchy C in Vendor Hierarchy created with transaction MKH1

In the vendor hierarchy example, the hierarchy node data is stored in the table LFMH (Figure 2). In this table, each node of the hierarchy is stored in fields LIFNR and EKORG. Links to child nodes are available in the fields HLIFNR and HEKORG. For example, in Figure 2, Vendor 1000/0008000001 has a parent node 1000/0008000000. Those records are the tabular representation of the Hierarchy Category C created with MKH1 in Figure 1.



MANDT	HITYP	LIFNR	EKORG	DATAB	DATBI	HLIFNR	HEKORG
120	C	0008000000	1000	04/07/2010	12/31/9999		
120	C	0008000001	1000	04/07/2010	12/31/9999	0008000000	1000
120	C	0008000006	1000	04/07/2010	12/31/9999	0008000001	1000
120	C	0017000000	1000	04/16/2010	12/31/9999	0080000001	1000
120	C	0070000005	1000	04/16/2010	12/31/9999	0017000000	1000
120	C	0080000001	1000	04/16/2010	12/31/9999		

Figure 2 Table LFMH for Hierarchy Category C

We need to load this hierarchy into 0VEN_PURORG info object in SAP BI.

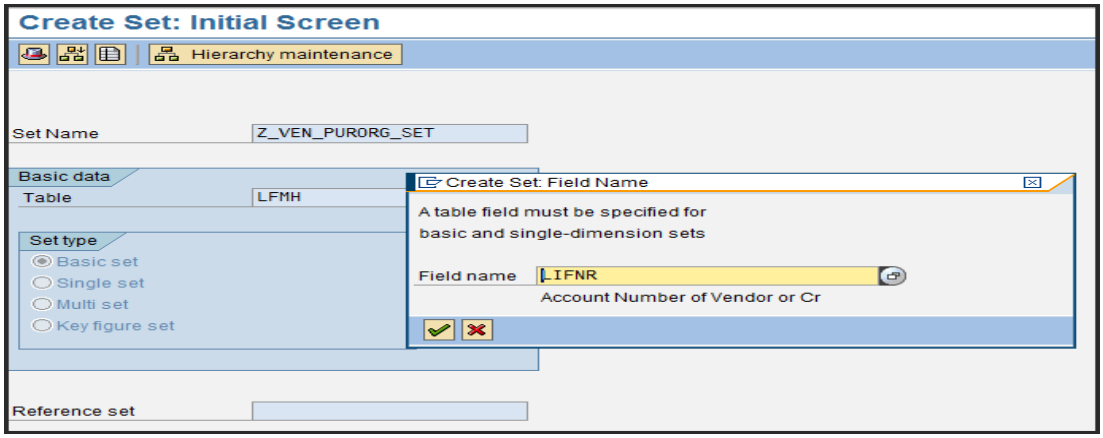
© 2011 SAP AG

Configure the Extractor in ECC

The following steps illustrate how the aforementioned workaround can be utilized. These steps include creating an empty hierarchy, generating the Datasource, and populating the Datasource with data from the LFMH table via EXIT_SAPLRSAP_004.

Step 1: Enter transaction GS01 to create an empty set

While any set name can be entered, theTable Name must be LFMH. Set type as basic set, and then press enter. In the pop-up window, select field name as LIFNR, press enter, and then click on the save icon.



Step 2: Use transaction BW07 to generate the Datasource for Customer Reference Set hierarchies

Enter LFMH in the table for set field and LIFNR in the field name. Because the hierarchy is being loaded to the compounded Info Object 0VEN_PURORG, the Datasource must be created by providing the data element for compounding which is EKORG in this example. Enter the Datasource name or you can also allow the system to generate the Datasource name. In this example, a custom name ZDS_VEN_PURORG_HIER is entered for the Datasource (refer the Figure 3 below). Then generate the Datasource by executing F8.

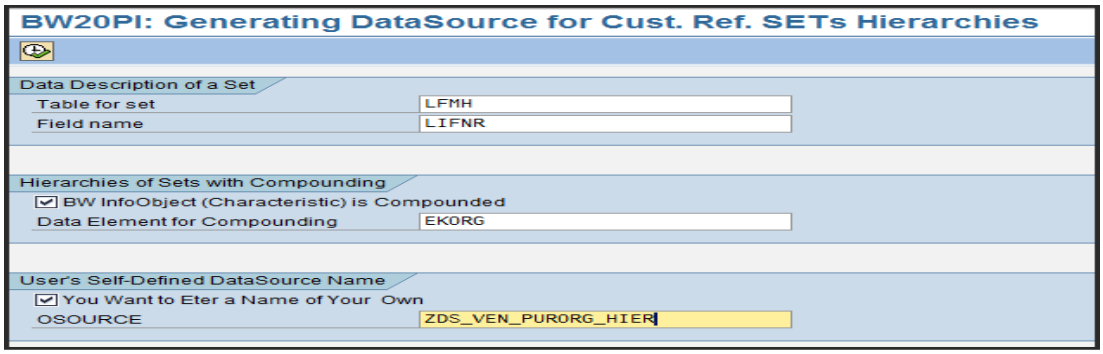
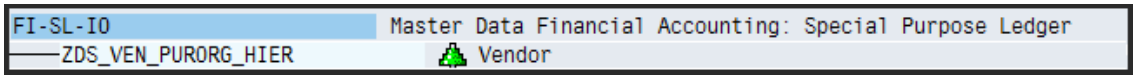


Figure 3: Generate Data source using transaction BW07

To ensure that a Datasource is generated correctly, use the Datasource browser (transaction code RSA6) to display it. As shown below, ZSD_VEN_PURORG_HIER is found via menu path SAP-R/3>FI>FI-SL>FI-SL-IO.



© 2011 SAP AG

Step 3: Add ABAP code in the user exit EXIT_SAPLRSAP_004

This coding allows the user to populate the Datasource created in step 2. If this exit is not active, transaction CMOD can be used to make it active. This process is well documented on the website, help.sap.com. The document is titled, “Enhancements to the SAP Business Information Warehouse.”

To access the user exit to enter the ABAP code, go into transaction SE38 and enter ZXRSAU04 in the Program field. Click on the *Change* button to enter the ABAP code given below:

```
CASE i_datasource.

  WHEN 'ZDS_VEN_PURORG_HIER'.

    CALL FUNCTION 'ZFM_VEN_PURORG_HIER_TRANSFER'
      EXPORTING
        I_S_HIEBAS    = I_S_HIEBAS
        I_S_HIEFLAG   = I_S_HIEFLAG
        I_S_HIERSEL   = I_S_HIER_SEL
      TABLES
        I_T_LANGU     = I_T_LANGU
        E_T_HIETEXT   = C_T_HIETEXT
        E_T_HIENODE   = C_T_HIENODE
        E_T_FOLDERT   = C_T_FOLDERT
        E_T_HIEINTV   = C_T_HIEINTV.

ENDCASE.
```

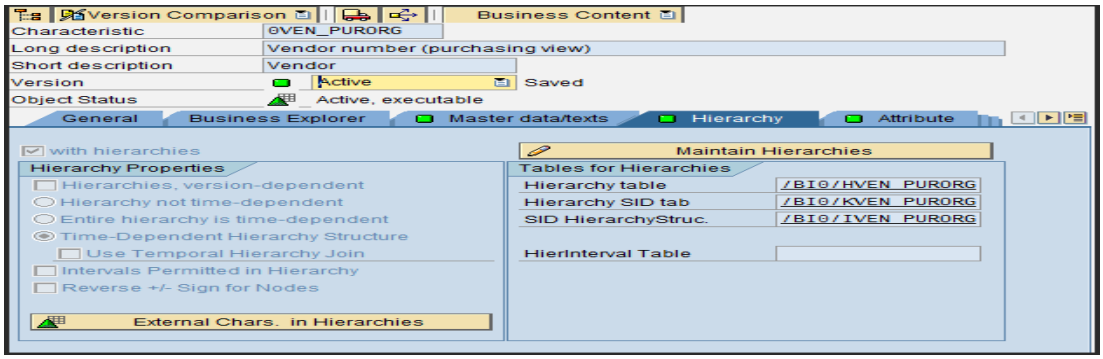
In the above, a function module, “ZFM_VEN_PURORG_HIER_TRANSFER “, has been called. Please find the function module code at the end.

Configure the Extractor in BW

Upon finishing the extractor on the ECC side, the BW portion will need to be configured to load the hierarchy into a BW Info Object. This section explains how to replicate the created Datasource in the BW and assign it to the Infosource that feeds the Info Object where the hierarchy is assigned. The section then illustrates how to create an Info Package from the Datasource, select the hierarchy, and load it into the BW, where the hierarchy will then be available for reporting.

Step 1: Prepare the Info Object to accept data from the Datasource

In BW, go to Administrator Workbench via RSA1 and find the desired Info Object; in this case, 0VEN_PURORG. On the Hierarchy tab, make sure the check box with hierarchies is selected. The Info Object 0VEN_PURORG is ready to hold the data coming from the Datasource.

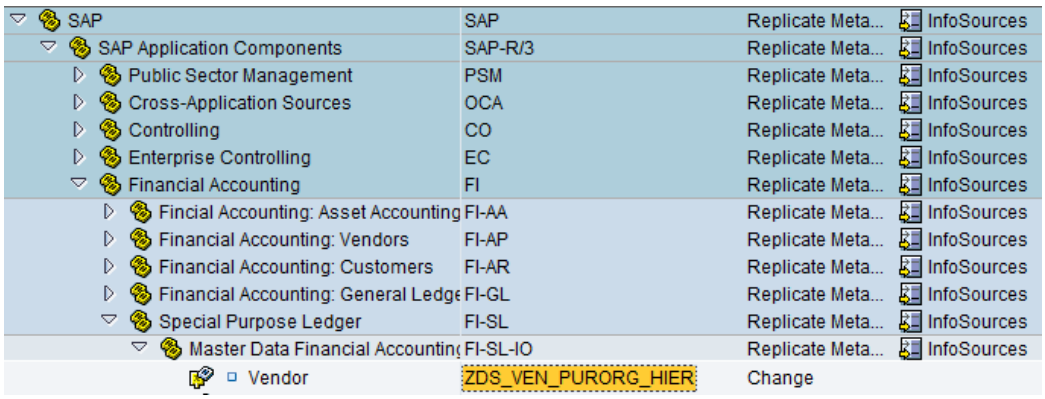


© 2011 SAP AG

Step 2: Replicate the Datasource in SAP BI

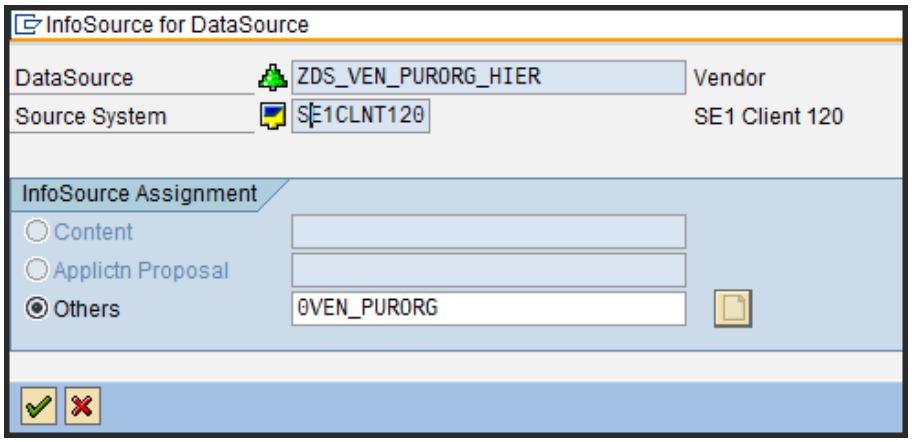
Click on source systems in the RSA1 transaction. Find the desired source system in the pane on the right side, right-click on it, and select Datasource Overview from the pop-up menu.

After completing the above, follow the path, SAP Application Components>Financial Accounting>Special Purpose Ledger> Master Data Financial Accounting: Special Purpose Ledger. All Datasources generated from the BW07 transaction in ECC fall under the Master Data Financial Accounting: Special Purpose Ledger node. Right-click on Master Data Financial Accounting: Special Purpose Ledger to perform Replicate Datasources. ZDS_VEN_PURORG_HIER appears under Data Financial Accounting: Special Purpose Ledger after the replication is complete.



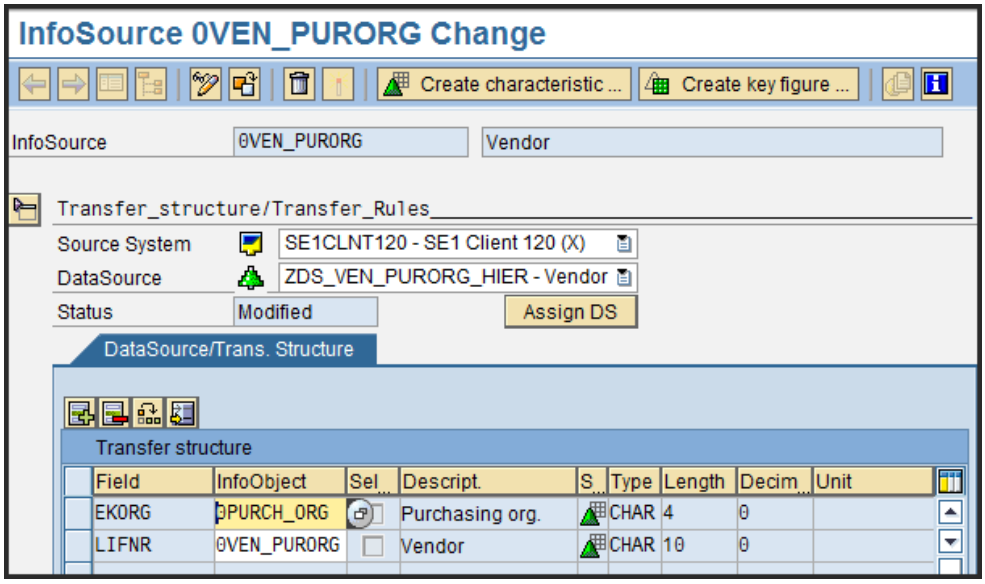
Step 3: Assign the Datasource to the Infosource

Click on the icon with a dash next to ZDS_VEN_PURORG_HIER in the screen shot below to assign it to an Infosource. Because the Infosource name is usually the same as the Info Object name, label the Infosource 0VEN_PURORG by entering 0VEN_PURORG in the 'Others' field. Click on the continue icon. Select 'Yes' on the pop-up screen to complete the assignment.



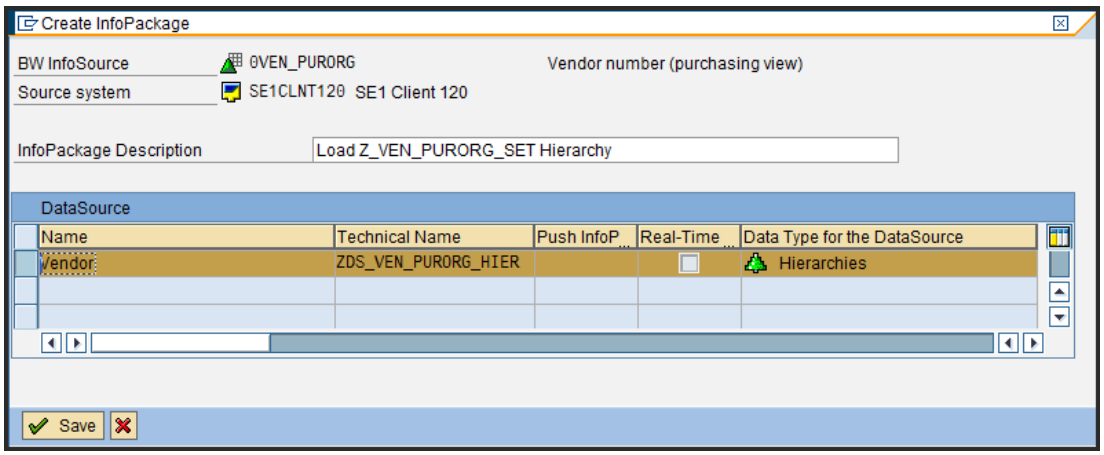
On the Transfer Rules screen that appears below, in the Info Object column, type the Info Object name, "0VEN_PURORG" in the field to the right of the LIFNR field. Type its compounding Info object's name, "0PURCH_ORG" in the field to the right of the EKORG field.

© 2011 SAP AG

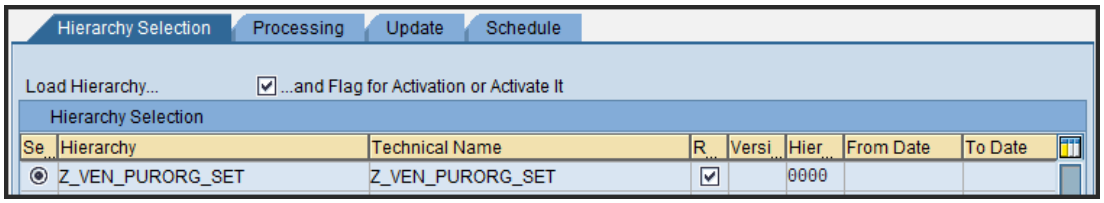


Step 4: Create the Info Package and load the hierarchy data.

Enter transaction RSA1. Under Infosource, find the Infosource 0VEN_PURORG. The Datasource ZDS_VEN_PURORG_HIER is now available under the ECC source system where the Datasource was created. Right-click on the ECC source system and select Create Info Package. Add a description of the Info Package in the Info Package Description field, as shown below, and click on the Save button.



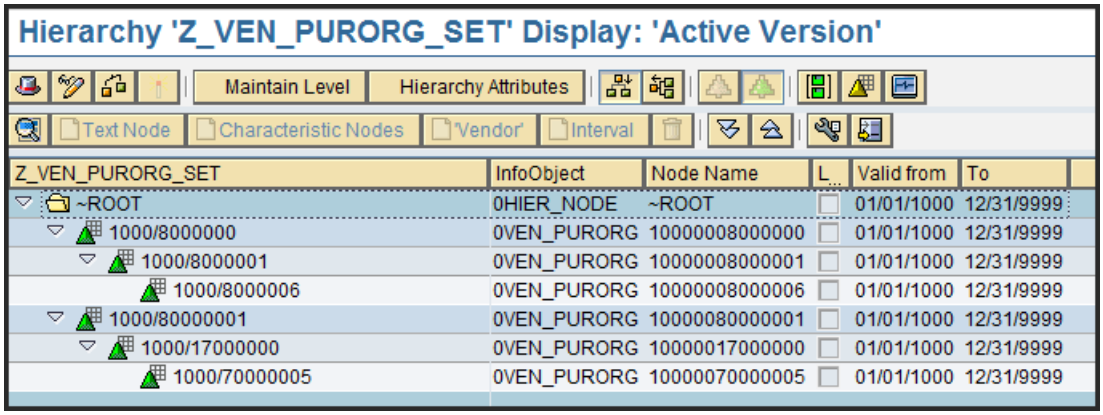
After clicking on the Save button, the Maintain Info Package screen will appears (shown below). On the Hierarchy Selection tab, select the hierarchy Z_VEN_PURORG_SET.



No other options need to be changed. The Load Hierarchy...and Flag for Activation or Activate It check box is checked by default. Select the Schedule tab and click on the Start button to load the hierarchy from the source system. Like any other Datasource, the load with the monitor icon can be checked.

© 2011 SAP AG

Once the load is successful, the Z_VEN_PURORG_SET hierarchy can be viewed from transaction RSH1 by double-clicking on it from the list. The icon depicting a tree can also be clicked in the toolbar of the monitor to display the hierarchy once the load is completed. The result is shown below:



Z_VEN_PURORG_SET	InfoObject	Node Name	L...	Valid from	To
~ROOT	0HIER_NODE	~ROOT		01/01/1000	12/31/9999
1000/8000000	0VEN_PURORG	10000008000000		01/01/1000	12/31/9999
1000/8000001	0VEN_PURORG	10000008000001		01/01/1000	12/31/9999
1000/8000006	0VEN_PURORG	10000008000006		01/01/1000	12/31/9999
1000/80000001	0VEN_PURORG	10000080000001		01/01/1000	12/31/9999
1000/17000000	0VEN_PURORG	10000017000000		01/01/1000	12/31/9999
1000/70000005	0VEN_PURORG	10000070000005		01/01/1000	12/31/9999

The hierarchy shown above depicts the ECC side from transaction MKH1. Now, the Z_VEN_PURORG_SET hierarchy can be included in any reports using the Info Object 0VEN_PURORG. The description of the nodes will not be present on the hierarchy. To load them, another text Datasource must be added and loaded into 0VEN_PURORG. This is because the texts are stored into a different table.

Function Module Code

```
FUNCTION ZFM_VEN_PURORG_HIER_TRANSFER.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(I_S_HIEBAS) TYPE  RSAP_S_HIEBAS
*"     VALUE(I_S_HIEFLAG) TYPE  RSAP_S_HIEFLAG
*"     VALUE(I_S_HIERSEL) TYPE  RSAP_S_HIER_LIST
*"  TABLES
*"      I_T_LANGU TYPE  SBIWM_T_LANGU
*"      E_T_HIETEXT TYPE  RSAP_T_HIETEXT
*"      E_T_HIENODE TYPE  RSAP_T_HIENODE
*"      E_T_FOLDERT TYPE  RSAP_T_FOLDERT OPTIONAL
*"      E_T_HIEINTV TYPE  RSAP_T_HIEINTV OPTIONAL
*"  EXCEPTIONS
*"      INVALID_HIERARCHY_FLAG
*"      APPLICATION_ERROR
*"----------------------------------------------------------------------

  DATA: lt_hiers       LIKE rshiertrsf OCCURS 0 WITH HEADER LINE,
        lt_nodetab     LIKE snodetext  OCCURS 0  WITH HEADER LINE,
        lv_linked_node LIKE roshienode-link.

*---- Define application data
  DATA: lt_hierarchies TYPE hierarchies,
        ls_hierarchy   LIKE LINE OF lt_hierarchies.

  gv_datefrom = i_s_hiersel-datefrom.
  gv_dateto = i_s_hiersel-dateto.
```


© 2011 SAP AG

```
*---- Get text of hierarchie
  IF i_s_hieflag-hietabfl IS INITIAL. "// Info object have to have hier.
    RAISE invalid_hierarchy_flag.
  ENDIF.
  gv_this_info_object = '0VEN_PURORG'. "i_s_hiebas-chabasnm.

  CLEAR: e_t_hietext, e_t_hietext[].
*---- Set top node and read vendor hierarchy into node table
* // Set first node
  CLEAR: e_t_hienode, e_t_hienode[].
  CLEAR: e_t_foldert, e_t_foldert[].

  e_t_hienode-iobjnm   = '0HIER_NODE'.
  e_t_hienode-nodename = '~ROOT'.
  e_t_hienode-tlevel = 1.
  APPEND e_t_hienode.
* // Get texts of root
  LOOP AT i_t_langu where langu = 'E'.
    CLEAR e_t_foldert.
    MOVE i_t_langu-langu   TO e_t_foldert-langu.
    MOVE i_t_langu-langu   TO e_t_hietext-langu.
    MOVE '0HIER_NODE'      TO e_t_foldert-iobjnm.
    MOVE '~ROOT'           TO e_t_foldert-nodename.

    SELECT SINGLE * FROM TLHITT WHERE spras EQ 'E'
                                AND hityp EQ 'C'.
    IF sy-subrc EQ 0.
      e_t_hietext-txtsh = tlhitt-vtext.
      e_t_hietext-txtmd = tlhitt-vtext.
      e_t_hietext-txtlg = tlhitt-vtext.
      MOVE e_t_hietext-txtsh TO e_t_foldert-txtsh.
      MOVE e_t_hietext-txtmd TO e_t_foldert-txtmd.
      MOVE e_t_hietext-txtlg TO e_t_foldert-txtlg.
    ENDIF.
    APPEND e_t_foldert.
    APPEND e_t_hietext.
  ENDLOOP.

*---- If Datasource time-independent -> Take current date
  IF i_s_hieflag-hiendtfl IS INITIAL.
    gv_datefrom = sy-datum.
    gv_dateto = sy-datum.
  ENDIF.

*---- Set selection date
  IF gv_dateto IS INITIAL.
    gv_dateto = '99991231'.
  ENDIF.

*---- Get customer hierarchy
  SELECT * FROM lfmh INTO TABLE lt_hierarchies
                     WHERE hityp  EQ 'C'
                     AND   hlifnr EQ space
                     AND  ( ( datab <= gv_dateto ) AND
                            ( datbi >= gv_datefrom ) ).
*  IF sy-subrc NE 0.
```


© 2011 SAP AG

```
*    RAISE hierarchy_tab_not_found.
*  ENDIF.
  LOOP AT lt_hierarchies INTO ls_hierarchy
                         WHERE hlifnr EQ space.  "// Get all roots
    PERFORM setbwnode TABLES   e_t_hienode
                      USING    ls_hierarchy '2'
                      CHANGING lv_linked_node.
    check lv_linked_node eq space.
    PERFORM getchildren TABLES e_t_hienode
                        USING  ls_hierarchy '2'.
  ENDLOOP.
*---- Set tlevel of hierarchy table for next call
  LOOP AT e_t_hienode.
    IF i_s_hieflag-hiendtfl IS INITIAL.
      clear e_t_hienode-datefrom.
      clear e_t_hienode-dateto.
    ENDIF.
    MOVE e_t_hienode-nodename  TO lt_nodetab-name.
    MOVE e_t_hienode-tlevel    TO lt_nodetab-tlevel.
    MOVE e_t_hienode-link      TO lt_nodetab-link.
    e_t_hienode-nodeid = sy-tabix.
    MODIFY e_t_hienode.
    APPEND lt_nodetab.
  ENDLOOP.
*---- Get index data for parent child next depending on tlevel
  CALL FUNCTION 'RS_TREE_CONSTRUCT'
       TABLES
            nodetab            = lt_nodetab
       EXCEPTIONS
            tree_failure       = 1
            id_not_found       = 2
            wrong_relationship = 3
            OTHERS             = 4.
  IF sy-subrc NE 0.
    RAISE application_error.
  ENDIF.
*---- Pass data to originale table
  DATA: l_hlp_sy_tabix LIKE sy-tabix.
  LOOP AT lt_nodetab.
    l_hlp_sy_tabix = sy-tabix.
    READ TABLE e_t_hienode INDEX l_hlp_sy_tabix.
    CHECK sy-subrc = 0.
    e_t_hienode-link      =  lt_nodetab-link.
    e_t_hienode-parentid  =  lt_nodetab-parent.
    e_t_hienode-childid   =  lt_nodetab-child.
    e_t_hienode-nextid    =  lt_nodetab-next.
    MODIFY e_t_hienode INDEX l_hlp_sy_tabix.
  ENDLOOP.
ENDFUNCTION.
```


© 2011 SAP AG

Declare Global data in the Global Include of the function module

```
function-pool vhbw.                    "MESSAGE-ID ..

TYPE-POOLS: rhie.

INCLUDE rmcsionm.

* Specific types and data
TABLES: tlhitt, thitt, lfmh.

TYPES: hierarchies LIKE SORTED TABLE OF lfmh
                   WITH NON-UNIQUE KEY hlifnr HEKORG,
       hierarchy   TYPE LINE OF hierarchies.

DATA: gv_this_info_object LIKE rodchabas-chabasnm.
*     Increment nodeid
DATA: gv_nodeid LIKE ROSHIENODE-NODEID.
DATA: gv_datefrom LIKE roshiersel-datefrom.
DATA: gv_dateto LIKE roshiersel-dateto.
DATA: BEGIN OF GS_NODE_IDX,
         nodename LIKE roshienode-nodename,
         datefrom LIKE roshienode-datefrom,
       END OF GS_NODE_IDX,
       GT_NODE_IDX LIKE HASHED TABLE OF GS_NODE_IDX
          WITH UNIQUE KEY nodename
                          datefrom.
DEFINE next_nodeid.
  add 1 to gv_nodeid.
  move gv_nodeid to &1.
END-OF-DEFINITION.
*&---------------------------------------------------------------------*
*&      Form  SETBWNODE
*&---------------------------------------------------------------------*
*       Maps nodo into Bw hierarchy format
*----------------------------------------------------------------------*
*      <->PTR_HIENODE   BW Nodes
*      -->PSI_HIERARCHY Actual node
*      -->PVI_LEVEL     Node level
*----------------------------------------------------------------------*
FORM setbwnode TABLES ptr_hienode          TYPE rsap_t_hienode
               USING  value(psi_hierarchy) TYPE hierarchy
                      value(pvi_level)     TYPE rsap_s_hienode-tlevel
               CHANGING value(pvc_link)    LIKE roshienode-link.
  STATICS: ls_hienode LIKE LINE OF ptr_hienode,
           BEGIN OF ls_nodename,
             ekorg LIKE psi_hierarchy-ekorg,
             lifnr like psi_hierarchy-lifnr,
           END OF ls_nodename.
* ls_hienode-nodeid   = psi_hierarchy-kunnr.
  next_nodeid ls_hienode-nodeid.
* Fill node name with attributes in following order
  ls_nodename-lifnr   = psi_hierarchy-lifnr.
  ls_nodename-ekorg   = psi_hierarchy-ekorg.
  ls_hienode-nodename = ls_nodename.
  ls_hienode-tlevel   = pvi_level.
```


© 2011 SAP AG

```
  ls_hienode-iobjnm   = gv_this_info_object.
  ls_hienode-datefrom = psi_hierarchy-datab.
  ls_hienode-dateto   = psi_hierarchy-datbi.
* Node already exist ?
*    { Mark this node as linked node and read no childs } : read childs
  READ TABLE GT_NODE_IDX TRANSPORTING NO FIELDS
         WITH TABLE KEY nodename = ls_nodename
                 datefrom = ls_hienode-datefrom.
  IF sy-subrc EQ 0.
    pvc_link = ls_hienode-link = 'X'.
  else.
    clear: pvc_link, ls_hienode-link.
    gs_node_idx-nodename = ls_nodename.
    gs_node_idx-datefrom = ls_hienode-datefrom.
    insert gs_node_idx into table gt_node_idx.
  ENDIF.
  APPEND ls_hienode TO ptr_hienode.
ENDFORM.                              " SETBWNODE
*&---------------------------------------------------------------------*
*&      Form  GETCHILDREN
*&---------------------------------------------------------------------*
*       Get all child nodes
*----------------------------------------------------------------------*
*      <->PTR_HIENODE      Nodes in BW Format
*      -->PSI_NODE         Actual node
*      -->PVI_LEVEL        Deep
*----------------------------------------------------------------------*
FORM GETCHILDREN TABLES ptr_hienode      TYPE rsap_t_hienode
                 USING  value(psi_node)  TYPE hierarchy
                        value(pvi_level) TYPE rsap_s_hienode-tlevel.
  DATA: lv_level       LIKE pvi_level,
        lv_linked_node LIKE roshienode-link.
*---- Define application data
  DATA: lt_hierarchies TYPE hierarchies,
        ls_hierarchy   LIKE LINE OF lt_hierarchies.
*---- Get customer hierarchy
  SELECT * FROM lfmh INTO TABLE lt_hierarchies
                     WHERE hityp  EQ psi_node-hityp
                     AND   hlifnr EQ psi_node-lifnr
                     AND   hekorg EQ psi_node-ekorg
                     AND   ( ( datab <= psi_node-datbi ) AND
                           ( datbi >= psi_node-datab ) )
                     AND   ( ( datab <= gv_dateto ) AND
                            ( datbi >= gv_datefrom ) ).
  lv_level = pvi_level + 1.
  LOOP AT lt_hierarchies INTO ls_hierarchy.
    PERFORM setbwnode TABLES   ptr_hienode
                      USING    ls_hierarchy lv_level
                      CHANGING lv_linked_node.
    check lv_linked_node eq space.
    PERFORM getchildren TABLES ptr_hienode
                        USING  ls_hierarchy lv_level.
  ENDLOOP.
ENDFORM.                              " GETCHILDREN
```


© 2011 SAP AG

Related Content

Hierarchies

Enhancements to the SAP Business Information Warehouse

For more information, visit the EDW homepage

© 2011 SAP AG

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.

© 2011 SAP AG