A to Z of Custom Change Pointer

TOPICS: BD50 BD52 BD60 BDCP2 Change Pointer SCDO

POSTED BY: SAP YARD OCTOBER 24, 2011

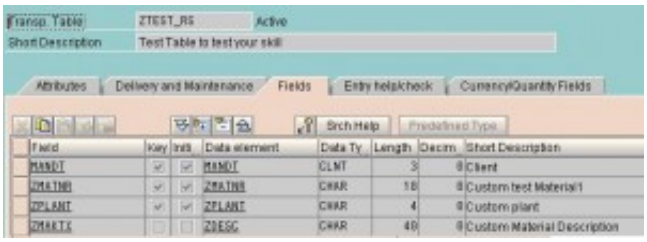
In this document, I have tried to provide every minute details needed to implement custom change pointers for the custom fields in the custom tables for custom message type and custom change object.

[adToAppearHere]

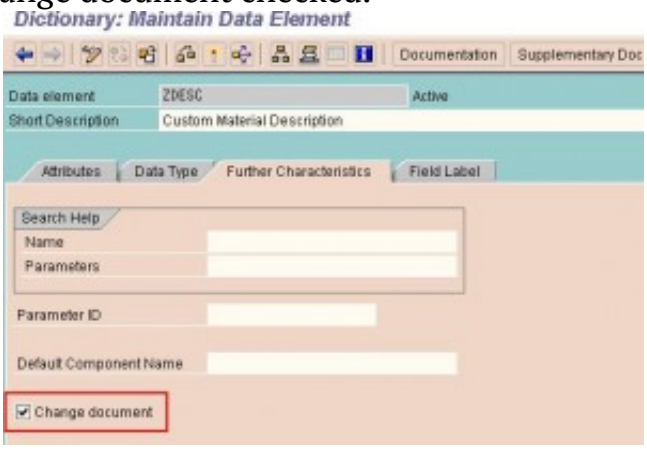
Configurations and Steps to implement custom change pointer:

Step 1. Make the data type of custom fields ready to record changes.

For demo, I have used the below custom table with custom fields.

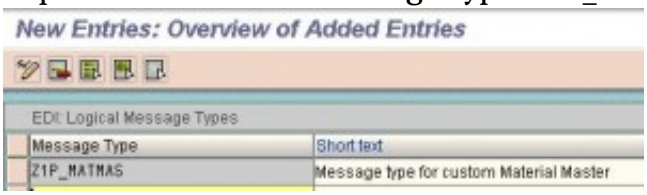


The data elements of all the three custom fields have the Change document checked.

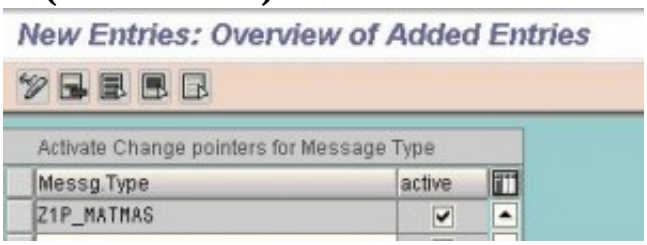


Step 2. Create custom message type (T-code WE81).

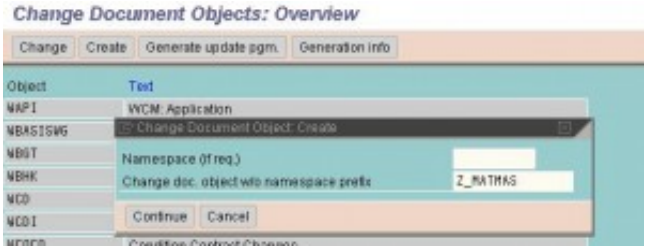
For example I have created message type Z1P_MATMAS.



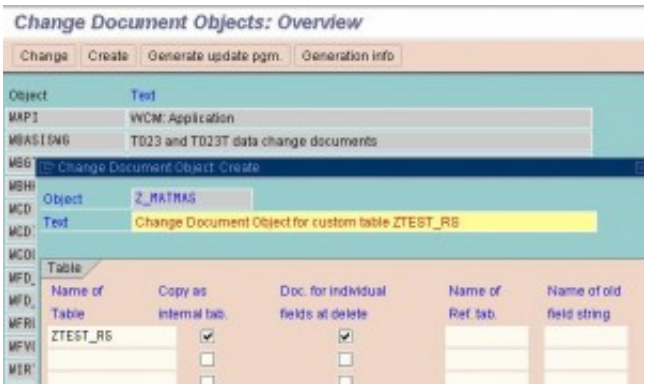
Step 3. Activate this message type to catch change pointers (T-code BD50).



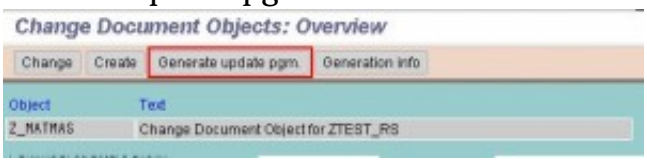
Step 4. Create custom Change Document Object to link changes (T-code SCDO).

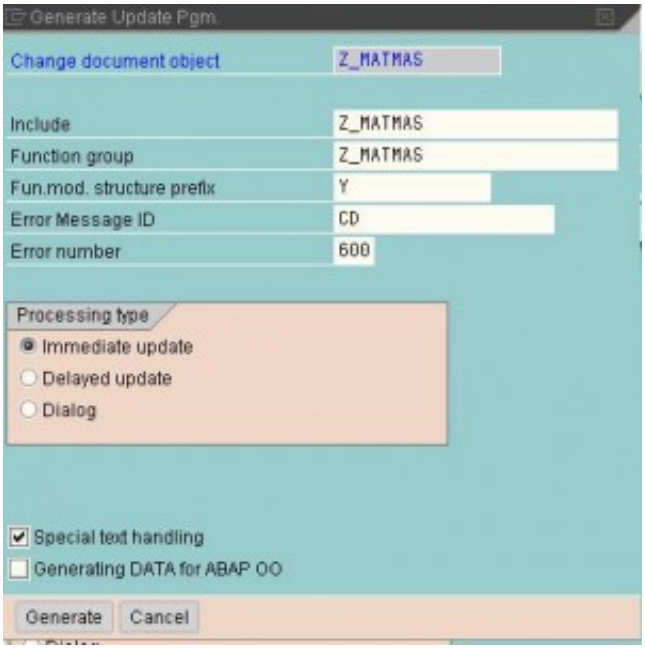


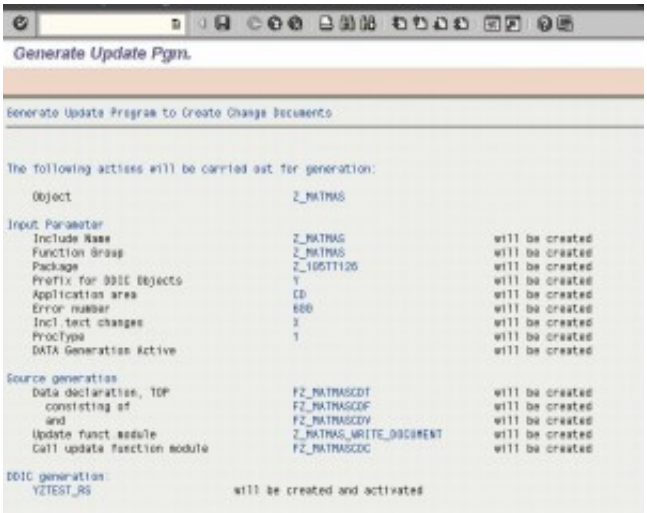
Put the custom table name (ZTEST_RS for this demo).
You can add more than one tables



Step 5. Generate the custom Function Module to update changed documents (T-code SCDO).
Click Generate update pgm button.

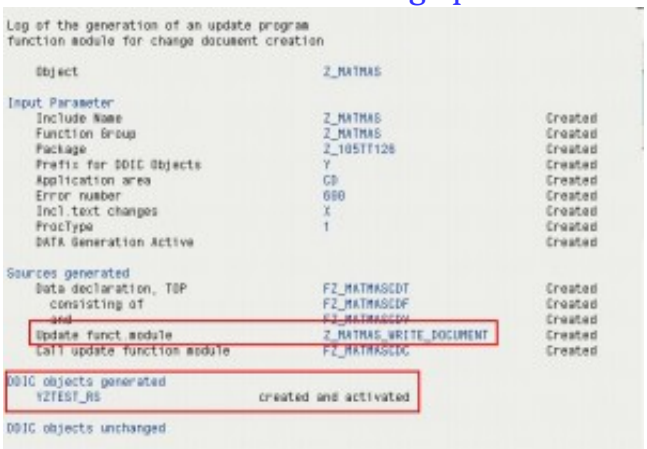




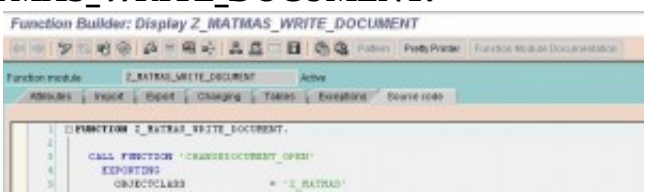


Save it.

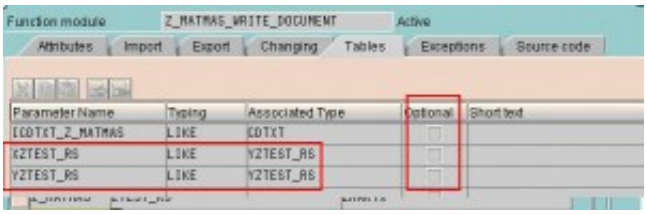
FM **Z_MATMAS_WRITE_DOCUMENT** to update change pointer is created and a new data dictionary structure YZTEST_RS is also created. FM Z_MATMAS_WRITE_DOCUMENT would be called from the custom code to trace the change pointers.



Step 6. Check the generated FM Z_MATMAS_WRITE_DOCUMENT.



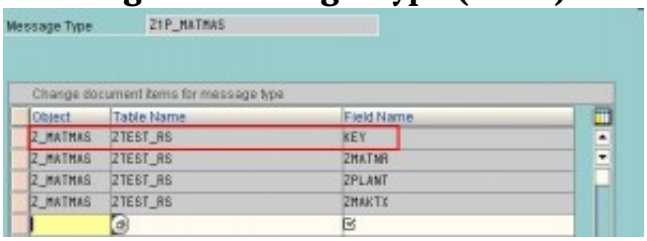
Check the Tables section. Two tables are defined. One is for data before change and the other is for data after change.



Please note: by default the Tables in the FM are mandatory. If you added more than one tables in the transaction SCDO while generating the FM, you can

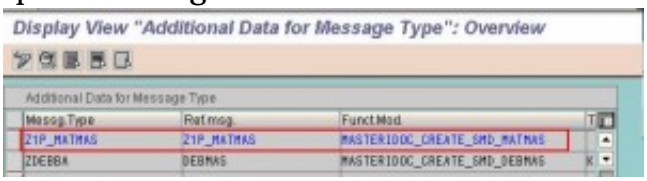
make these tables as optional manually in SE37, so that you can use the same FM for change pointers for multiple tables.

Step 7. Link the change document Object, Table and Fields for the given Message Type (BD52).



Please note, field name KEY should be given for every table even though it is not field of any table.
The message type is directly referenced to fields and tables of the material master except for the KEY field. This field isn't part of the respective table but assumes a very important, additional control role. If the KEY field is specified in Transaction BD52, a change pointer is written during the creation of the corresponding object.

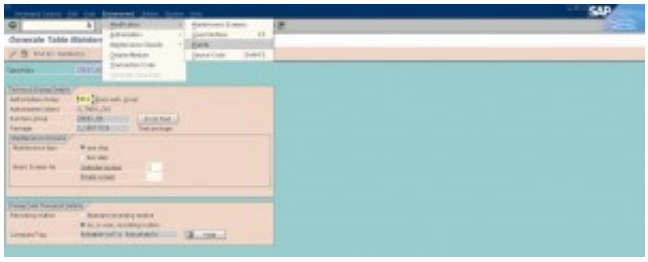
Step 8. Evaluating Change Pointers (T-code BD60).
The change pointers are then evaluated. It just depends on the object concerned which function module is used here.
The function module, MASTERIDOC_CREATE_SMD_MATMAS, which uses change pointers to generate IDocs is called.



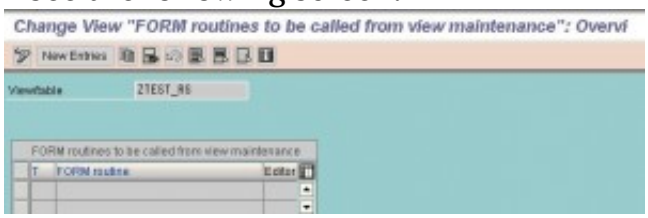
Step 9. Write the custom code in the TMG event to capture changed/created data. (This is not the only way. You can write your code in other suitable areas)

Details to add event is also there in other post:
http://help-sap.blogspot.com/search?q=event

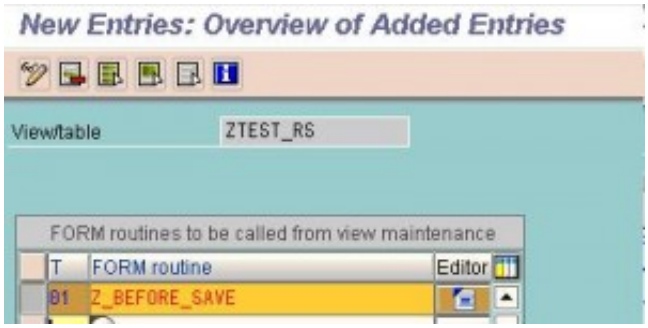
In the Generate Table Maintenance screen go to Menu Environment->Modification->Events.



You will see the following screen.



Click the New Entries button and select the event you want to use and the Routine name.



Click the Editor and create your routine in the Include program and write the code in the routine to meet your requirement.

[adToAppearHereLink]

Sample Code to catch change pointers

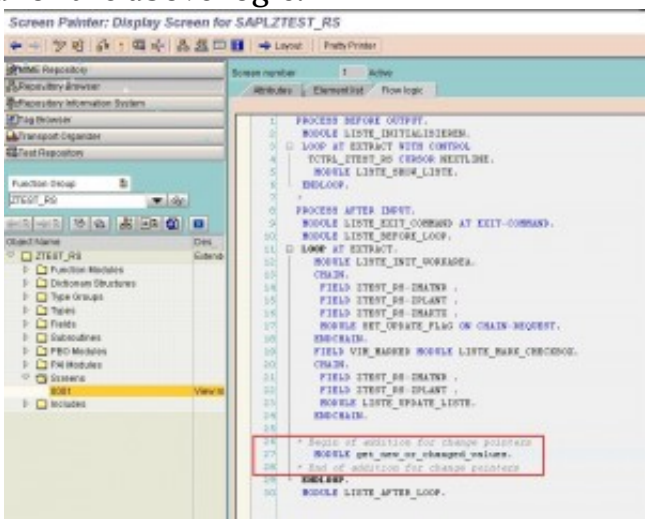
```
FORM z_before_save.

CONSTANTS:
c_insert TYPE cdchngind VALUE 'I',
c_update TYPE cdchngind VALUE 'U',
c_check TYPE boolean VALUE 'X'.

DATA: li_ztest_rs_old TYPE STANDARD TABLE OF
ztest_rs INITIAL SIZE 0,
li_xztest_rs TYPE STANDARD TABLE OF yztest_rs
INITIAL SIZE 0,
li_yztest_rs TYPE STANDARD TABLE OF yztest_rs
INITIAL SIZE 0,
li_cdtext TYPE STANDARD TABLE OF cdtxt INITIAL
SIZE 0,
```

```
lk_ztest_rs_x TYPE ztest_rs,
lk_ztest_rs_old TYPE ztest_rs,
lk_xztest_rs TYPE yztest_rs,
lk_yztest_rs TYPE yztest_rs,
lv_object TYPE cdobjectv.

* Sort
SORT i_ztest_rs_x[] BY zmatnr zplant.
* This will have the most recent data
DELETE ADJACENT DUPLICATES FROM i_ztest_rs_x[]
COMPARING zmatnr zplant.

* Initial check
IF i_ztest_rs_x[] IS NOT INITIAL.

* Get old data
SELECT * FROM ztest_rs
INTO TABLE li_ztest_rs_old " Old Data
FOR ALL ENTRIES IN i_ztest_rs_x
WHERE zmatnr = i_ztest_rs_x-zmatnr
AND zplant = i_ztest_rs_x-zplant.
* Looping through the current data set
LOOP AT i_ztest_rs_x INTO lk_ztest_rs_x.

* Read the old data if available
READ TABLE li_ztest_rs_old INTO lk_ztest_rs_old
WITH KEY zmatnr = lk_ztest_rs_x-zmatnr
zplant = lk_ztest_rs_x-zplant.

IF sy-subrc EQ 0.
* If found then check whether anything is changed or
not
IF lk_ztest_rs_x-zmatnr NE lk_ztest_rs_old-zmatnr
OR lk_ztest_rs_x-zplant NE lk_ztest_rs_old-zplant
OR lk_ztest_rs_x-zmaktx NE lk_ztest_rs_old-zmaktx .

MOVE-CORRESPONDING lk_ztest_rs_x TO lk_xztest_rs.
* This is important
lk_xztest_rs-kz = c_update.
* Keep the new data
APPEND lk_xztest_rs TO li_xztest_rs[].

MOVE-CORRESPONDING lk_ztest_rs_old TO
lk_yztest_rs.
* This is important
lk_yztest_rs-kz = c_update.
```

```
* Keep the old data
APPEND lk_yztest_rs TO li_yztest_rs[].

lv_object = lk_ztest_rs_x-zmatnr.

* Write the change pointer for changed record
CALL FUNCTION 'Z_MATMAS_WRITE_DOCUMENT'
EXPORTING
objectid = lv_object
tcode = sy-tcode
utime = sy-uzeit
udate = sy-datum
username = sy-uname
upd_ztest_rs = c_update
TABLES
icdtxt_z_matmas = li_cdtext[]
xztest_rs = li_xztest_rs[]
yztest_rs = li_yztest_rs[].

REFRESH: li_cdtext[], li_xztest_rs[], li_yztest_rs[].
CLEAR: lv_object.
ENDIF.
* If not found, that means it is a new entry
ELSE.
MOVE-CORRESPONDING lk_ztest_rs_x TO lk_xztest_rs.
* This is important
lk_xztest_rs-kz = c_insert.
* Keep the new data
APPEND lk_xztest_rs TO li_xztest_rs[].

lv_object = lk_ztest_rs_x-zmatnr.

* Write the change pointer for new record/insert record
CALL FUNCTION 'Z_MATMAS_WRITE_DOCUMENT'
EXPORTING
objectid = lv_object
tcode = sy-tcode
utime = sy-uzeit
udate = sy-datum
username = sy-uname
upd_ztest_rs = c_insert
TABLES
icdtxt_z_matmas = li_cdtext[]
xztest_rs = li_xztest_rs[]
yztest_rs = li_yztest_rs[].
```

REFRESH: li_cdtext[], li_xztest_rs[], li_yztest_rs[].
CLEAR: lv_object.
ENDIF.
ENDLOOP.
ENDIF.
ENDFORM. “z_before_save

In order to feed data in the event above, we nee to populate data (I_ZTEST_RS_X[ ]) in the custom database table screen. Write code to keep the run-time table data needed for the above logic.



MODULE get_new_or_changed_values INPUT.
* Keep the screen number
CONSTANTS: c_screen_0001 TYPE sydynnr VALUE ‘0001’.

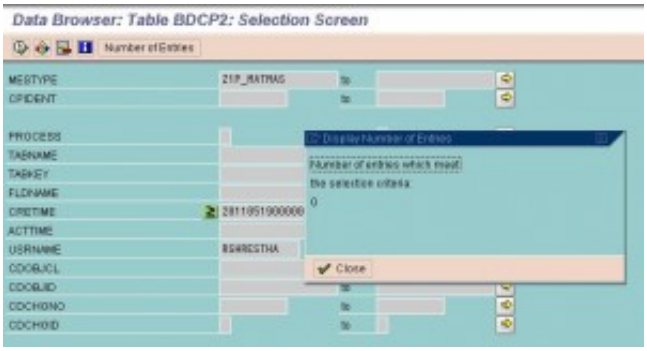
* Get the data and keep in internal table to be used later
IF sy-dynnr = c_screen_0001.
APPEND ztest_rs TO i_ztest_rs_x.
ENDIF.

ENDMODULE.

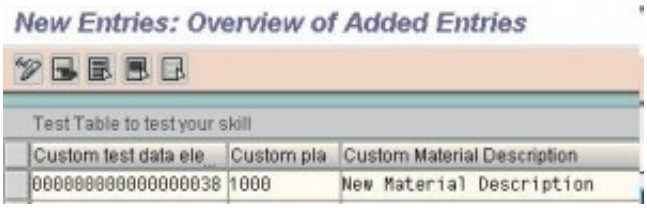
You may need to declare some global data.
This completes the configuration and custom code to catch the change detection.

Sample Output to register change pointers for Create and Change:

The changes are logged in the standard table BDCP2.
Let us check for any entry in the table for my userid and today’s date

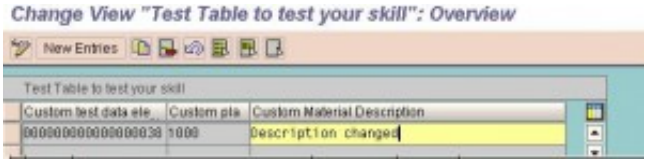


No entry is found.
Now I will Create/Add an entry in the custom table by t-code SM30 and save it.



Go to table BDCP2 and look for records. Check one entry is successfully logged.

Change an entry and check the table for the change pointer.



Check another entry has been added.

Transactions to remember

1. SE11 to check the data type is active to record changes and to create events.
2. SE81 to create message type.
3. BD50 to activate the message type to catch change pointers.
4. SCDO to create Change Objects and generate program/function module to write change pointers.
5. BD52 to link Message Type, Change Objects, Tables and Fields.
6. BD60 to link Message Type to Standard Function Module.

Point to remember

1. In transaction BD52, field name ‘KEY’ should be

given for every table even though it is not a field of any table.

If you liked this post, you might like to check our other post on *Idocs* ***'You'hv got an IDoc !!'***.

Image source : www.nbc.com & www.vectorstock.com (modified)

Related Posts:

How to update custom field of PRPS table

Change Impact Analysis with SAP Solution Manager

Sales Office Data … Can you change it even if config…

Create & Change Variants without Fire Fighter

How to change the Maintenance Plan Category in T-code IP02?

Isn't it Fair to Share??

Previous post Next post

4 COMMENTS ON "A TO Z OF CUSTOM CHANGE POINTER"

rekha kommaka | July 31, 2014 at 6:21 am |

Nice post.This is very useful information.Thanks for sharing.

shivanagh | August 7, 2015 at 9:15 pm |

Very Nice .. Thanks for sharing..

SAP Yard | August 7, 2015 at 9:44 pm |

You are welcome Shiva!!

Regards,
Raju

VK | February 13, 2016 at 5:30 pm |

In My requirement I Dont have TMG, need to evaluate change pointers based on custom field change in XD02. When I put breakpoint inside Events in TMG, on changing field in XD02 not triggering the 01 event.

Comments are closed.