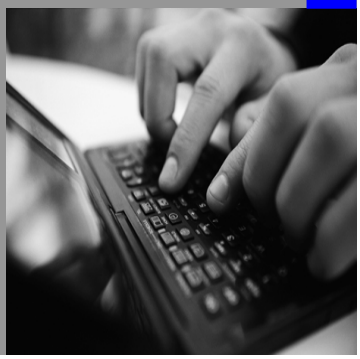


How-to Guide
SAP NetWeaver 7.0 (2004s)

How To... Line Items in BI Integrated Planning

Version 2.00 – February 2008

Applicable Releases:
SAP NetWeaver 7.0
Business Information Management
Business Planning and Analytical Services

1 Scenario

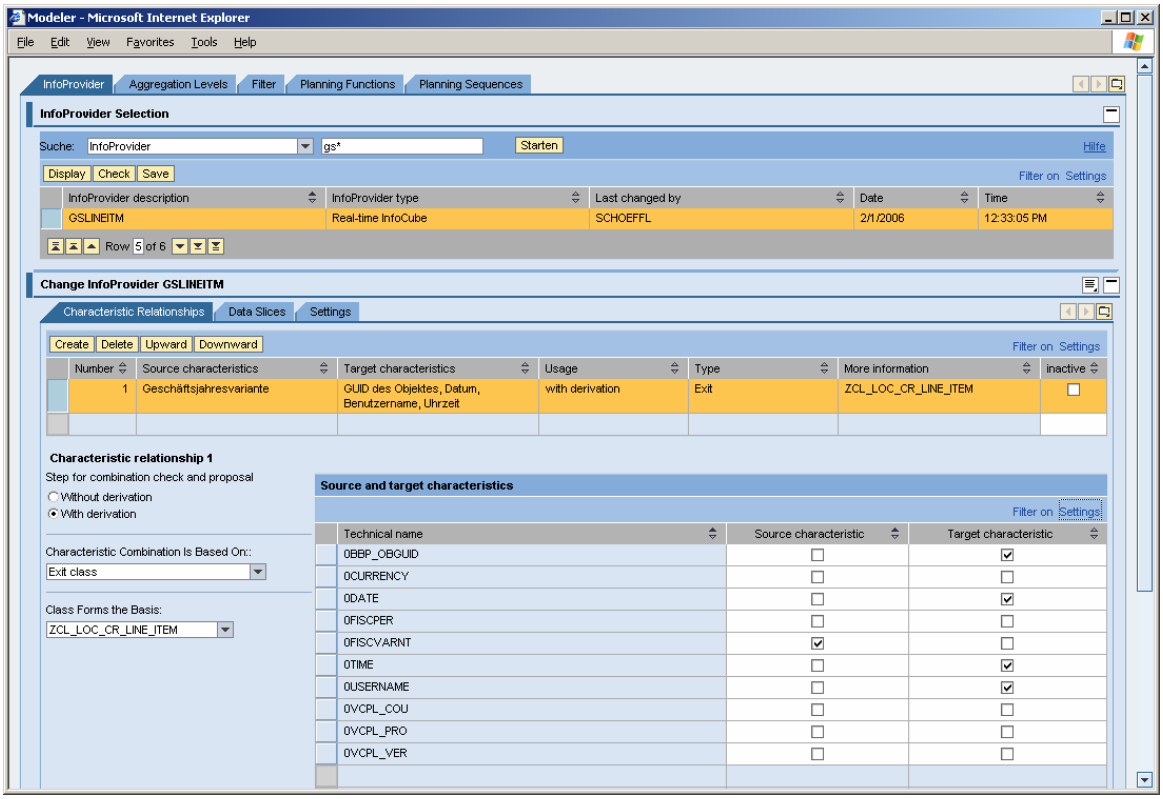
In the context of the delivered standard functions, it is not possible to reconstruct the history of a plan data record after the event in BI Integrated Planning. For example, if a sales target is planned for a business area for a certain time frame, you can no longer see when and from whom this sales target was actually entered. It is also not possible to see whether this value was planned directly, or whether the value was formed from targets, changes and corrections in the context of a long process.
However, if such proof is desired/requested in particular cases, then it can be realized by proceeding as follows.

2 Concept of the Solution

The solution described here uses the technology of characteristic derivation, which is a part of the standard functionality in BI Integrated Planning.
The “normal” application case of characteristic derivation consists of deriving the characteristic customer group from the characteristic group, or the company code currency from the company code.
In the present case, this interface is used to make a unique identification of the current data record. Since in BI Integrated Planning a delta posting is carried out internally, this unique identification means that the complete history of a plan figure is logged to the current characteristic combination.
You can find more details on the topic of characteristic derivation in the BI Integrated Planning documentation under the keyword Characteristic Derivation

3 The Step By Step Solution

To provide every data record with a unique identification, the Realtime InfoCube used for planning must contain a suitable InfoObject or if necessary be extended by one. A characteristic of the type Char with length 32 without a check table or text table (for example, 0BBP_OBGUID) is suitable for this, which can be filled with a GUID (globally unique identifier) during characteristic derivation. Alternatively, this unique identification can also be achieved by pulling a number from a number range or similar techniques. In each case, a suitable InfoObject must be available in the InfoCube. In addition to that, other fields can also be included in the InfoCube to save additional organizational data, for example, 0UNAME (created by), 0DATE (date) and 0TIME (time).
On the side of BI Integrated Planning, the characteristic relationships must be maintained in the Planning Modeler for the InfoCube as shown below. The derivation itself is done in an Exit that has to be implemented as a class.



Here the source characteristic of the derivation can be any characteristic which is contained each of the aggregation levels used. All characteristics which are to be filled with control data are to be entered as target characteristics. In the example above, these are 0BBP_OBGUID, 0USERNAME, 0DATE and 0TIME. These target characteristics are not to be included in any of the aggregation levels used, since they are automatically filled by the characteristic derivation (here: class ZCL_LOC_CR_LINE_ITEM).

In the class you have to implement the method ‘derive’ in accordance with the data model used. It is important that also to other required methods (for combination check and combination proposal) are implemented. The class CL_RSPLS_CR_EXIT_BASE is a delivered template that can be used as a copy template or from which the class can inherit. It is highly recommended to use this class in one of these ways and then re-implement the derive method as shown below. When using the template class you do not have to do anything about the combination proposal and combination check method.

If the fields used in the example (0BBP_OBGUID, 0UNAME, 0DATE and 0TIME) are used exactly, then this module can also be used directly. If you use other characteristics the coding has to be adopted accordingly.

Note: Modifying this class for the purposes of adjustment can reduce the stability of the entire system! Even if no static usage is displayed, a dynamic usage cannot be ruled out!

After these preparations, you can plan as normal with the existing data model.

4 Result

With the technique described above, for example the history of data

Amount in 2007

30.000,00 USD

can be presented in all single steps as below:

Fiscal year	2007	
Cost center	4711	
Currency	USD	US Dollar

Date	Time	Created By	Object GUID	Amount
27.08.2006	14:01:52	HORNC	3D1FD47DB098D511AE780800062AFB0F	50.000,00 USD
11.01.2006	13:30:44	KOLATA	8082E2EB7C803F40B5797895DBF3C268	-20.000,00 USD
Date	Time	Created By	Object GUID	30.000,00 USD

5 Possible Problems and Solutions

Using this technique avoids different postings being aggregated to one characteristic combination and thus being grouped together in one data record. This is also absolutely necessary for a complete log.

On the other hand, this of course results in a much larger data volume than normal. Therefore, possible performance problems must be taken into account with this approach.

If necessary, this effect can be reduced or even completely avoided with a suitable organizational task.

For example, it is imaginable that not the whole planning process is logged, but only certain steps. Recurring steps such as transferring old actual data as the basis for planning and changing it with automatic planning functions (revaluate: +10%) could be documented sufficiently, for example, in the process description of the planning cycle. During such “uncritical“ steps, the characteristic derivation can be removed from the characteristic relationships of the InfoCube. Only when, for example, a decentralized manual change of data is executed by the person responsible, can the described functionality be activated. Instead of changing the customizing of the characteristic relationships, you can program a mechanism to activate and deactivate the functionality described and integrate it in the method for characteristic derivation. Unfortunately, an automatic control is not possible here, since the method only receives the current data record, but not the call up context (query or name of a planning function).

Alternatively to that a second InfoCube can be used without line item display in which the plan data is transferred aggregated before “uncritical“ steps of the planning cycle.

6 Sample Coding

```
METHOD IF_RSPLS_CR_METHODS~DERIVE.
*---------------------------------------------------------------------*
*  <-- e_t_mesg         messages
*  <-> c_s_chas         characteristic combination: source and target
*                       fields included; do not change the source
*                       fields
*  <<- cx_rspls_failed  exception
*---------------------------------------------------------------------*

* begin example code:
* infrastructure needed by the buffer:
*  DATA: l_s_mesg   TYPE if_rspls_cr_types=>tn_s_mesg,
*        l_is_valid TYPE rs_bool.
*
*  FIELD-SYMBOLS: <l_th_buf> TYPE HASHED TABLE,
*                 <l_s_buf>  TYPE ANY.
* end example code:


  CLEAR e_t_mesg.


* begin of example code:
* use the buffer?
* o_use_buffer is switched on by default in the constructor
*  IF o_use_buffer = rs_c_true.
**   yes:
*    ASSIGN o_r_th_buf->* TO <l_th_buf>.
*    ASSIGN o_r_s_buf->*  TO <l_s_buf>.
*    <l_s_buf> = c_s_chas.
*    READ TABLE <l_th_buf> INTO <l_s_buf> FROM <l_s_buf>.
*    IF sy-subrc = 0.
*      IF o_r_is_valid->* = rs_c_true.
*        c_s_chas = <l_s_buf>.
*        RETURN.
*      ELSE.
*        IF e_t_mesg IS SUPPLIED.
*          APPEND o_r_s_mesg->* TO e_t_mesg.
*        ENDIF.
*        RAISE EXCEPTION TYPE cx_rspls_failed
*          EXPORTING
*            msgid = o_r_s_mesg->msgid
*            msgty = o_r_s_mesg->msgty
*            msgno = o_r_s_mesg->msgno
*            msgv1 = o_r_s_mesg->msgv1
*            msgv2 = o_r_s_mesg->msgv2
*            msgv3 = o_r_s_mesg->msgv3
*            msgv4 = o_r_s_mesg->msgv4.
*      ENDIF.
*    ENDIF.
*  ENDIF.
* implement your derivation algorithm here:



  FIELD-SYMBOLS: <l_chavl> TYPE ANY.

* fill ID
  ASSIGN COMPONENT 'BBP_OBGUID' OF STRUCTURE c_s_chas
                           TO <l_chavl>.
  CALL FUNCTION 'GUID_CREATE'
```

```
    IMPORTING
      ev_guid_32 = <l_chavl>.

* fill user
  ASSIGN COMPONENT 'USERNAME' OF STRUCTURE c_s_chas
                                TO <l_chavl>.
  <l_chavl> = sy-uname.

* fill date
  ASSIGN COMPONENT 'DATE0' OF STRUCTURE c_s_chas
                                TO <l_chavl>.
  <l_chavl> = sy-datlo.

* fill time
  ASSIGN COMPONENT 'TIME' OF STRUCTURE c_s_chas
                               TO <l_chavl>.
  get time field <l_chavl>.



* update the buffer with the result:
* l_s_mesg should contain a message in the 'invalid' case
* l_is_valid should indicate whether derivation was possible
* <l_s_buf> should contain the derived fields
*  IF o_use_buffer = rs_c_true.
*    o_r_is_valid->* = l_is_valid.
*    IF o_r_is_valid->* = rs_c_true.
*      INSERT <l_s_buf> INTO TABLE <l_th_buf>.
*      c_s_chas = <l_s_buf>.
*    ELSE.
*      IF e_t_mesg IS SUPPLIED.
*        o_r_s_mesg->* = l_s_mesg.
*        APPEND l_s_mesg TO e_t_mesg.
*      ENDIF.
*      INSERT <l_s_buf> INTO TABLE <l_th_buf>.
*      RAISE EXCEPTION TYPE cx_rspls_failed
*        EXPORTING
*          msgid = l_s_mesg-msgid
*          msgty = l_s_mesg-msgty
*          msgno = l_s_mesg-msgno
*          msgv1 = l_s_mesg-msgv1
*          msgv2 = l_s_mesg-msgv2
*          msgv3 = l_s_mesg-msgv3
*          msgv4 = l_s_mesg-msgv4.
*    ENDIF.
*  ENDIF.
* end of example code

ENDMETHOD.
```

www.sdn.sap.com/irj/sdn/howtoguides