Stock Requirements List (MD04) Report in SAP BW/BI

Applies to:

SAP BW 3.x and higher versions as BW system. For more information, visit the Business Intelligence homepage

Summary

This document is intended to show how to create Stock Requirements Report in SAP BW system similar to SAP R/3 transaction MD04. As the report will be based on virtual Infocube, it will show the current stock at runtime.

Author: Amol Ishwarlal Jaiswal

Company: Infosys Technologies Ltd.

Created on: 21 April 2010

Author Bio

Amol Jaiswal is a BI consultant with over 2.5 years of industry experience in implementation and maintenance of SAP BW/BI systems. Presently, Amol is working with Infosys Technologies Ltd and is involved in execution of SAP BW/BI projects.

Table of Contents

Introduction to Stock Requirements List....3
MRP List:....3
Stock/Requirements List:....3
Datasource Creation in SAP R/3....4
BW Data Modeling and Reporting....17
Related Content....20
Disclaimer and Liability Notice....21

Introduction to Stock Requirements List

The main function of material requirements planning is to guarantee material availability, that is, it is used to procure or produce the requirement quantities on time both for internal purposes and for sales and distribution. This process involves the monitoring of stocks and, in particular, the automatic creation of procurement proposals for purchasing and production.

During the planning run, the system analyzes the requirements that exist for the planned materials and creates procurement elements that cover these requirements.

The following two main evaluations are available for analyzing the planning result:

1. MRP list
2. Stock/Requirements list

These evaluations display all receipt and issue elements for a material in the form of a table and enable you to gain a quick overview of the stock/requirements situation for the material as well as to branch into the editing function for the MRP elements for this material.

MRP List:

The system creates MRP lists during the planning run according to how you set the creation indicator. These lists contain the planning result for the material. The MRP list always displays the stock/requirements situation **at the time of the last planning run** and it also provides a work basis for the MRP controller. Changes that are made after the planning date are not taken into consideration, so the list is **static**.

MRP lists are stored in the system until they are either deleted manually or replaced by new lists from a subsequent planning run. The transaction code for MRP List is MD05.

Stock/Requirements List:

In the stock/requirements list, the **most up-to-date stock and requirements situation** is displayed.

The main difference between the MRP list and the stock/requirements list is that each time the stock/requirements list is called up; the system selects the various MRP elements and displays the most up-to-date situation. You thus always see the current availability situation of the material in the stock/requirements list. Changes that are made after the planning date are displayed directly, so the list is therefore **dynamic**. The transaction code for this is MD04.

The advantage of taking this data in BW report, you can see this data at different granularity levels and with different filters and selection criteria. Currently with MD04 transaction, you can either see the report for one material and plant combination or combination of plant and one of the following: MRP controller, Product group, vendor etc.

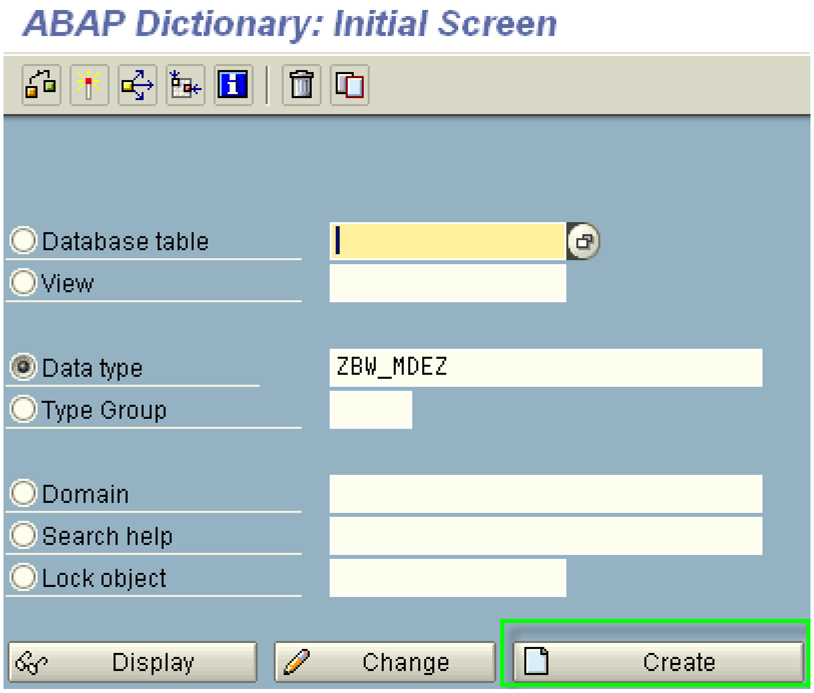
Stock/requirements lists are not saved in a fixed state in the system, but are subject to change and only exist in the working memory. In order to capture all these dynamic values in BW report, we can create a datasource and virtual Infocube, which will pull data from R/3 to BW at query (BW report) runtime and users can report as if they are using transactional system. This may cause some performance hit to report compared to the reports based on basic Infocubes, but this can be improved if we use optimized method of getting this data to BW. In this case, we will be creating datasource based on a function module to have good performance of BW report.

Datasource Creation in SAP R/3

This development would require creation of generic datasource using function module. Let's create this datasource in SAP R/3 system.

For creation of this datasource, we need to create structure and customized function module first. The customized function module which we will be creating will be using standard function module MD_STOCK_REQUIREMENTS_LIST_API. This function module takes Material and Plant as input and provides MD04 output in the table MDEZX.

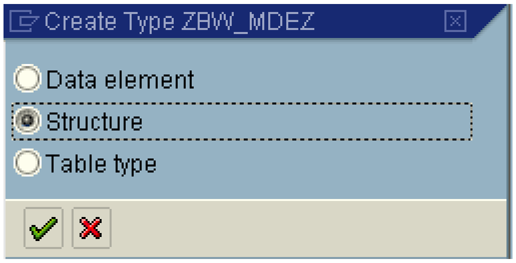
Let's create the structure ZBW_MDEZ in transaction code SE11 as shown in *fig-1* and *fig-2*. This would be similar to standard structure MDEZ.



ABAP Dictionary: Initial Screen

○ Database table
○ View
● Data type ZBW_MDEZ
○ Type Group
○ Domain
○ Search help
○ Lock object

Fig-1: Data Type ZBW_MDEZ Creation



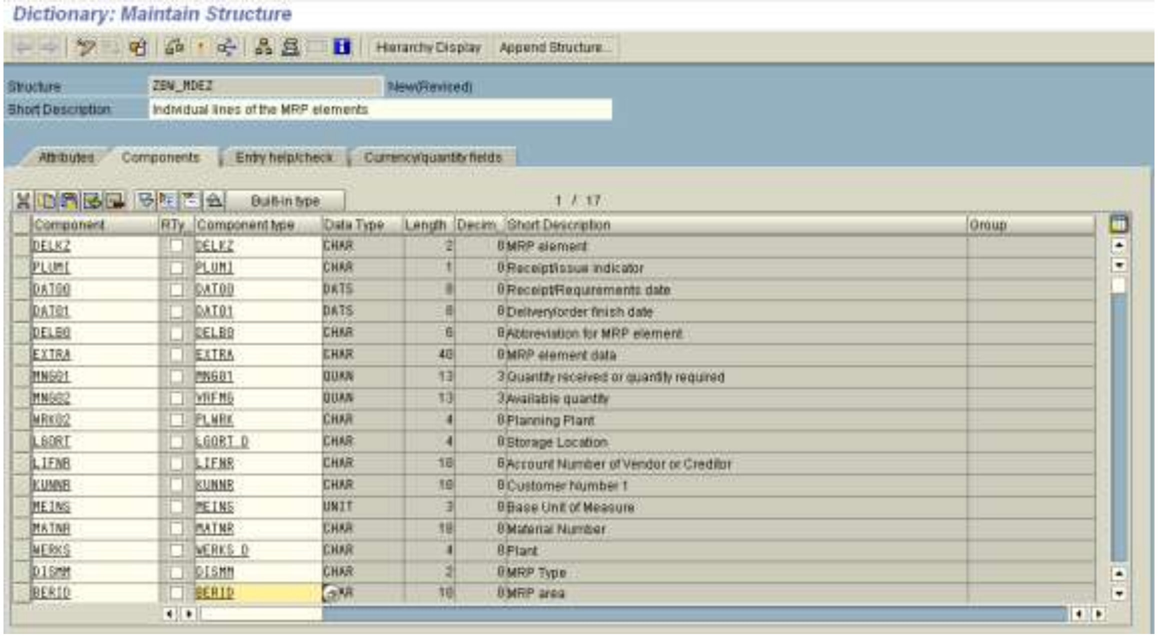
Create Type ZBW_MDEZ

○ Data element
● Structure
○ Table type

Fig-2: Structure Data type

Then put the fields as shown in the *fig-3* below. The field list is mentioned in the table below. Please note, I have chosen only those fields from MDEZ which are required for this BW report. There are many other fields which can be used as per your requirement.

Component	Component Type
DELKZ	DELKZ
PLUMI	PLUMI
DAT00	DAT00
DAT01	DAT01
DELB0	DELB0
EXTRA	EXTRA
MNG01	MNG01
MNG02	VRFMG
WRK02	PLWRK
LGORT	LGORT_D
LIFNR	LIFNR
KUNNR	KUNNR
MEINS	MEINS
MATNR	MATNR
WERKS	WERKS_D
DISMM	DISMM
BERID	BERID
MEINH	MEINH



Dictionary: Maintain Structure

Structure: ZBW_MDEZ New(Revised)

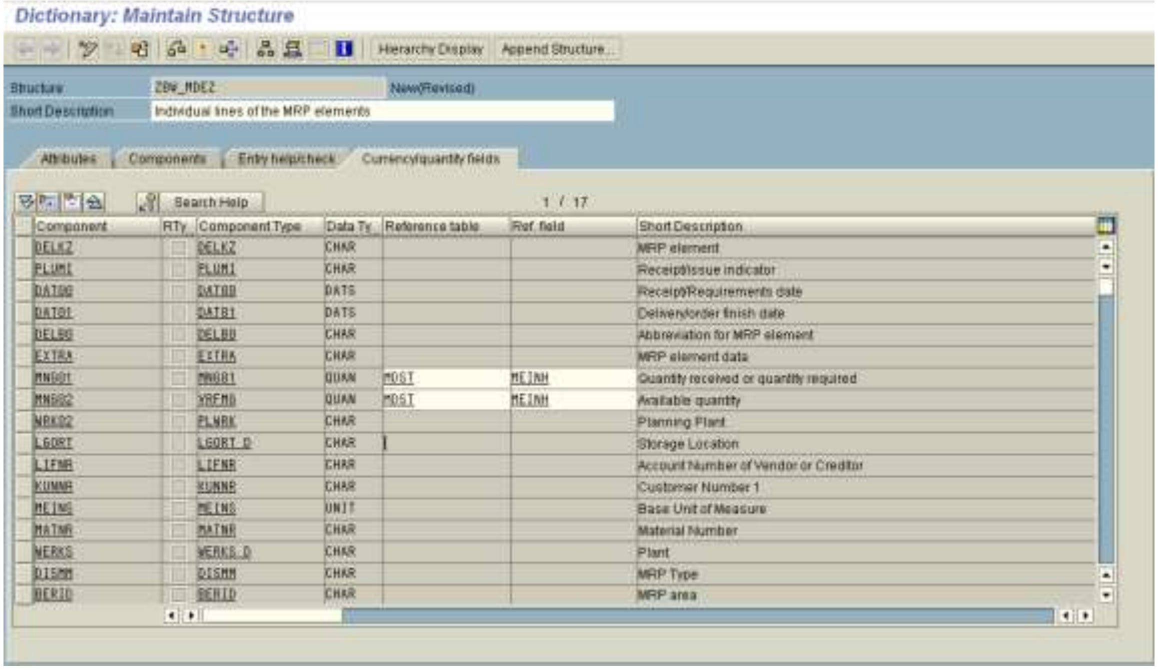
Short Description: Individual lines of the MRP elements

1 / 17

Component	RTy.	Component type	Data Type	Length	Decim.	Short Description	Group
DELKZ	□	DELKZ	CHAR	2	0	MRP element	
PLUMI	□	PLUMI	CHAR	1	0	Receipt/issue indicator	
DAT00	□	DAT00	DATS	8	0	Receipt/Requirements date	
DAT01	□	DAT01	DATS	8	0	Delivery/order finish date	
DELB0	□	DELB0	CHAR	6	0	Abbreviation for MRP element	
EXTRA	□	EXTRA	CHAR	40	0	MRP element data	
MNG01	□	MNG01	QUAN	13	3	Quantity received or quantity required	
MNG02	□	VRFMG	QUAN	13	3	Available quantity	
WRK02	□	PLWRK	CHAR	4	0	Planning Plant	
LGORT	□	LGORT_D	CHAR	4	0	Storage Location	
LIFNR	□	LIFNR	CHAR	10	0	Account Number of Vendor or Creditor	
KUNNR	□	KUNNR	CHAR	10	0	Customer Number 1	
MEINS	□	MEINS	UNIT	3	0	Base Unit of Measure	
MATNR	□	MATNR	CHAR	18	0	Material Number	
WERKS	□	WERKS_D	CHAR	4	0	Plant	
DISMM	□	DISMM	CHAR	2	0	MRP Type	
BERID	□	BERID	CHAR	10	0	MRP area	

Fig-3: Structure ZBW_MDEZ fields

You need to give reference table and field for Currency/Quantity fields as shown in *fig-4*.



Dictionary: Maintain Structure

Structure: ZBW_MDEZ New(Revised)
Short Description: Individual lines of the MRP elements

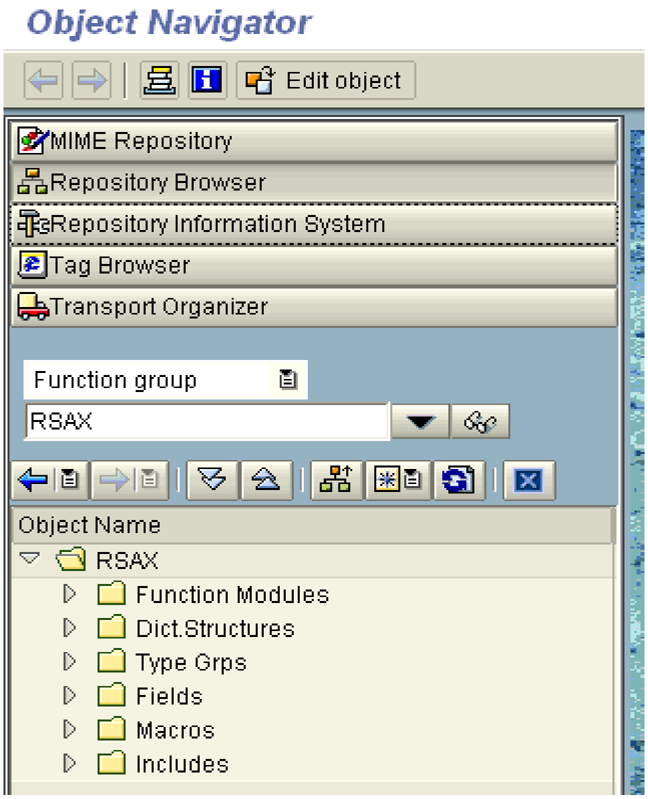
Component	RTy	Component Type	Data Ty	Reference table	Ref. field	Short Description
DELKZ	□	DELKZ	CHAR			MRP element
PLUMI	□	PLUMI	CHAR			Receipt/issue indicator
DAT00	□	DAT00	DATS			Receipt/Requirements date
DAT01	□	DAT01	DATS			Delivery/order finish date
DELB0	□	DELB0	CHAR			Abbreviation for MRP element
EXTRA	□	EXTRA	CHAR			MRP element data
MNG01	□	MNG01	QUAN	MDST	MEINH	Quantity received or quantity required
MNG02	□	VRFMG	QUAN	MDST	MEINH	Available quantity
WRK02	□	PLWRK	CHAR			Planning Plant
LGORT	□	LGORT_D	CHAR			Storage Location
LIFNR	□	LIFNR	CHAR			Account Number of Vendor or Creditor
KUNNR	□	KUNNR	CHAR			Customer Number 1
MEINS	□	MEINS	UNIT			Base Unit of Measure
MATNR	□	MATNR	CHAR			Material Number
WERKS	□	WERKS_D	CHAR			Plant
DISMM	□	DISMM	CHAR			MRP Type
BERID	□	BERID	CHAR			MRP area

Fig-4: Reference tables in structure ZBW_MDEZ

Now activate the structure. It will ask for Object Directory Entry, you can create it as Local Object ($TMP). This will create our structure.

Next step is creation of function module.

Go to transaction code SE80 and select ‘Function Group’, put ‘RSAX’ as shown in *fig-5* and press enter.



Object Navigator

Function group
RSAX

Object Name
- RSAX
  - Function Modules
  - Dict.Structures
  - Type Grps
  - Fields
  - Macros
  - Includes

Fig-5: Function Group RSAX

Now select RSAX folder. Right click and press ‘Copy’ as shown in *fig-6*. Then give name of new function group as ZBW_RSAX in the prompted window as shown in *fig-7*.

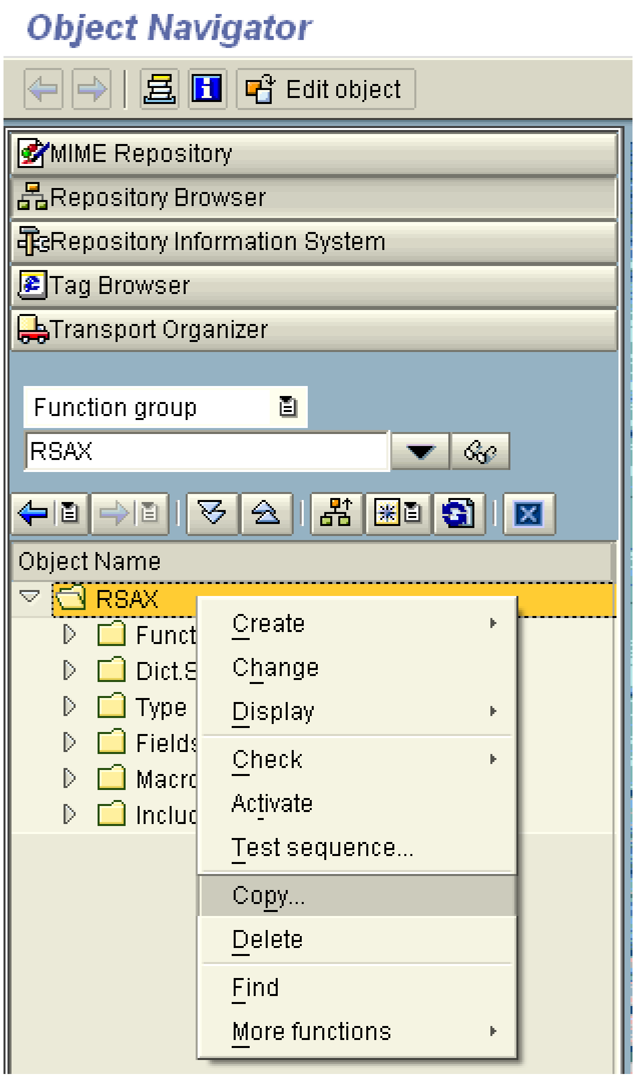
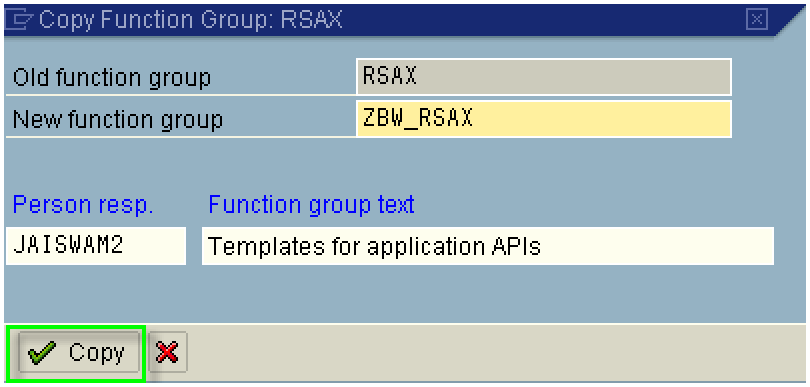


Fig-6: Copying Function Group RSAX

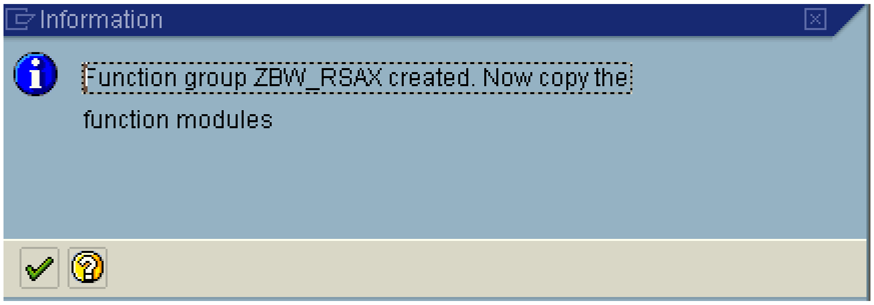

Copy Function Group: RSAX

Old function group	RSAX
New function group	ZBW_RSAX

Person resp.	Function group text
JAISWAM2	Templates for application APIs

fig-7: Function Group ZBW_RSAX

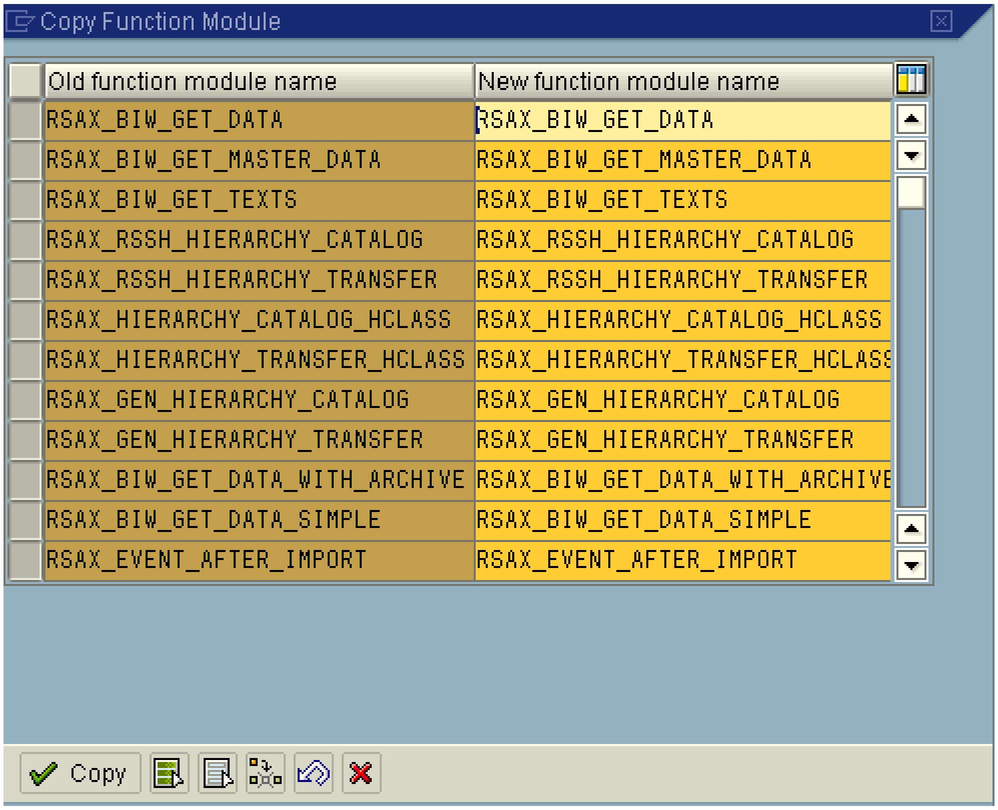
It will ask for Object Directory Entry, once you click on ‘Copy’. You can create it as Local Object ($TMP). After that another prompt will come as shown in *fig-8*, then click on continue button, which will take you to screen shown in *fig-9*.



Information

Function group ZBW_RSAX created. Now copy the function modules

Fig-8

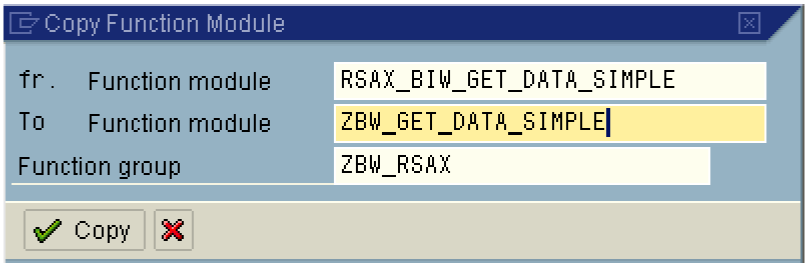


Copy Function Module

Old function module name	New function module name
RSAX_BIW_GET_DATA	RSAX_BIW_GET_DATA
RSAX_BIW_GET_MASTER_DATA	RSAX_BIW_GET_MASTER_DATA
RSAX_BIW_GET_TEXTS	RSAX_BIW_GET_TEXTS
RSAX_RSSH_HIERARCHY_CATALOG	RSAX_RSSH_HIERARCHY_CATALOG
RSAX_RSSH_HIERARCHY_TRANSFER	RSAX_RSSH_HIERARCHY_TRANSFER
RSAX_HIERARCHY_CATALOG_HCLASS	RSAX_HIERARCHY_CATALOG_HCLASS
RSAX_HIERARCHY_TRANSFER_HCLASS	RSAX_HIERARCHY_TRANSFER_HCLASS
RSAX_GEN_HIERARCHY_CATALOG	RSAX_GEN_HIERARCHY_CATALOG
RSAX_GEN_HIERARCHY_TRANSFER	RSAX_GEN_HIERARCHY_TRANSFER
RSAX_BIW_GET_DATA_WITH_ARCHIVE	RSAX_BIW_GET_DATA_WITH_ARCHIVE
RSAX_BIW_GET_DATA_SIMPLE	RSAX_BIW_GET_DATA_SIMPLE
RSAX_EVENT_AFTER_IMPORT	RSAX_EVENT_AFTER_IMPORT

Fig-9: Copy Function Module

Select only one needed Function module Ex: RSAX_BIW_GET_DATA_SIMPLE and press Enter. Give the name you want to have for your function module and click on ‘Copy’ as shown in *fig-10*. This will create our function group and function module.



Copy Function Module

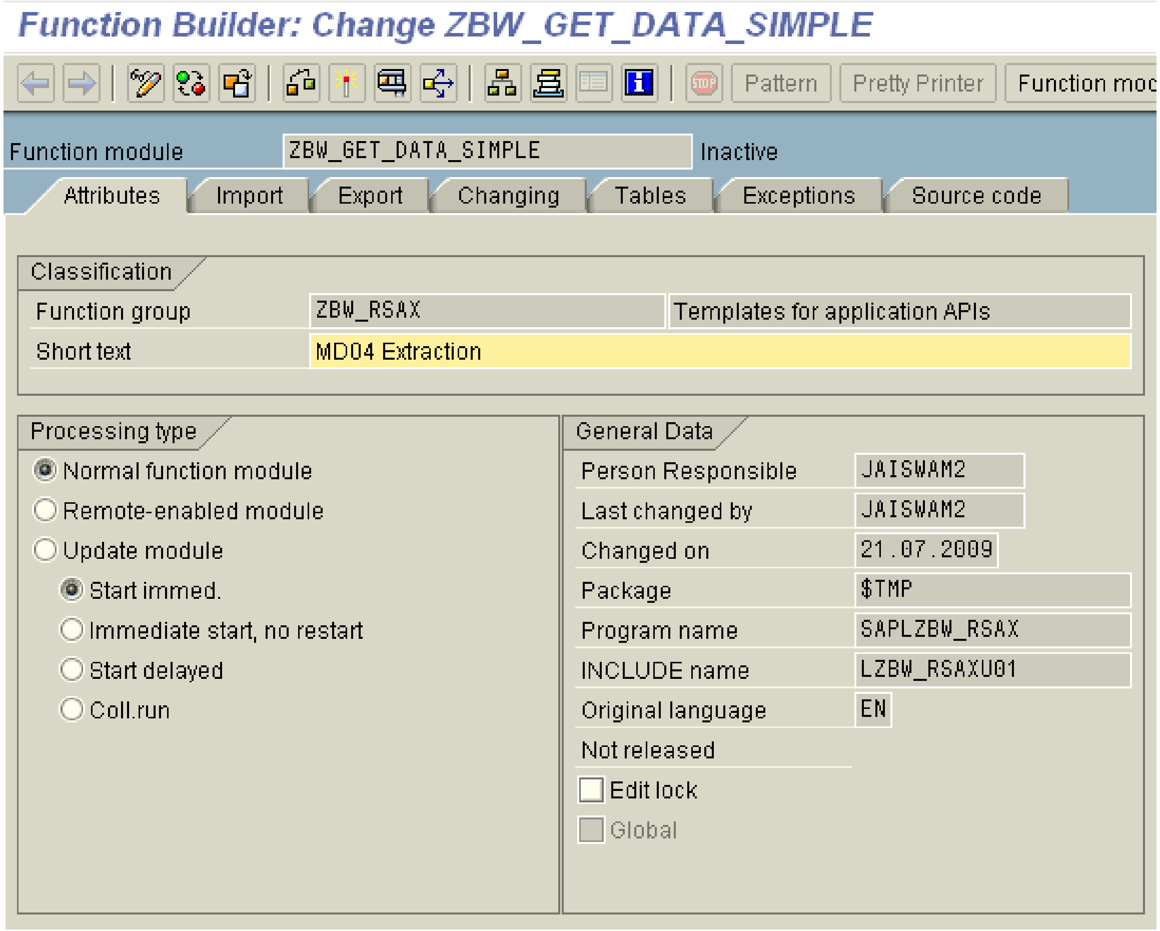
fr. Function module RSAX_BIW_GET_DATA_SIMPLE

To Function module ZBW_GET_DATA_SIMPLE

Function group ZBW_RSAX

Fig-10: Function Module ZBW_GET_DATA_SIMPLE

Now Go to T-Code SE37 and give Function module name ZBW_GET_DATA_SIMPLE and press ‘Change’ button. You can change the text for your function module in ‘Attibutes’ tab as shown in *fig-11*.



Function Builder: Change ZBW_GET_DATA_SIMPLE

Function module ZBW_GET_DATA_SIMPLE Inactive

Classification

Function group	ZBW_RSAX	Templates for application APIs
Short text	MD04 Extraction	

Processing type

- ● Normal function module
- ○ Remote-enabled module
- ○ Update module
  - ● Start immed.
  - ○ Immediate start, no restart
  - ○ Start delayed
  - ○ Coll.run

General Data

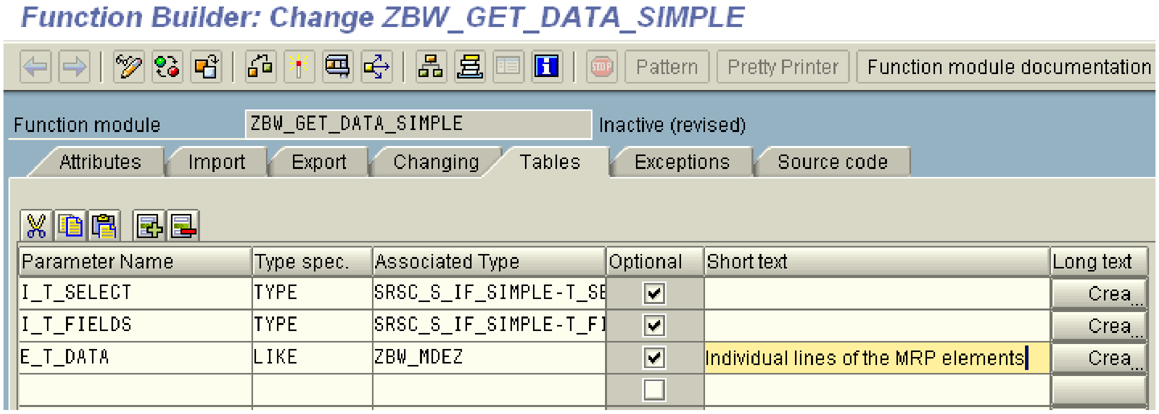
Person Responsible	JAISWAM2
Last changed by	JAISWAM2
Changed on	21.07.2009
Package	$TMP
Program name	SAPLZBW_RSAX
INCLUDE name	LZBW_RSAXU01
Original language	EN
Not released	

□ Edit lock

□ Global

Fig-11: Attributes of Function module ZBW_GET_DATA_SIMPLE

Don't change Import, Export, and Changing Tabs. Now go to 'Tables' tab and give Type spec 'LIKE' and Associated Type 'ZBW_MDEZ' as shown in *fig-12* and press enter.



Function Builder: Change ZBW_GET_DATA_SIMPLE

Function module ZBW_GET_DATA_SIMPLE Inactive (revised)

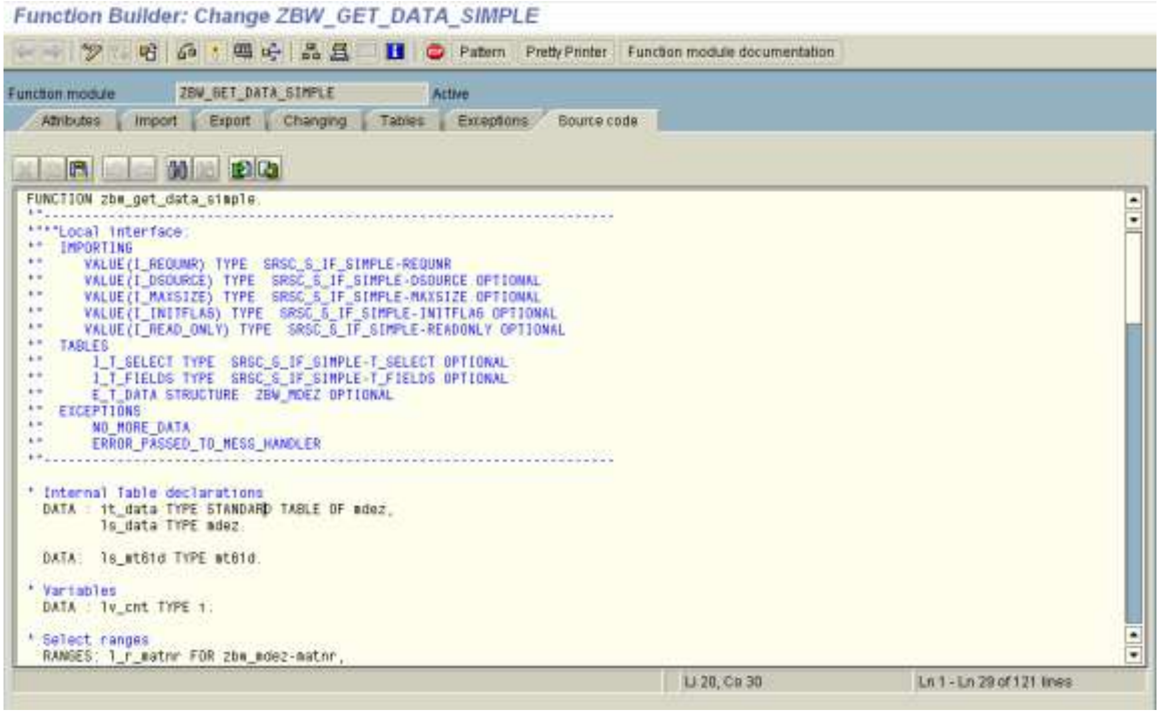
Parameter Name	Type spec.	Associated Type	Optional	Short text	Long text
I_T_SELECT	TYPE	SRSC_S_IF_SIMPLE-T_SE	☑		Crea...
I_T_FIELDS	TYPE	SRSC_S_IF_SIMPLE-T_FI	☑		Crea...
E_T_DATA	LIKE	ZBW_MDEZ	☑	Individual lines of the MRP elements	Crea...
			□		

fig-12: Function Module ZBW_GET_DATA_SIMPLE 'Tables' Parameters

Now go to Source Code to edit as shown in *fig-13*. Put the below mentioned code and activate the function module. Now go to t-code SE80 and select function group ZBW_RSAX and activate function group. **Please note activation of function group is an important step.**

```
FUNCTION zbw_get_data_simple.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  IMPORTING
*"     VALUE(I_REQUNR) TYPE  SRSC_S_IF_SIMPLE-REQUNR
*"     VALUE(I_DSOURCE) TYPE  SRSC_S_IF_SIMPLE-DSOURCE OPTIONAL
*"     VALUE(I_MAXSIZE) TYPE  SRSC_S_IF_SIMPLE-MAXSIZE OPTIONAL
*"     VALUE(I_INITFLAG) TYPE  SRSC_S_IF_SIMPLE-INITFLAG OPTIONAL
*"     VALUE(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  ZBW_MDEZ OPTIONAL
*"  EXCEPTIONS
*"      NO_MORE_DATA
*"      ERROR_PASSED_TO_MESS_HANDLER
*"----------------------------------------------------------------------
* Internal Table declarations
  DATA : it_data TYPE STANDARD TABLE OF mdez,
         ls_data TYPE mdez.
  DATA:  ls_mt61d TYPE mt61d.
* Variables
  DATA : lv_cnt TYPE i.
* Select ranges
  RANGES: l_r_matnr FOR zbw_mdez-matnr,
          l_r_werks FOR zbw_mdez-werks.
  STATICS: BEGIN OF ss_mat,
             matnr TYPE   marc-matnr,
             werks TYPE   marc-werks,
           END OF ss_mat,
           st_mat LIKE TABLE OF ss_mat.
* Auxiliary Selection criteria structure
  DATA: l_s_select TYPE srsc_s_select.
* Maximum number of lines for DB table
  STATICS: s_s_if TYPE srsc_s_if_simple.
* 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
************************************************************************
* Fill parameter buffer for data extraction calls
    s_s_if-maxsize   = i_maxsize.
    LOOP AT i_t_select INTO l_s_select WHERE fieldnm = 'MATNR'.
      MOVE-CORRESPONDING l_s_select TO l_r_matnr.
      APPEND l_r_matnr.
    ENDLOOP.
    LOOP AT i_t_select INTO l_s_select WHERE fieldnm = 'WERKS'.
      MOVE-CORRESPONDING l_s_select TO l_r_werks.
      APPEND l_r_werks.
    ENDLOOP.
    SELECT matnr werks FROM marc INTO TABLE st_mat
      WHERE matnr IN l_r_matnr
```

```
        AND werks IN l_r_werks.
  ELSE.                 "Initialization mode or data extraction ?
    CLEAR e_t_data.
    REFRESH e_t_data.
    LOOP AT st_mat INTO ss_mat.
      CLEAR ls_mt61d.
      CALL FUNCTION 'MD_STOCK_REQUIREMENTS_LIST_API'
        EXPORTING
          matnr                    = ss_mat-matnr
          werks                    = ss_mat-werks
        IMPORTING
          e_mt61d                  = ls_mt61d
        TABLES
          mdezx                    = it_data
        EXCEPTIONS
          material_plant_not_found = 1
          plant_not_found          = 2
          error_message            = 3
          OTHERS                   = 4.
      IF sy-subrc NE 0.
        CONTINUE.
      ENDIF.
      LOOP AT it_data INTO ls_data.
        MOVE-CORRESPONDING ls_data TO e_t_data .
        e_t_data-meins = ls_mt61d-meins.
        e_t_data-matnr = ss_mat-matnr.
        e_t_data-werks = ss_mat-werks.
        e_t_data-dismm = ls_mt61d-dismm.
        e_t_data-berid = ls_mt61d-berid.
        APPEND e_t_data.
      ENDLOOP.
      DELETE TABLE st_mat FROM ss_mat.
      ADD 1 TO lv_cnt.
      IF lv_cnt > s_s_if-maxsize.
        EXIT.
      ENDIF.
    ENDLOOP.
    IF sy-subrc NE 0.
      RAISE no_more_data.
    ENDIF.
  ENDIF.              "Initialization mode or data extraction ?
ENDFUNCTION.
```

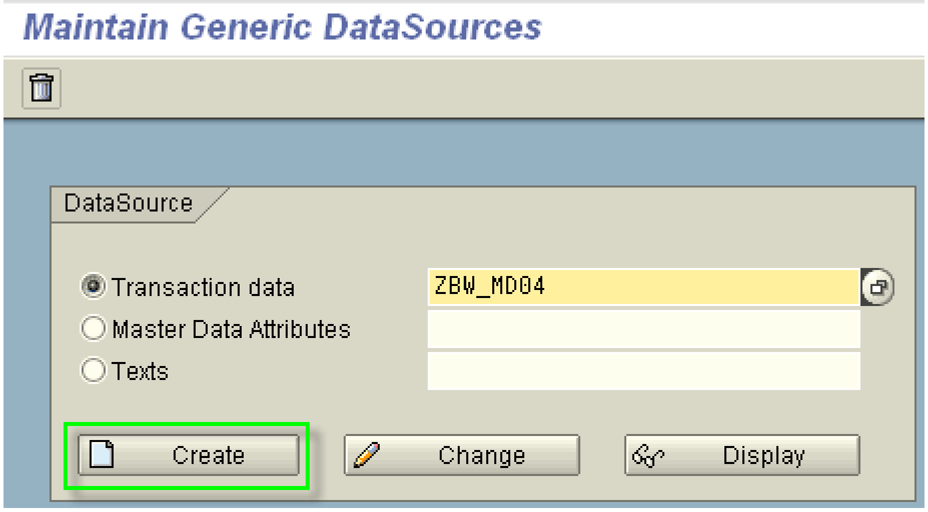


```
FUNCTION zbw_get_data_simple.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  IMPORTING
*"     VALUE(I_REQUNR) TYPE  SRSC_S_IF_SIMPLE-REQUNR
*"     VALUE(I_DSOURCE) TYPE  SRSC_S_IF_SIMPLE-DSOURCE OPTIONAL
*"     VALUE(I_MAXSIZE) TYPE  SRSC_S_IF_SIMPLE-MAXSIZE OPTIONAL
*"     VALUE(I_INITFLAG) TYPE  SRSC_S_IF_SIMPLE-INITFLAG OPTIONAL
*"     VALUE(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  ZBW_MDEZ OPTIONAL
*"  EXCEPTIONS
*"      NO_MORE_DATA
*"      ERROR_PASSED_TO_MESS_HANDLER
*"----------------------------------------------------------------------

* Internal Table declarations
  DATA : it_data TYPE STANDARD TABLE OF mdez,
         ls_data TYPE mdez.

  DATA:  ls_mt61d TYPE mt61d.

* Variables
  DATA : lv_cnt TYPE i.

* Select ranges
  RANGES: l_r_matnr FOR zbw_mdez-matnr,
```

Fig-13: Source code of Function Module ZBW_GET_DATA_SIMPLE

This finishes creation of function module required for our datasource. Now next step is to create the datasource.

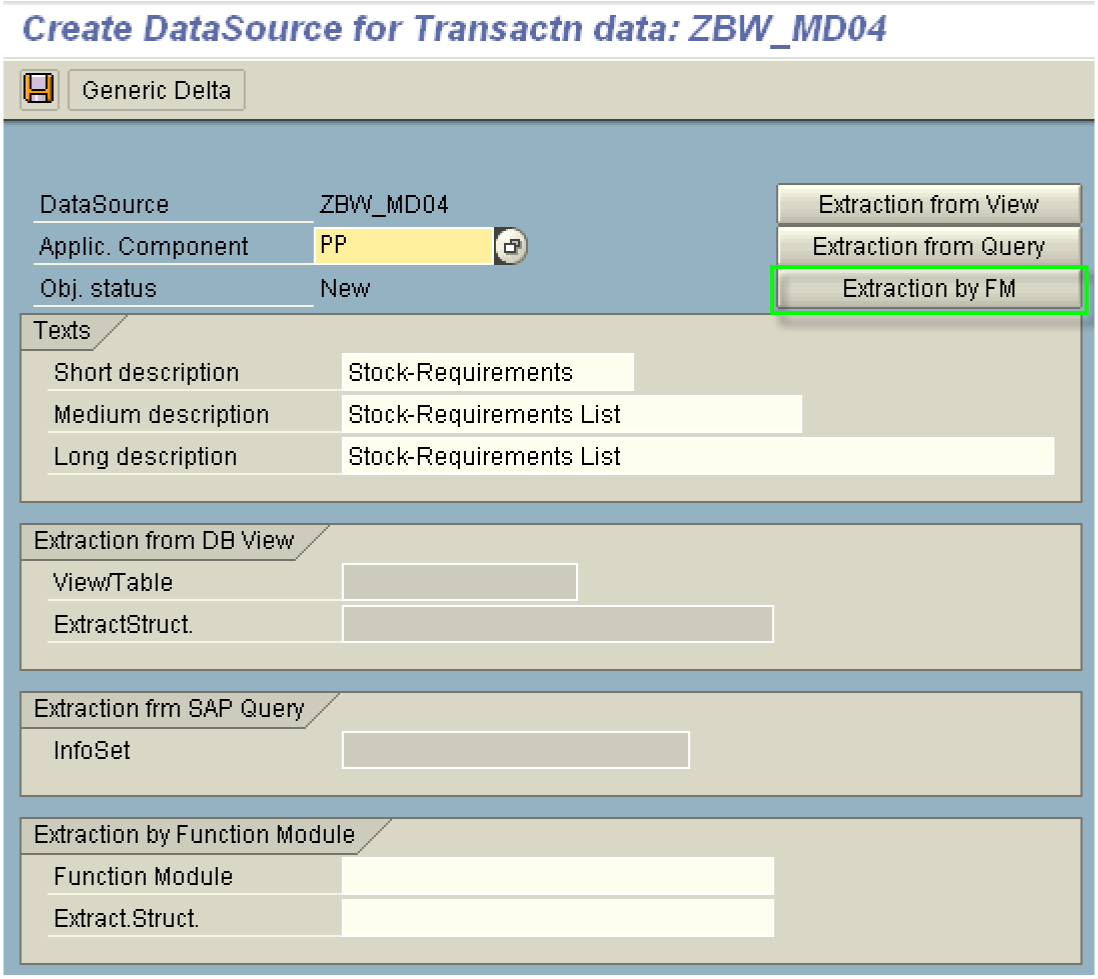
Go to transaction code RSO2. Put datasource name as ZBW_MD04 and click on 'Create' as shown in *fig-14*.



● Transaction data: ZBW_MD04
○ Master Data Attributes
○ Texts

Fig-14: Creation of Transaction datasource ZBW_MD04

Put Application component as 'PP' and provide descriptions as shown in *fig-15* and click on button 'Extraction by FM'.



Create DataSource for Transactn data: ZBW_MD04

DataSource	ZBW_MD04
Applic. Component	PP
Obj. status	New

Texts

Short description	Stock-Requirements
Medium description	Stock-Requirements List
Long description	Stock-Requirements List

Extraction from DB View

View/Table	
ExtractStruct.	

Extraction frm SAP Query

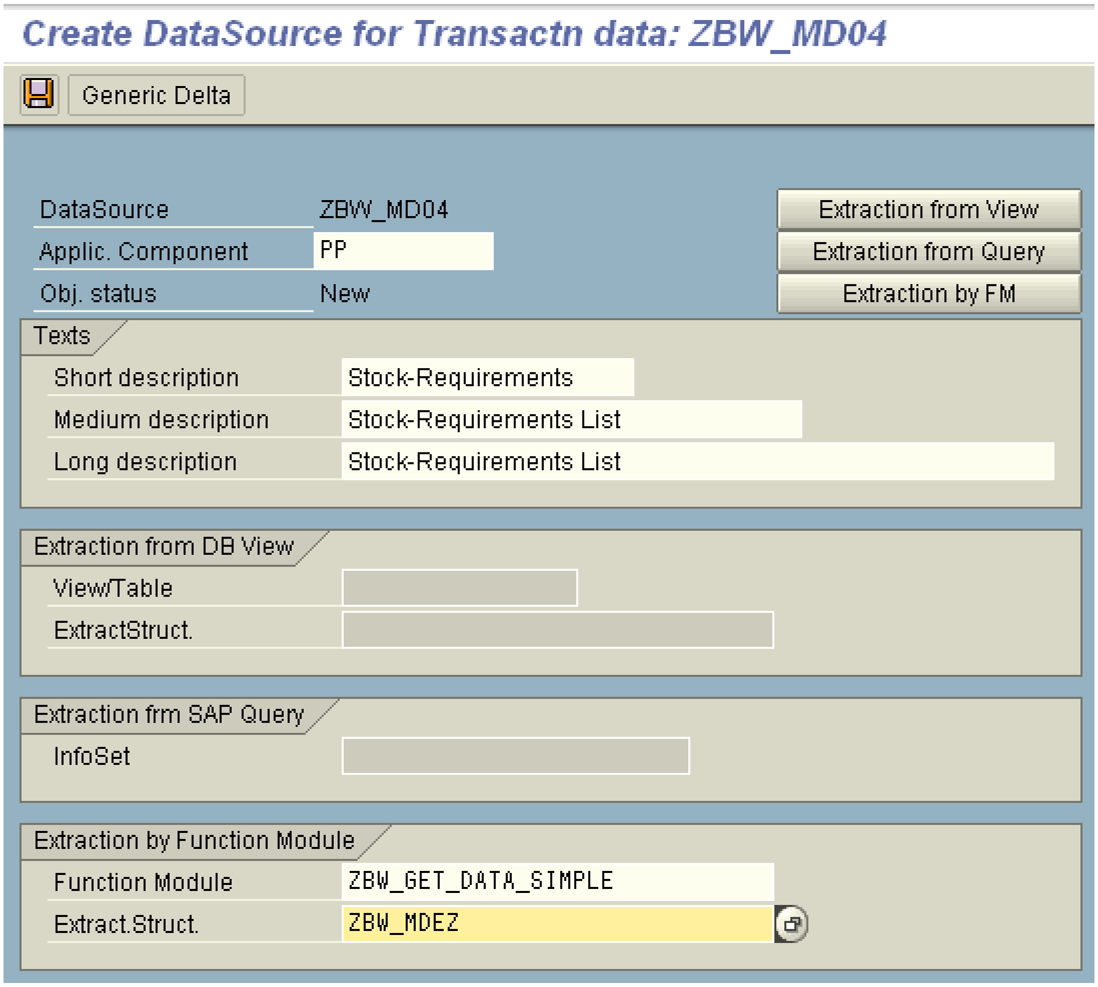
InfoSet	

Extraction by Function Module

Function Module	
Extract.Struct.	

Fig-15: Datasource ZBW_MD04- Extraction by FM

Now put Function module and Extract Structure name as shown in *fig-16* and save it.



Create DataSource for Transactn data: ZBW_MD04

DataSource	ZBW_MD04
Applic. Component	PP
Obj. status	New

Texts

Short description	Stock-Requirements
Medium description	Stock-Requirements List
Long description	Stock-Requirements List

Extraction from DB View

View/Table	
ExtractStruct.	

Extraction frm SAP Query

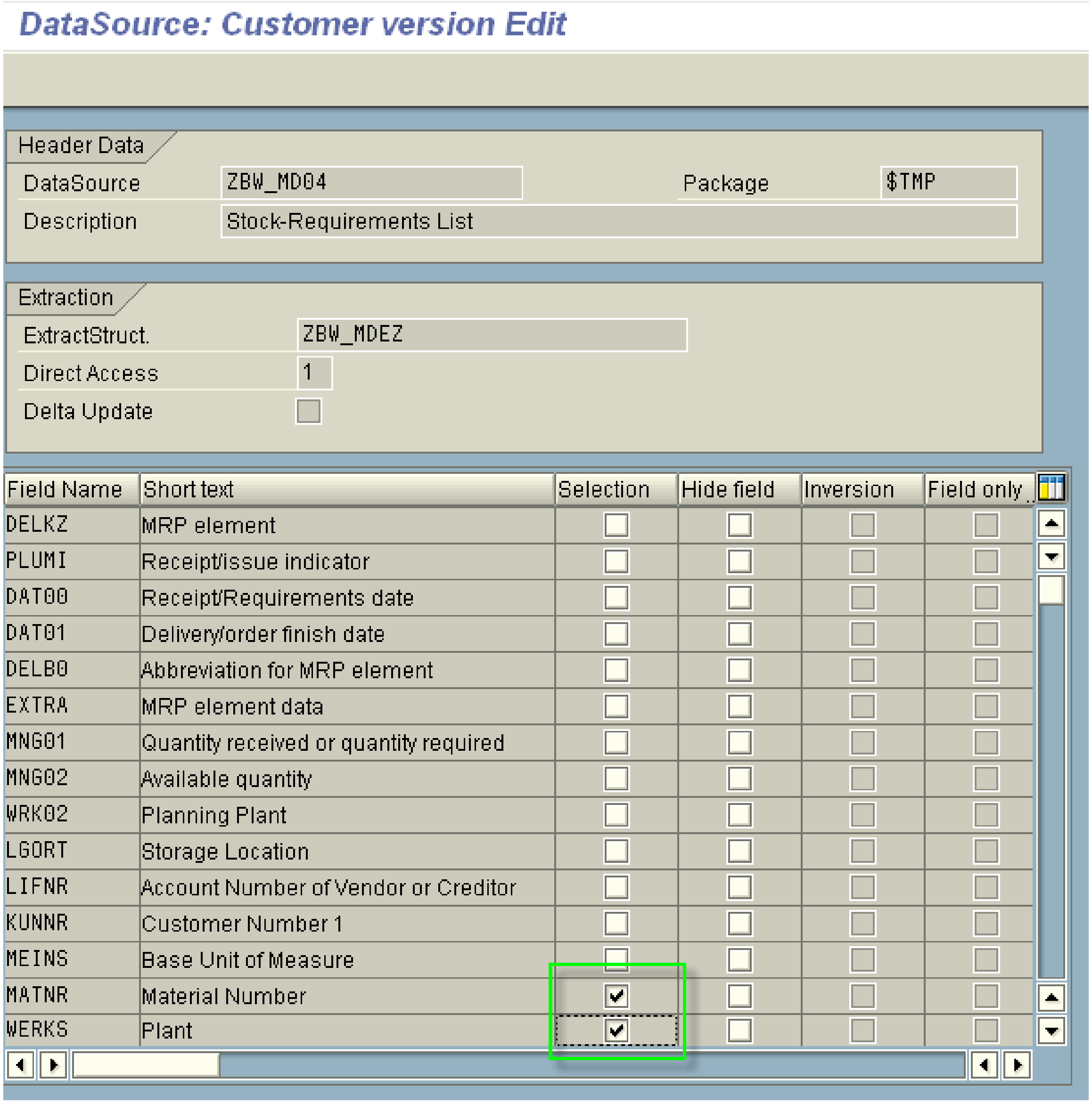
InfoSet	

Extraction by Function Module

Function Module	ZBW_GET_DATA_SIMPLE
Extract.Struct.	ZBW_MDEZ

Fig-16: Datasource ZBW_MD04- Extraction by FM

It will take you to the screen as shown in *fig-17*. Then mark checkboxes of Selection for fields MATNR and WERKS and save it. This concludes creation of datasource.



DataSource: Customer version Edit

Header Data

DataSource	ZBW_MD04	Package	$TMP
Description	Stock-Requirements List		

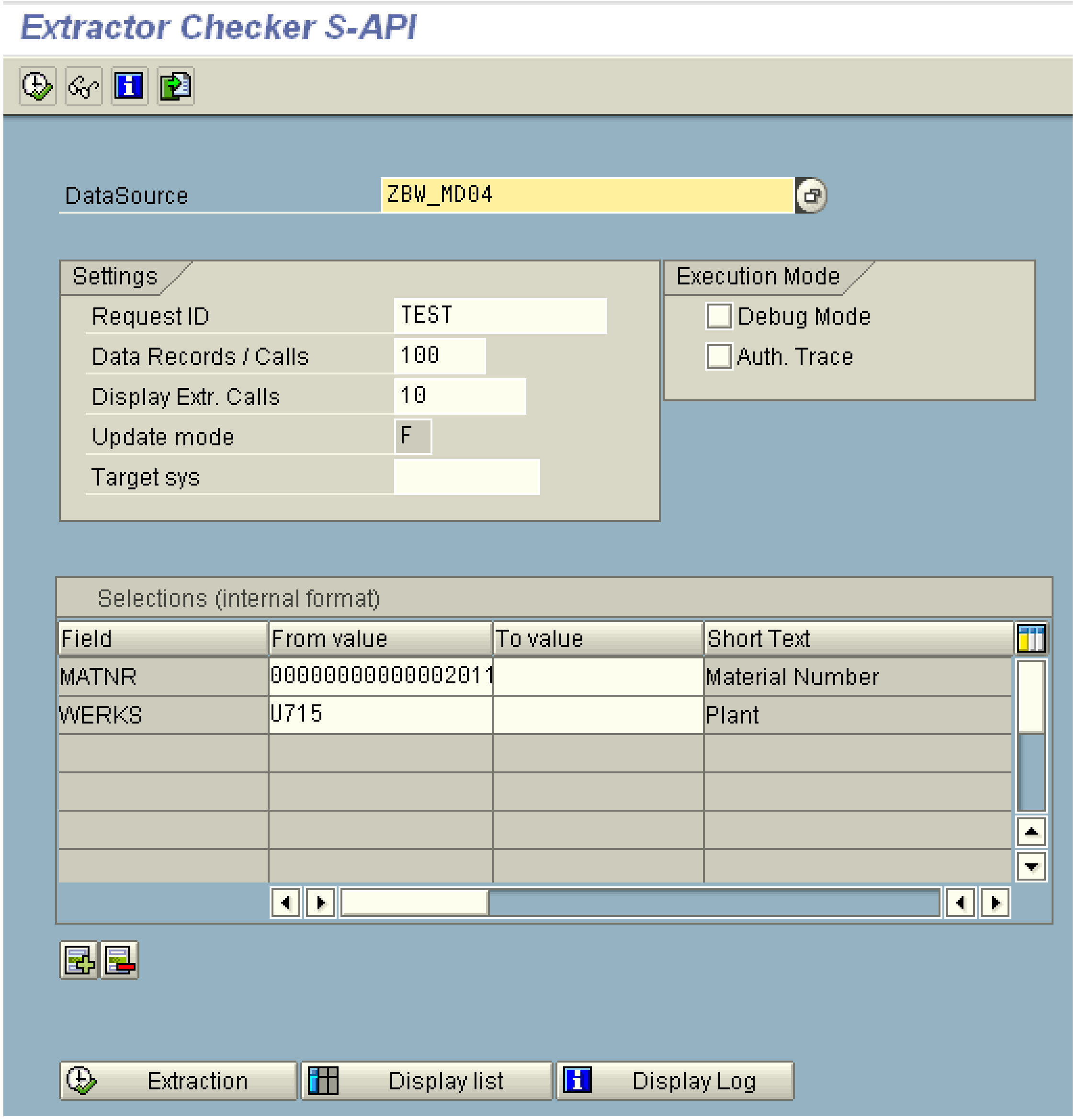
Extraction

ExtractStruct.	ZBW_MDEZ
Direct Access	1
Delta Update	□

Field Name	Short text	Selection	Hide field	Inversion	Field only
DELKZ	MRP element	□	□	□	□
PLUMI	Receipt/issue indicator	□	□	□	□
DAT00	Receipt/Requirements date	□	□	□	□
DAT01	Delivery/order finish date	□	□	□	□
DELB0	Abbreviation for MRP element	□	□	□	□
EXTRA	MRP element data	□	□	□	□
MNG01	Quantity received or quantity required	□	□	□	□
MNG02	Available quantity	□	□	□	□
WRK02	Planning Plant	□	□	□	□
LGORT	Storage Location	□	□	□	□
LIFNR	Account Number of Vendor or Creditor	□	□	□	□
KUNNR	Customer Number 1	□	□	□	□
MEINS	Base Unit of Measure	□	□	□	□
MATNR	Material Number	☑	□	□	□
WERKS	Plant	☑	□	□	□

Fig-17: Selection Fields for datasource ZBW_MD04

You can check the extraction of datasource ZBW_MD04 in transaction code RSA3 for any material/plant combination as shown in *fig-18* and *fig-19* and can compare the result with transaction code MD04.



Extractor Checker S-API

DataSource: ZBW_MD04

Settings

Request ID	TEST
Data Records / Calls	100
Display Extr. Calls	10
Update mode	F
Target sys	

Execution Mode

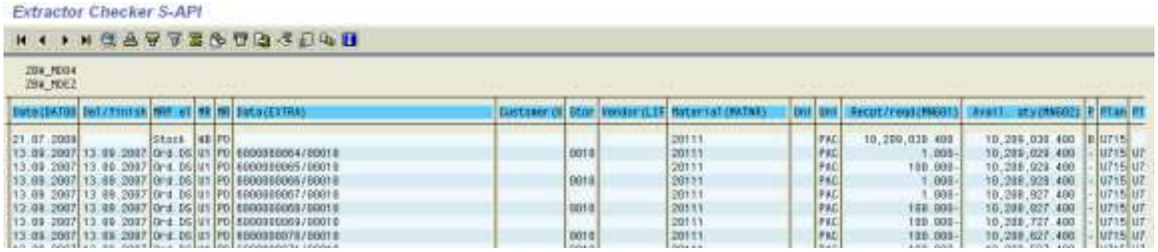
□ Debug Mode

□ Auth. Trace

Selections (internal format)

Field	From value	To value	Short Text
MATNR	000000000000002011		Material Number
WERKS	U715		Plant

Fig-18: Transaction code RSA3



Extractor Checker S-API

ZBW_MD04
ZBW_MDEZ

Date(DAT00	Del/finish	MRP el	MR	MR	Data(EXTRA)	Customer (K	Stor	Vendor(LIF	Material(MATNR)	Uni	Uni	Recpt/reqd(MNG01)	Avail. qty(MNG02)	P	Plan	Pl
21.07.2008		Stock	4B	PD					20111		PAC	10,209,030.400	10,209,030.400	B	U715	
13.09.2007	13.09.2007	Ord.DS	U1	PO	6000000064/00010		0010		20111		PAC	1.000-	10,209,029.400	-	U715	U7
13.09.2007	13.09.2007	Ord.DS	U1	PO	6000000065/00010				20111		PAC	100.000-	10,208,929.400	-	U715	U7
13.09.2007	13.09.2007	Ord.DS	U1	PO	6000000066/00010		0010		20111		PAC	1.000-	10,208,928.400	-	U715	U7
13.09.2007	13.09.2007	Ord.DS	U1	PO	6000000067/00010				20111		PAC	1.000-	10,208,927.400	-	U715	U7
13.09.2007	13.09.2007	Ord.DS	U1	PO	6000000068/00010		0010		20111		PAC	100.000-	10,208,827.400	-	U715	U7
13.09.2007	13.09.2007	Ord.DS	U1	PO	6000000069/00010				20111		PAC	100.000-	10,208,727.400	-	U715	U7
13.09.2007	13.09.2007	Ord.DS	U1	PO	6000000070/00010		0010		20111		PAC	100.000-	10,208,627.400	-	U715	U7

Fig-19: ZBW_MD04 Extractor output in t-code RSA3

BW Data Modeling and Reporting

The BW development involves creation of Infosource and Virtual Infocube.

Let’s create Infosource first with Infoobjects Material, Plant, Vendor, Customer, Calendar month/year, MRP element etc. Please add Infoobjects as mentioned in the below table to the Infosource and map it to the fields of datasource ZBW_MD04 in transfer rules. There is need to create few new Infoobjects as well.

Infoobjects	Data Type	Length	Description	Field mapping in Transfer Rule
Z_MRPIND	CHAR	2	MRP element	DELKZ
Z_PLUMI	CHAR	1	Receipt/issue indicator	PLUMI
Z_REQD	DATS	8	Receipt/Requirements date	DAT00
Z_FIND	DATS	8	Delivery/order finish date	DAT01
Z_MRPABR	CHAR	6	Abbreviation for MRP element	DELB0
Z_MRPELM	CHAR	40	MRP element data	EXTRA
Z_REQQTY	QUAN	13	Quantity received or quantity required	MNG01
Z_SCRQTY	QUAN	13	Available quantity	MNG02
0MOVE_PLANT	CHAR	4	Planning Plant	WRK02
0STOR_LOC	CHAR	4	Storage Location	LGORT
0VENDOR	CHAR	10	Account Number of Vendor or Creditor	LIFNR
0CUSTOMER	CHAR	10	Customer Number 1	KUNNR
0BASE_UOM	UNIT	3	Base Unit of Measure	MEINS
0MATERIAL	CHAR	18	Material Number	MATNR
0PLANT	CHAR	4	Plant	WERKS
0DISMM	CHAR	2	MRP Type	DISMM
Z_BERID	CHAR	10	MRP area	BERID
0SOURSYSTEM	CHAR	2	Source system ID	SOURSYSTEM
0MAT_PLANT	CHAR	18	Material Plant View	Routine
AH_RINQTY	QUAN	13	Required Inflow Quantity	Routine
AH_ROUQTY	QUAN	13	Required Outflow Quantity	Routine
0CALDAY	DATS	8	Calendar Day	Routine
0CALMONTH	NUMC	6	Cal. year / month	Routine
0CALMONTH2	NUMC	2	Calendar month	Routine

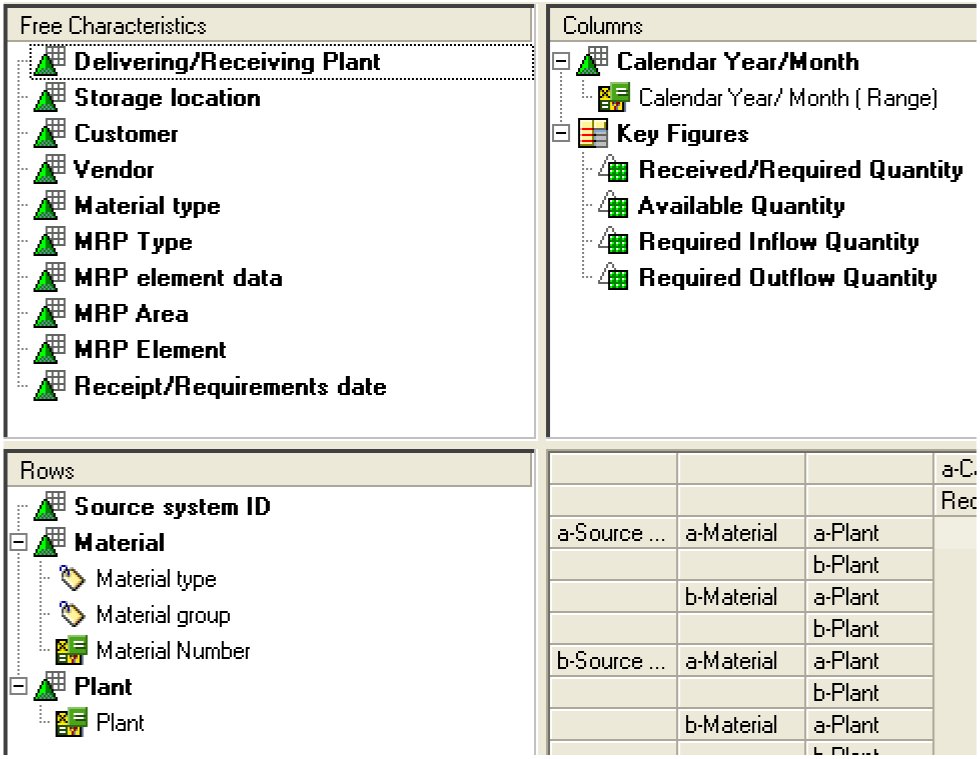
We need to write routine for some of the Infoobjects as mentioned below and then activate the Infosource.

```
* Routine for 0MAT_PLANT
  RESULT = TRAN_STRUCTURE-matnr.
* Routine for 0CALDAY
  IF NOT TRAN_STRUCTURE-dat01 IS INITIAL.
    RESULT = TRAN_STRUCTURE-dat01.
  ELSE.
    RESULT = sy-datum.
  ENDIF.
* Routine for 0CALMONTH
  IF NOT TRAN_STRUCTURE-dat01 IS INITIAL.
    RESULT = TRAN_STRUCTURE-dat01(6).
  ELSE.
    RESULT = sy-datum(6).
  ENDIF.
* Routine for 0CALMONTH2
  IF NOT TRAN_STRUCTURE-dat01 IS INITIAL.
    RESULT = TRAN_STRUCTURE-dat01+4(2).
  ELSE.
    RESULT = sy-datum+4(2).
  ENDIF.
* Routine for Z_RINQTY
  IF TRAN_STRUCTURE-plumi = '+' OR TRAN_STRUCTURE-plumi = 'B'.
    RESULT = TRAN_STRUCTURE-mng01 - TRAN_STRUCTURE-mng02.
  ENDIF.
* Routine for Z_ROUQTY
  IF TRAN_STRUCTURE-plumi = '-'.
    RESULT = TRAN_STRUCTURE-mng01 - TRAN_STRUCTURE-mng02.
  ENDIF.
```

This concludes our Infosource creation. Now we need to create Virtual Infocube.

Create a virtual Infocube of type ‘SAP Remotecube’ with all the Infoobjects mentioned in the Infosource. While creating the ‘SAP Remotecube’, we need to provide Infosource name. Here you need to put the name of Infosource which you have just created. Once Infocube is activated, we need to ‘Assign Sourcesystem’ to the Infocube as well, because this is SAP Remotecube. Assign the R/3 Sourcesystem to the Infocube. Please note this assignment of source system to Infocube is non-transportable. **You need to assign it manually in each system.**

This concludes creation of our BW modeling objects. Now we can create BW report based on this Infocube. You can create the report as shown in the *fig-20*. This will give the report output for Material and plant per calendar year/month.



Free Characteristics

- Delivering/Receiving Plant
- Storage location
- Customer
- Vendor
- Material type
- MRP Type
- MRP element data
- MRP Area
- MRP Element
- Receipt/Requirements date

Columns

- Calendar Year/Month
  - Calendar Year/ Month (Range)
- Key Figures
  - Received/Required Quantity
  - Available Quantity
  - Required Inflow Quantity
  - Required Outflow Quantity

Rows

- Source system ID
- Material
  - Material type
  - Material group
  - Material Number
- Plant
  - Plant

			a-C
			Rec
a-Source ...	a-Material	a-Plant	
		b-Plant	
	b-Material	a-Plant	
		b-Plant	
b-Source ...	a-Material	a-Plant	
		b-Plant	
	b-Material	a-Plant	
		b-Plant	

fig-20: MD04 Report in BW

This concludes creation of BW report for MD04 data pulled via virtual Infocube from R/3 datasource ZBW_MD04.

Related Content

https://www.sdn.sap.com/irj/sdn/weblogs
http://help.sap.com/saphelp_nw04/helpdata/en/84/fcb63d799f6b3be10000000a114084/frameset.htm

http://forums.sdn.sap.com/thread.jspa?threadID=1158776

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.