**ABAP: Change Document**

**Applies to:**
This document applies to SAP ECC 6.0, SAP Netweaver 2004s. For more information, visit the [ABAP homepage](http://abap.sap.com).

**Summary**
This article contains the step by step procedure to guide you to make change log entries in CDHDR and CDPOS tables in SAP ECC 6.0 version.

The requirement is to enter change log entries for custom tables.

**Author:** Rahul Muraleedharan  
**Company:** Applexus Software Solutions (P) Ltd  
**Created on:** 22 September 2010

**Author Bio**
Rahul Muraleedharan is working as SAP Technology Consultant with Applexus Software Solutions (P) Ltd. He has an experience of over 1 year in SAP ABAP.
Table of Contents

Introduction .................................................................................................................................................. 3
  Steps for creating change log entries: .................................................................................................. 3
Step by Step Process .................................................................................................................................. 4
Related Content ......................................................................................................................................... 14
Disclaimer and Liability Notice ............................................................................................................... 15
Introduction

In a transaction a lot of database updates will occur. It will be very useful if we are able to trace the changes made. For this purpose we can make use of the change document tables viz., CDHDR and CDPOS. The change log entries in Change Document tables can be either actual changes or planned changes.

Steps for creating change log entries:

1. Create an object class.
2. Set the change document flag.
3. Include the R3TR generated include files in the program.
4. Call the R3TR generated update function module where the database table is getting updated.
5. Pass the initial values of the records which are getting updated in the database table to the update function module.
6. Pass the new values of the records to the update function module.
Step by Step Process

1. Go to the transaction SCDO for creating an object class.

   All the tables should be included in an object class in order to log the data changes in the change document tables.

   A popup will be displayed asking the namespace and the name of the object class to be created.

   Namespace is an optional field but the name of the object class is a mandatory field. Enter a suitable name for the object class starting with ‘Z’ and press CONTINUE.

   If you don’t enter the namespace then a warning message will be shown as a popup, press YES.
2. A new popup will be displayed. Here you have to give the following details:
   a) A short text for the change document object.
   b) Name of the database tables to be included in the object class whose data changes are to be logged.
   c) The check box for ‘Copy as internal tab.’ has to be checked if multiple records of the database table will be updated at the same time (multiple case).
   d) The check box for ‘Doc. for individual fields at delete’ has to be checked if we want separate log entry for each field when a particular(s) record is deleted.
   e) If the currency and unit fields are defined in a reference table, rather than in the table passed, you must pass the name of the reference table, and the field referred to, to the function module. Create an INTTAB structure in the Dictionary, and define fields for this structure, which are made up of the names of the associated reference table and the reference fields.
       Enter the name of this structure in the field for ‘Name of ref. tab’.
   f) The field for ‘Name of the old record fields’ has significance only if we haven’t checked the checkbox for ‘Copy as internal tab.’. Even then this field is optional. If it’s not filled then the name of the work area which holds the old record will have the name *work area otherwise the name the name which we have specified in that field.

After filling all the fields required press **INSERT ENTRIES**.

![Change Document Object: Create](image)

<table>
<thead>
<tr>
<th>Table</th>
<th>Name of Table</th>
<th>Copy as internal tab</th>
<th>Doc. for individual fields at delete</th>
<th>Name of Ref tab</th>
<th>Name of old field string</th>
</tr>
</thead>
<tbody>
<tr>
<td>zrh_flight</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Here ZRH_FLIGHT is the database table for which the data changes are to be logged. All other fields are optional and it can be set according to the requirement. Here one record will be updated to the database table at a single time. If we are updating the table through an internal table then the checkbox for ‘Copy as internal tab.’ has to be checked.

**Note:** We can include more than one database table in a single change doc object
3. Now we should decide for which all fields in the database tables we require the log entries. And the change document flag for the corresponding data elements have to be checked. The change document property is to be set in the ‘Further Characteristics’ tab of the data element.

![Dictionary: Maintain Data Element](image-url)
4. Now the change doc object is to be saved.

5. Go to the Utilities in the menu bar. Select **GENERATE UPDATE PGM**. A pop up will be displayed saying that the change doc object is not in your namespace if you have not specified it in the initial screen. Press **YES**.

Now a dialog box will be displayed.

Here you have to enter:

a) **Include Name (Maximum 26 character):** This name is used to complete the name of the R3TR generated INCLUDE programs.
b) Function group: Enter the name of the function group to which the change document update program is to belong. If this function group does not yet exist in the system, it is automatically created during generation. Exactly one function group must belong to each change document object. Other function modules may not be assigned to this function group.

c) Fun.mod. structure prefix: For multiple cases table transfer structures are created at generation. Their names are constructed from the prefix and the name of the table.

Note: Since this will be a parameter to an update function module its length should be longer than 28 characters. So consider the table name when you set the prefix.

d) Error message ID: The application-specific error messages generated are stored under this message ID (work area). A value is proposed.

e) Error Number: Number with which errors occurring in connection with this change document object can be identified in the system. A value is proposed.

f) Processing type: The processing type of the function module should be selected from any of the three options viz., Immediate Update, delayed Update, Dialog.

g) Special Text handling Flag: This should be checked if long text changes are to be logged. The old and new status of long texts is not logged. Only the fact that they have been changed is noted.

h) Generating DATA for OO: This is an alternative for generating data and tables statements for ABAP OO environment.

After filling all the required fields press **GENERATE**.

When you press generate button a pop up will be displayed if the function group that you have Specified is not yet created. Press **YES**. Now the system will show all the actions that will be carried out for generation.
Generate Update Pgm.

The following actions will be carried out for generation:

<table>
<thead>
<tr>
<th>Object</th>
<th>ZRH_CHNG_OBJ</th>
</tr>
</thead>
</table>

Input Parameter

<table>
<thead>
<tr>
<th>Include Name</th>
<th>ZRH_CHNG_OBJ</th>
<th>will be created</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Group</td>
<td>ZRH_FLIGHT</td>
<td>will be created</td>
</tr>
<tr>
<td>Package</td>
<td>ZRH_DEMO</td>
<td>will be created</td>
</tr>
<tr>
<td>Prefix for DDIC Objects</td>
<td>Y</td>
<td>will be created</td>
</tr>
<tr>
<td>Application area</td>
<td>CD</td>
<td>will be created</td>
</tr>
<tr>
<td>Error number</td>
<td>600</td>
<td>will be created</td>
</tr>
<tr>
<td>Incl. text changes</td>
<td></td>
<td>will be created</td>
</tr>
<tr>
<td>Protype</td>
<td>2</td>
<td>will be created</td>
</tr>
<tr>
<td>DATA Generation Active</td>
<td></td>
<td>will be created</td>
</tr>
</tbody>
</table>

Source generation

<table>
<thead>
<tr>
<th>Data declaration, TOP</th>
<th>FZRH_CHNG_OBJDOT</th>
<th>will be created</th>
</tr>
</thead>
<tbody>
<tr>
<td>consisting of</td>
<td>FZRH_CHNG_OBJDF</td>
<td>will be created</td>
</tr>
<tr>
<td>and</td>
<td>FZRH_CHNG_OBJDV</td>
<td>will be created</td>
</tr>
<tr>
<td>Update funct.module</td>
<td>ZRH_CHNG_OBJ_WRITE_DOCUMENT</td>
<td>will be created</td>
</tr>
<tr>
<td>Call update function module</td>
<td>FZRH_CHNG_OBJDC</td>
<td>will be created</td>
</tr>
</tbody>
</table>

DDIC generation:

no actions

Press the SAVE button.
All the details of the update program generated are displayed. The INCLUDE program generated are F<include name>DT, F<include name>DF, F<include name>DV and F<include name>DC. F<include name>DT includes programs F<include name>DF and F<include name>DC. F<include name>DF includes all declaration of all the variables created. F<include name>DV includes all the work areas and internal tables created. F<include name>DC includes the form routine for calling the update function module <Include Name>_WRITE_DOCUMENT. The update function module <Include Name>_WRITE_DOCUMENT calls the following object specific parameters.

CHANGEDOCUMENT_OPEN
CHANGEDOCUMENT_SINGLE_CASE – If single case
CHANGEDOCUMENT_MULTIPLE_CASE – If multiple case
CHANGEDOCUMENT_TEXT_CASE – If Special Text handling Flag is set.
CHANGEDOCUMENT_CLOSE
Include the include program in the custom program. Include all the R3TR generated in the custom program.

```
ZRH_CHANGE_DOC
```

And call the subroutine which includes the update function module where the database updation takes place.
Before calling the subroutine set all the input parameters for the update function module viz.,

a) **OBJECTID**: A parameter to identify a unique record of a table usually all the primary key values are concatenated together and given.

b) **TCODE**: The transaction in which the updation happens.

c) **UDATE**: Change date.

d) **UTIME**: Change time.

e) **USERNAME**: User responsible for the change.

f) **UPD_<Table Name>**: Update Flag (Insert, update or delete).

Also the old value and new values have to be passed to the update function module.

**Note**: Since it’s an update function module COMMIT WORK should be called after the execution of the function module.
7. Execute the transaction.

Sample Scenario:

a) Inserting a new record to the flight details table.

On saving the record a new entry will be created in CDHDR and CDPOS tables.

b) On modifying an existing record of the flight details.

CDPOS will have new entries. CDHDR will have a single new record.
c) On deleting an existing record.

CDPOS and CDHDR will have one more entry.
Related Content

Need to log entries in CDPOS and CDHDR

Change Log

Change Documents

For more information, visit the ABAP 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.