Applies to:
SAP BW 3.5 and SAP BI NW 7.0. For more information, visit the Business Intelligence homepage.

Summary
The purpose of this paper is to provide the procedure on how to extract BOM header and BOM item details into BW. This development is done to make available the BOM data which can further be used in lookups to enable Component level reporting.

Author: Dhanya A
Company: Satyam Computers Services Ltd.
Created on: 10th October 2008

Author Bio
Dhanya A is a SAP BI Consultant at Satyam Computer Services Limited, Hyderabad - India. She is working in BI space for past 4 years and has rich experience in Enterprise DataWarehousing, Enterprise Reporting and Performance Tuning for solutions to Retail, Utilities, Finance industry. She has experience in BW-BPS implementation of FI-AA and Microsoft Integration with SAP BI.
Abstract

The purpose of this paper is to describe the Custom BW Extractors which will incorporate the logic of extracting the BOM header details and BOM item (component) details from R/3. This specification is developed in BW to make available the BOM data which can further be used in lookups to enable Component level reporting.

This extracted data when loaded to the DSO can be used to perform Material Bill Of Material (BOM) explosion wherever required.

Business Scenario

In BW, the client had Sales data available at Kit level (Material) through CO-PA extraction - actual and plan DataSource. But, there was no component level information in BW.

They needed to report on component level information like the Quantity, UOM, MRP Controller, Quota Arrangement usage etc.

They required the Kit level information to be exploded to component level information and displayed.

Challenges:

- There was no master data available for BOM details.
- 2LIS_04_P_COMP does provide material conception at Component level, but it is only focused on Production order and explodes the BOM used in Production order.
- The option of getting the MAST, STKO and STPO Table entries to BW and then writing ABAP Code in BI to get item details was not a viable solution as there was no Function Module in BI which would generate the components based on material and BOM combination.
- There was a challenge to get Valid To date of a component as it was not stored in any Base tables in ECC.

Solution Benefits:

- Using this design, we can get the BOM Header and item level details readily in the BW system. This can be further looked up in the transformations to perform BOM explosion to split Material level information to component level information based on valid time interval
- This BOM data in the DSO can be used across applications to report at component level data.
- This minimized coding on the BI side too.
- The ABAP code can be used as a reference, whenever we need to get the material’s components, its quantity and validity period.
- To perform BOM explosion on transactional records.
Development Specification

BOM Header DataSource

Create a view based on the tables MAST and STKO in ECC5.0 System. The joining conditions for the table are:

- MAST-MANDT = STKO-MANDT
- MAST-STLNR = STKO-STLNR
- MAST-STLAL = STKO-STLAL

Select the fields of the view from these tables based on business requirement.

Create a new BW Generic DataSource of type Master Data and assign it to view created above.

Provide selections for this DataSource based on business requirement.

Design Overview:

BOM Item DataSource

Create a function module in ECC system which has the function module RSAX_GET_DATA_SIMPLE as the template.

Algorithm of the Function module:

- Selections to be included are Plant, Alternative BOM, BOM Usage and material.
- Among these selections Plant needs to be entered. If there is no specific value enter (0001 to 9999).
- Select all the relevant materials from MAST table into internal table for the selections given above.
- Call the FM CSAP_MAT_BOM_ITEM_SELECT for each material.
- This FM will give all the Components, its Qty, Its UOM, Valid From and Valid to Date for each material passed.
- Include the logic for conversions to different data types like DATE and QUAN.

The ABAP source code for this FM is in Appendix A.

Create a new Generic DataSource of type Master data and assign it to application component PP. This Generic DataSource which will use the above function module to extract data.
Design Overview:

The first process in BI is to replicate Generic DataSources in BW in application component PP. Create a standard DSO which will have the BOM Header Generic DataSource as the source. Create another standard DSO which will have the BOM Item Generic DataSource as the source. This BOM data when loaded to the DSO can be used across applications to report at component level data, to perform BOM explosion to component data.
Design Overview

**YPcS BOM HDR:** (Data Source Based on View)
- **MATNR** Material Number
- **WERKS** Plant
- **STLAN** BOM Usage
- **STLAN** Bill of material
- **STLAL** Alternative BOM
- **STLTY** BOM category
- **STKZ** Internal counter
- **DATUV** Valid-From Date
- **BMEIN** Base UCM for BOM
- **BMENG** Base quantity

### InfoPackage Selection
- **MATNR** Material Number (O)
- **WERKS** Plant
- **STLAN** BOM Usage = 3
- **STLAL** Alternative BOM = 1
- **STLTY** BOM category = M
- **STKZ** Bill of material

0FISCVARNT = KO' (Constant)

YVALIDWKF = CalcFrom (Valid-From Date)

YVALIDMF = CalcFrom (Valid-From Date)

YVALIDMT = CalcFrom (Valid To Date)

**YPcS BOM ITM:** (Data Source Based on FM)
- **MATNR** Material Number
- **WERKS** Plant
- **STLAN** BOM Usage
- **STLAN** Bill of material
- **STLAN** Alternative BOM
- **STLAN** BOM category
- **STLAN** Internal counter
- **DATUV** Valid-From Date
- **FDNRM** BOM component
- **POSTP** Item category (BOM)
- **POSNR** BOM item number
- **MENG** Component quantity
- **MINS** Component UCM

0FISCVARNT = KO' (Constant)

YVALIDWKF = CalcFrom (Valid-From Date)

YVALIDMF = CalcFrom (Valid-From Date)

YVALIDMT = CalcFrom (Valid To Date)

### InfoPackage Selection
- **MATNR** Material Number
- **WERKS** Plant
- **STLAN** BOM Usage = 0001 To 9999
- **STLAN** Bill of material = 3
- **STLAN** Alternative BOM = 1
- **VALID_FROM** Valid To
FUNCTION YCPS_BOM_ITM.

**"----------------------------------------------------------------------
**"  IMPORTING
**"     REFERENCE(I_REQUNR) TYPE  SRSC_S_IF_SIMPLE-REQUNR
**"     REFERENCE(I_DSOURCE) TYPE  SRSC_S_IF_SIMPLE-DSOURCE OPTIONAL
**"     REFERENCE(I_MAXSIZE) TYPE  SRSC_S_IF_SIMPLE-MAXSIZE OPTIONAL
**"     REFERENCE(I_INITFLAG) TYPE  SRSC_S_IF_SIMPLE-INITFLAG OPTIONAL
**"     REFERENCE(I_READ_ONLY) TYPE  SRSC_S_IF_SIMPLE-READONLY OPTIONAL
**"  TABLES
**"     I_T_SELECT TYPE  SRSC_S_IF_SIMPLE-T_SELECT OPTIONAL
**"     I_T_FIELDS TYPE  SRSC_S_IF_SIMPLE-T_FIELDS OPTIONAL
**"     E_T_DATA STRUCTURE  YCPS_BOM_STR OPTIONAL
**"  EXCEPTIONS
**"     NO_MORE_DATA
**"     ERROR_PASSED_TO_MESS_HANDLER
**"----------------------------------------------------------------------

* Auxiliary Selection criteria structure
DATA: L_S_SELECT TYPE SRSC_S_SELECT,
   LT_DATE    LIKE STKO-DATUV,
   I_QTY TYPE STPO-MENGE,
   L_TABIX    LIKE SY-TABIX.
DATA: L_COMP_QTY(18).
DATA: BEGIN OF I_MAST OCCURS 0,
   MATNR       LIKE   MAST-MATNR,    "MATERIAL
   WERKS       LIKE   MAST-WERKS,    "PLANT
   STLAN       LIKE   MAST-STLAN,    "BOM USAGE
   STLNR       LIKE   MAST-STLNR,    "BOM
   STLAL       LIKE   MAST-STLAL,    "ALTERNATIVE BOM
END OF I_MAST.
DATA: I_STPO LIKE STPO_API02  OCCURS 0 WITH HEADER LINE,
   YCPS_BOM_TAB LIKE YCPS_BOM_STR OCCURS 0 WITH HEADER LINE.
TABLES: MAST.
RANGES: R_MATNR FOR MAST-MATNR,
        R_WERKS FOR MAST-WERKS,
        R_STLAN FOR MAST-STLAN,
        R_STLAL FOR MAST-STLAL.

* Maximum number of lines for DB table
STATICS: S_S_IF TYPE SRSC_S_IF_SIMPLE,
        S_COUNTER_DATAPKID LIKE SY-TABIX.

* Initialization mode (first call by SAPI) or data transfer mode
* (following calls) ?
IF I_INITFLAG = SBIWA_C_FLAG_ON.
************************************************************************
* Initialization: check input parameters
*                buffer input parameters
*                prepare data selection
************************************************************************

* Check DataSource validity
CASE I_DSOURCE.
   WHEN 'YCPS_BOM_ITM'.
   WHEN OTHERS.
   IF 1 = 2. MESSAGE E009(R3). ENDF.
* this is a typical log call. Please write every error message like this
   LOG_WRITE 'E' "message type
      'R3' "message class
'009' "message number
I_DSOURCE  "message variable 1
' '     "message variable 2
RAISE ERROR_PASSED_TO_MESS_HANDLER.
ENDCASE.
APPEND LINES OF I_T_SELECT TO S_S_IF-T_SELECT.
* Fill parameter buffer for data extraction calls
  S_S_IF-REQUNR = I_REQUNR.
  S_S_IF-DSOURCE = I_DSOURCE.
  S_S_IF-MAXSIZE = I_MAXSIZE.
* Fill field list table for an optimized select statement
* (in case that there is no 1:1 relation between InfoSource fields
* and database table fields this may be far from being trivial)
  APPEND LINES OF I_T_FIELDS TO S_S_IF-T_FIELDS.
ELSE.  "Initialization mode or data extraction ?
************************************************************************
* Data transfer: First Call     OPEN CURSOR + FETCH
*                        Following Calls FETCH only
************************************************************************
* First data package -> OPEN CURSOR
  IF S_COUNTER_DATAPAKID = 0.
  CLEAR: R_MATNR,
          R_WERKS,
          R_STLAN,
          R_STLAL.
  REFRESH: R_MATNR,
          R_WERKS,
          R_STLAN,
          R_STLAL.
*-- Material Number
  LOOP AT S_S_IF-T_SELECT INTO L_S_SELECT WHERE FIELDNM = 'MATNR'.
    MOVE-CORRESPONDING L_S_SELECT TO R_MATNR.
  APPEND R_MATNR.
  ENDDO.
*-- Plant
  LOOP AT S_S_IF-T_SELECT INTO L_S_SELECT WHERE FIELDNM = 'WERKS'.
    MOVE-CORRESPONDING L_S_SELECT TO R_WERKS.
  APPEND R_WERKS.
  ENDDO.
*-- Item Category
  LOOP AT S_S_IF-T_SELECT INTO L_S_SELECT WHERE FIELDNM = 'STLAN'.
    MOVE-CORRESPONDING L_S_SELECT TO R_STLAN.
  APPEND R_STLAN.
  ENDDO.
*-- Alternative BOM
  LOOP AT S_S_IF-T_SELECT INTO L_S_SELECT WHERE FIELDNM = 'BOM_ALT'.
    MOVE-CORRESPONDING L_S_SELECT TO R_STLAL.
  APPEND R_STLAL.
  ENDDO.
SELECT * INTO CORRESPONDING FIELDS OF TABLE I_MAST
FROM MAST
WHERE WERKS IN R_WERKS
AND MATNR IN R_MATNR
AND STLAN IN R_STLAN
AND STLAL IN R_STLAL.
IF SY-SUBRC EQ 0.
  SORT I_MAST BY WERKS MATNR STLN.
  DELETE I_MAST WHERE WERKS = ".
ENDIF.
LOOP AT I_MAST.
  CALL FUNCTION 'CSAP_MAT_BOM_ITEM_SELECT'
  EXPORTING
    * I_STPO =
    MATERIAL = i_mast-matnr
    PLANT = i_mast-werks
    BOM_USAGE = i_mast-stlan
    ALTERNATIVE = i_mast-stlal
    * VALID_FROM = p_datuv
    * VALID_TO = p_datub
    FL_MATERIAL_CHECK = 'X'
    FL_FOREIGN_KEY_CHECK = 'X'
  IMPORTING
    * FL_WARNING =
    TABLES
      T_STPO = i_stpo
    EXCEPTIONS
      ERROR = 1
      OTHERS = 2
      .
      IF SY-SUBRC <> 0.
        * MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
        * WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
      ENDIF.
      LOOP AT I_STPO.
        L_COMP_QTY = I_STPO-COMP_QTY.
        REPLACE ALL OCCURRENCES OF ',' IN L_COMP_QTY WITH SPACE.
        CONDENSE L_COMP_QTY.
        I_QTY = L_COMP_QTY.
      MOVE-CORRESPONDING I_MAST TO YCPS_BOM_TAB.
      *Individual mapping is done as the valid to and from date is not in BW format
      YCPS_BOM_TAB-BOM_ALT = I_STPO-BOM_ALT.
      YCPS_BOM_TAB-ITEM_CATEG = I_STPO-ITEM_CATEG.
      YCPS_BOM_TAB-BOM_NO = I_STPO-BOM_NO.
      YCPS_BOM_TAB-ITEM_NO = I_STPO-ITEM_NO.
      YCPS_BOM_TAB-ITEM_NODE = I_STPO-ITEM_NODE.
      YCPS_BOM_TAB-ITEM_COUNT = I_STPO-ITEM_COUNT.
      YCPS_BOM_TAB-COMPONENT = I_STPO-COMPONENT.
      YCPS_BOM_TAB-COMP_QTY = I_QTY.
      YCPS_BOM_TAB-COMP_QTY+0(4) = I_STPO-COMP_QTY+6(4).
      YCPS_BOM_TAB-COMP_QTY+4(2) = I_STPO-COMP_QTY+0(2).
      YCPS_BOM_TAB-COMP_QTY+6(2) = I_STPO-COMP_QTY+3(2).
      YCPS_BOM_TAB-VALID_FROM+0(4) = I_STPO-VALID_FROM+6(4).
      YCPS_BOM_TAB-VALID_FROM+4(2) = I_STPO-VALID_FROM+0(2).
      YCPS_BOM_TAB-VALID_FROM+6(2) = I_STPO-VALID_FROM+3(2).
      YCPS_BOM_TAB-VALID_TO+0(4) = I_STPO-VALID_TO+6(4).
      YCPS_BOM_TAB-VALID_TO+4(2) = I_STPO-VALID_TO+0(2).
      YCPS_BOM_TAB-VALID_TO+6(2) = I_STPO-VALID_TO+3(2).
      YCPS_BOM_TAB-RECURSIVE = I_STPO-RECURSIVE.
      YCPS_BOM_TAB-DEP_LINK = I_STPO-DEP_LINK.
      YCPS_BOM_TAB-ALE_IND = I_STPO-ALE_IND.
      YCPS_BOM_TAB-CHG_NO_TO = I_STPO-CHG_NO_TO.
      YCPS_BOM_TAB-CREATED_ON = I_STPO-CREATED_ON.
      YCPS_BOM_TAB-CREATED_BY = I_STPO-CREATED_BY.
      YCPS_BOM_TAB-CHANGED_ON = I_STPO-CHANGED_ON.
      YCPS_BOM_TAB-CHANGED_BY = I_STPO-CHANGED_BY.
      YCPS_BOM_TAB-FLDELETE = I_STPO-FLDELETE.
      APPEND YCPS_BOM_TAB.
      ENDLOOP.
REFRESH I_STPO.
ENDLOOP.
ENDIF.  "First data package ?
  IF I_STPO IS INITIAL.
    RAISE NO_MORE_DATA.
  ENDIF.
  APPEND LINES OF YCPS_BOM_TAB TO E_T_DATA.
  S_COUNTER_DATAPAKID = S_COUNTER_DATAPAKID + 1.
  REFRESH YCPS_BOM_TAB.
ENDIF.
ENDFUNCTION.
Related Content

http://help.sap.com/saphelp_nw04/helpdata/en/43/40b8aeeaa58ba4d9a81c7332119a4b4/content.htm

For more information, visit the Business Intelligence homepage
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.