Applies to:
SAP BI 7.0, SAP ABAP, For more information, visit the Business Intelligence homepage.

Summary
The objective of the article is to outline & explain the steps to enhance or create a data source in context to SAP BI in its source system (SAP R/3, ECC, etc).

Authors: Seema John and Tapan Kumar Jain
Company: Accenture
Created on: 11th March 2010

Author Bio
Seema John is working as SAP BI Consultant in Accenture Services Private Ltd and having extensive experience in implementation of BI projects specializing in HR areas.

Tapan Kumar Jain is working as SAP BI Consultant in Accenture Services Private Ltd and having extensive experience in implementation of BI projects across various SCM areas.
# Table of Contents

Pre-Requisites .......................................................................................................................................................... 3  
Enhancing a Datasource in ECC/SRM using Tcode RSA6 .................................................................................. 3  
Related Content ...................................................................................................................................................... 10  
Disclaimer and Liability Notice .......................................................................................................................... 11
Pre-Requisites
- Basic understanding of SAP ABAP programming language
- Basic understanding of dataflow in SAP BI 7.0.

As part of this article, we will have a detailed look at the enhancement process of an existing datasource.

Enhancing a Datasource in ECC/SRM using Tcode RSA6

The first step involves logging onto Tcode RSA6 and finding the relevant datasource which will then be enhanced as per the additional requirements.

Search for the required Datasource in the datasource list:

Click on enhance extraction structure:
An append structure needs to be created which will contain all the enhanced fields along with their metadata definition. Give a name to the append structure starting with 'Z':
Define the structure with the fields:

![Image of the Dictionary: Maintain Append Structure](image1.jpg)

Activate the append structure and save the datasource:

![Image of the Postprocess DataSources and Hierarchy](image2.jpg)
Click on change datasource and unhide the new fields if required:

Post activation, one needs to write the exit code in the source system to populate the source fields. The same is done through Tcode CMOD wherein you can then refer to the associated project and log onto enhancement RSAP0001.

The following four exits exists and will be enhanced depending on the requirements:
- EXIT_SAPLRSAP_001 – Transaction data
- EXIT_SAPLRSAP_002 – Master Data Attributes
- EXIT_SAPLRSAP_003 – Master Data Text
- EXIT_SAPLRSAP_004 – Master Data Hierarchy

Refer to screenshots below:

Tcode: CMOD:

Project Management of SAP Enhancements
Enhancement for writing code for enhanced fields of datasource:

### SAP Enhancements in Enhancement Project ZBIDEV

<table>
<thead>
<tr>
<th>Enhancement</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Different User Exits:

<table>
<thead>
<tr>
<th>Project</th>
<th>Enhancement assignments</th>
<th>Enhancement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ZBIDEV - Enhancement for DS GOE_D or ALT финанс Dispute Amount</td>
<td></td>
</tr>
<tr>
<td>Enhancement</td>
<td>Impl</td>
<td>Exp</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RSAP0001 Customer function calls in the service API</td>
</tr>
<tr>
<td>Function exit</td>
<td></td>
<td>EXIT_SAPLSAP_B81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXIT_SAPLSAP_B82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXIT_SAPLSAP_B83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXIT_SAPLSAP_B84</td>
</tr>
</tbody>
</table>
Sample Code follows:

```
FUNCTION EXIT_SAPLSAP_001

** Lokale Schmittsteile:
** IMPORTING
**  VALUE(l_DATASOURCE) TYPE R3AD_QQ_DATASOURCE
**  VALUE(l_SOURCE) TYPE SDIAR_S_INTERFACE-INTERFACE
**  VALUE(l_UP_MODE) TYPE SDIAR_S_INTERFACE-UPMOD
** TABLES
**  _T_SELECT TYPE SDIA_T_SELECT
**  _T_FIELDS TYPE SDIA_T_FIELDS
**  _T_DATA
**  _T_MESSAGES STRUCTURE BCM: OPTIONAL
** EXCEPTIONS
**  ZRPA0SAU01
**

ENDFUNCTION.
```

The include as shown below will contain all the relevant code controls which define the field population. This is a generic include and the code piece needs to be controlled through datasource name as shown below in the sample code (screenshot). The existing fields along with their values are stored in the internal table C_T_Data which is to be further modified to also include and populate values for the newly enhanced and added fields.

Further modularization can be done by creating datasource specific includes. The technique also helps in better readability of the code.
Once the changes are done & new fields are fetched, internal table C_T_Data needs to be modified accordingly. The modifications and the new fields population is then automatically transferred to the fields of the datasource. The same can then be validated by executing the datasource through Tcode RSA3.

```plaintext
^ Internal table declarations
data : it_YOWO type standard table of t_YOWO,
it_t type standard table of t_pspid,
it_ICWBSCST type standard table of ICWBSCST.

^ Work area
data : ls_ICWBSCST TYPE ICWBSCST,
     ls_YOWO type t_YOWO,
     ls_t type t_pspid.

^ Local variables
data : lv_pspnr type ZIT_RPS_VYWD-pspnr.

lt_ICWBSCST[] = c_t_data[].
sort lt_ICWBSCST by pspid posid fispcer.
delete adjacent duplicates from lt_ICWBSCST comparing pspid posid fispcer.

loop at lt_ICWBSCST into ls_ICWBSCST.
  ls_t-pspid = ls_ICWBSCST-pspid.
  ls_t-monat = ls_ICWBSCST-fispcer.
  CALL FUNCTION 'CONVERSION_EXIT_ABPS_P_INPUT'
    EXPORTING
      input = 1s_ICWBSCST-posid
      importing
      output = 1s_t-pspnr
    EXCEPTIONS
      not_found = 1
      others = 2
    append ls_t to lt_t.
endloop.

1 Get all Value of Work Done
  when lt_t[] is initial
    select pspid pspnr monat vwd from ZIT_RPS_VYWD
    into table lt_t
    for all entries in lt_t
      where pspid = lt_t-pspid and
        pspnr = lt_t-pspnr and
        monat = lt_t-monat.
    if sy-subrc eq 0.
      sort lt_t-vwd.
    endif.
  endif.
  clear : 1s_ICWBSCST,lv_tabix.

2 Modify values in data
  loop at c_t_data into ls_ICWBSCST.
    clear lv_pspnr.
    lv_tabix = sy-tabix.
    CALL FUNCTION 'CONVERSION_EXIT_ABPS_P_INPUT'
      EXPORTING
        input = 1s_ICWBSCST-posid
        importing
        output = lv_pspnr
      EXCEPTIONS
        not_found = 1.
    if sy-subrc eq 0.
      read table lt_t-vwd into 1s_vwd with key pspid = 1s_ICWBSCST-pspid
        pspnr = lv_pspnr
        monat = 1s_ICWBSCST-fispcer
        binary search.
      if sy-subrc eq 0.
        1s_ICWBSCST-vywd = ls_vwd-vywd.
        MODIFY c_t_data FROM ls_ICWBSCST INDEX lv_tabix.
      endif.
    endif.
  endloop.
```
Related Content

Help.sap.com

Book - > ABAP Development for SAP BW by Dirk Herzog

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.