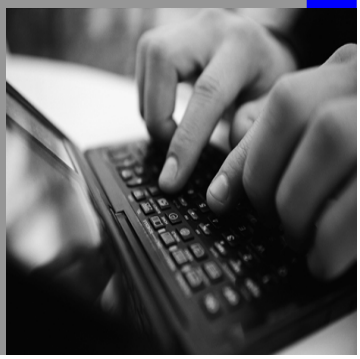


How-to Guide
SAP NetWeaver 2004s

How To... Create a Planning Function in BI Integrated Planning that calls BPS Exits

Version 1.00 – February 2006

Applicable Releases:
SAP NetWeaver 2004s

1 Scenario

In BW-BPS customers could develop their own planning functions by creating some function modules in which the business logic was implemented. This How to paper describes how in the BI Integrated Planning these function modules can be reused without using the BW-BPS itself.

2 Introduction

In BW-BPS there is one predefined planning function type for implementing customer defined business logic – the so-called planning functions type exit. Upon creation of such a planning function one had to specify the function module(s) containing the business logic and the interface of importing parameters – if necessary. There are two types of function modules used in a planning function type exit:

- The “exit” function module: this function module is MANDATORY and contains the business logic itself. Depending on the available data and on which of the characteristics are marked as “fields to be changed” this function module might be called several times. The available data is cut partitioned into smaller subsets or blocks and the function module is called with each of these blocks.
- The “Init” function module: this function module is OPTINAL. It is called once per execution before the exit module is called. It is used to do preliminary work. No transaction data can be changed in the init module.

In BI Integrated Planning (we will use BI IP as an abbreviation) this exit technique has been changed and made more powerful. Customers can now create their own planning function types that implement their business logic. Each planning function type requires a class where the business logic is implemented and defines its own parameter input interface. There is a variety of types that can be used as input parameters: values of InfoObjects, structures, and even tables. Also conditions can be used.

This paper describes how the predefined and delivered class CL_RSPLFC_BPS_EXITS can be used to reuse the business logic implemented as a planning function type exit in BW-BPS in the BI IP.

We want to stress the point here that the exits in BI IP are much more powerful than the exits in BPS. For example there are ‘Start’ and ‘End’ methods that are called at the beginning and at the end of the execution of the planning function. Reference data can be provided by the system automatically and no programming is necessary to read additional reference data. Also the building of blocks (alias subsets) can be switched off if it is not necessary. Exits in BI IP thus offer higher flexibility and better performance than the ones in BPS. Therefore it should always be considered whether it makes more sense to reuse a BPS exits as described in this paper or to re-implement it in the BI IP environment.

3 Functionality Implemented in the Class

The class CL_RSPLFC_BPS_EXITS is delivered as an implementation of a planning function type in BI Integrated Planning. It allows you to reuse function modules of type

Exit in BI Integrated Planning that were written for planning functions in BW-BPS or SEM-BPS.

The wrapper class receives the transaction data and information about the selection from the BI IP, converts these tables to the tables used in the interfaces of the BPS function modules and passes them on to the specified function modules. After the data has been changed in the BPS exit function modules the tables are converted back to the BI IP format. Note that the old BPS planning functions themselves are not called, only the underlying function modules. This means that BPS objects (for example the BPS data buffer) are not used. Due to the specific requirements of the interfaces and the fact that BPS objects are not used, there are certain limitations which have to be taken into account for the planning function types that are realized. Here are some examples for situations that can cause problems or unexpected results:

- The BPS exit needs the information from which planning level or with which planning package it is called.
- The BPS exit reads reference data from the BPS buffer via the BPS API or other interfaces.
- The BPS exit reads values of BPS variables.
- The BPS exit calls function modules of the BPS API such as “post” or “refresh”.

Note that filter selections in BI Integrated Planning can be formulated more generally (with exclusions, for example). The selection is passed on to the function module in this form. If coding is written for this selection in the function module, the coding has to be modified as appropriate.

Also consider the new software paradigm used in the BI Integrated Planning. No GUI connections are used and thus sending popups from an exit function module is not possible anymore.

The class assumes that the names of the function modules are transferred in two elementary parameters with fixed definitions. The names of the parameters are defined in constant P_C_NAME_EXIT_PARAM and in P_C_NAME_INIT_PARAM.

There is another fixed optional parameter for filling the importing parameter I_AREA of the function module(s) - in many function modules, the name of the area is used from which the module is called. The area is not used in BI Integrated Planning but it may be the case that an area exists in BPS which only uses the current InfoCube. For this reason, you can create a parameter with the name specified in constant P_C_NAME_AREA_PARAM and later fill it with the name of this area. The value of this parameter is passed on to the function module in the interface as the value of the importing parameter I_AREA. Note that this technology can generally only be used for Basis InfoCubes and simple planning areas (not multi- planning areas or MultiProviders).

The class recognizes additional parameters generically - therefore this class can be used as the implementing class for various planning function types: in addition to the predefined parameters the planning function types can define additional elementary parameters; the class recognizes these additional elementary parameters automatically and passes them on to the exit parameter table IT_EXITP of the function module.

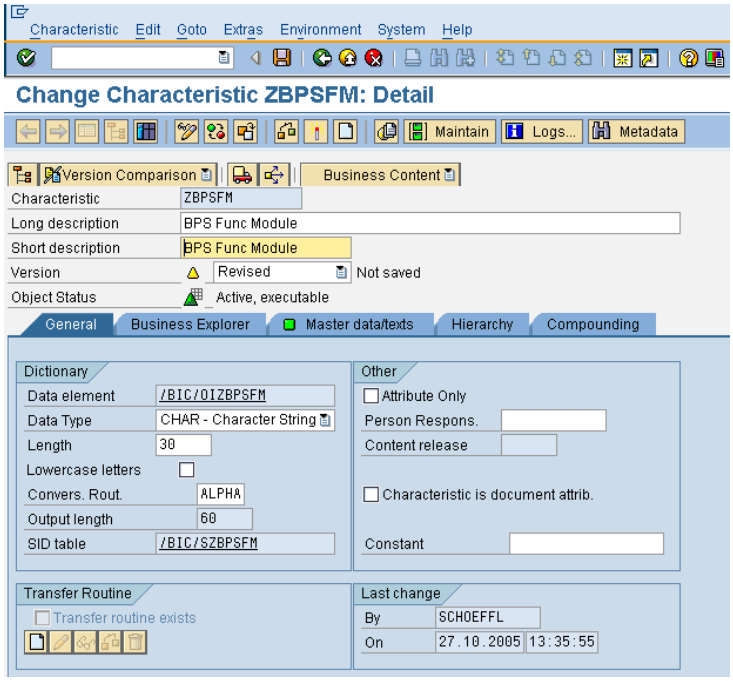
The class is realized in such a way that conditions can also be used - you can even specify a different set of the function modules for each condition. The function module for the exit always has to be filled; the function module for the init module is optional.

4 The Step By Step Solution

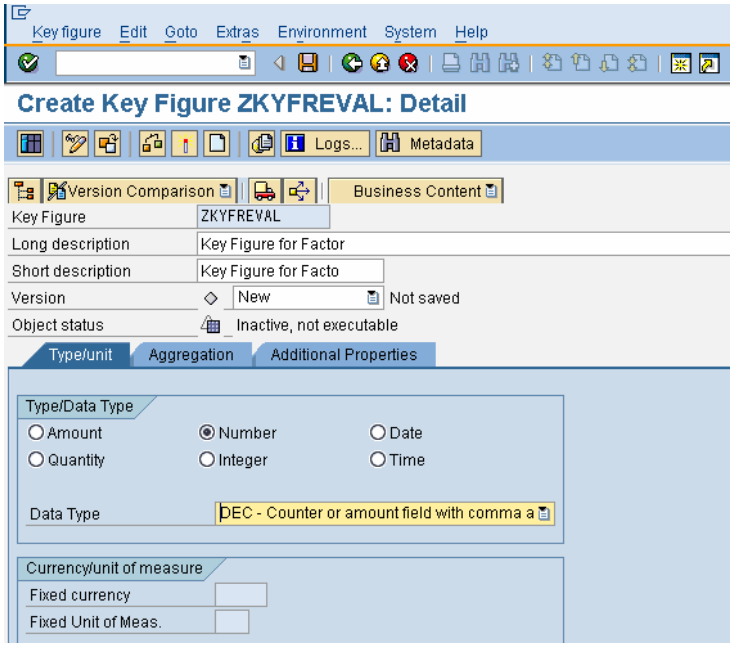
In this chapter we will provide a step by step example how you can use the delivered class to create a planning function type that calls BPS function modules. The planning function type will have two parameters for the function modules. We assume that we have a function that needs some value as a revaluation factor and create an additional parameter for this factor.

4.1 Create the InfoObjects for the Parameters

1. Create an InfoObject that accepts the name of the function module as a characteristic value. We recommend that you create a characteristic of type character with length 30 and indicate that this InfoObject is "Without Master Data".

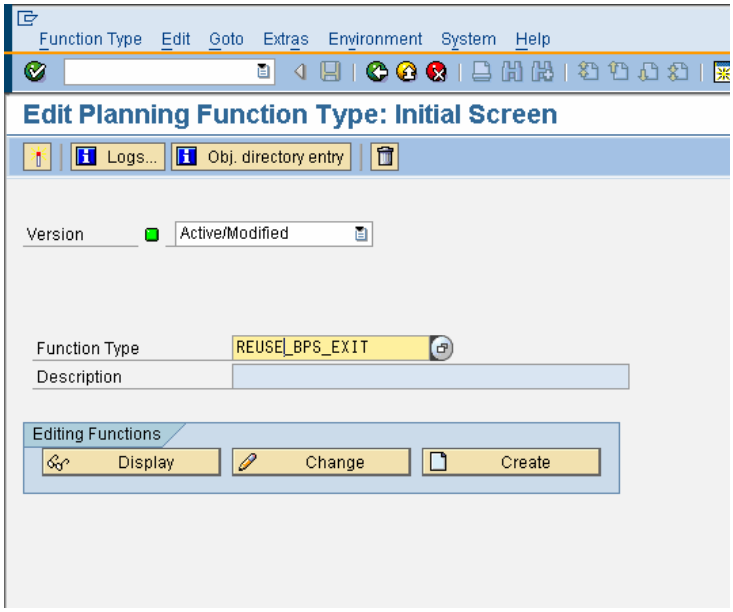


2. Create a key figure that can be used for the definition of the parameter for the revaluation factor.

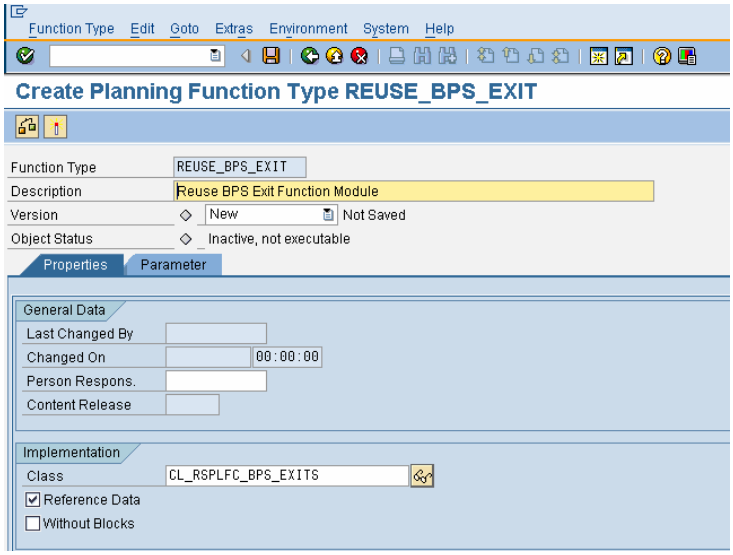


4.2 Create the Planning Function Type

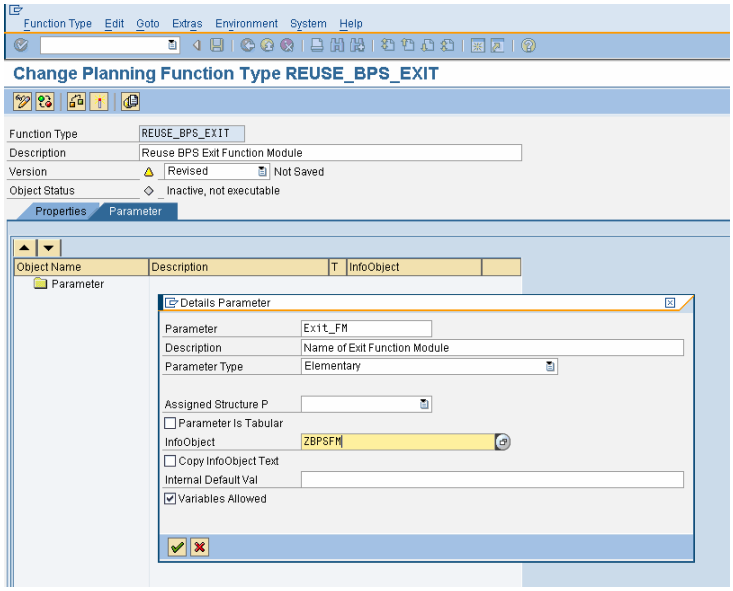
3. In transaction RSPLF1, create a new function type.



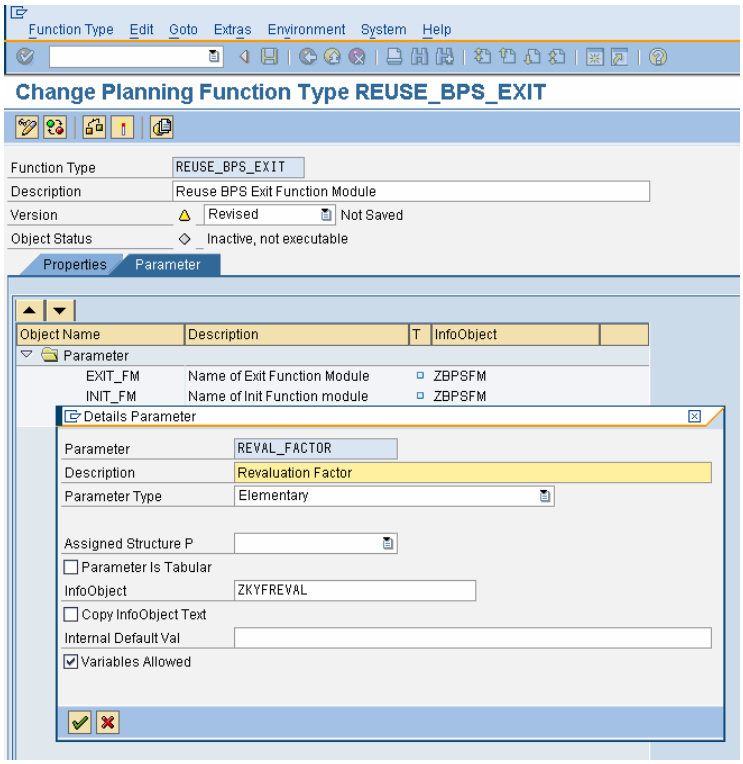
4. Choose the "Reference Data" option and enter the name of the class. This does NOT imply that reference data is read automatically, but results from the requirements for implementing the class. Enter CL_RSPLFC_BPS_EXITS as the name of the implementing class.



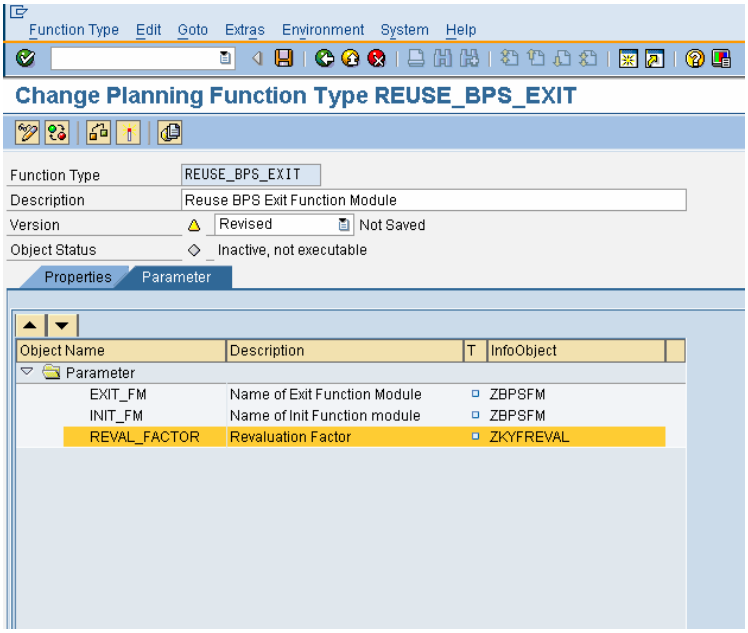
5. On the "Parameter" tab page, create two parameters - one for the exit module and one for the init module. The parameters should be "elementary". Chose the InfoObject you created in step one as the dedicated InfoObject. The names of the parameters are determined from the values of constant P_C_NAME_EXIT_PARAM and P_C_NAME_INIT_PARAM.



6. Create the additional parameter for the revaluation factor.

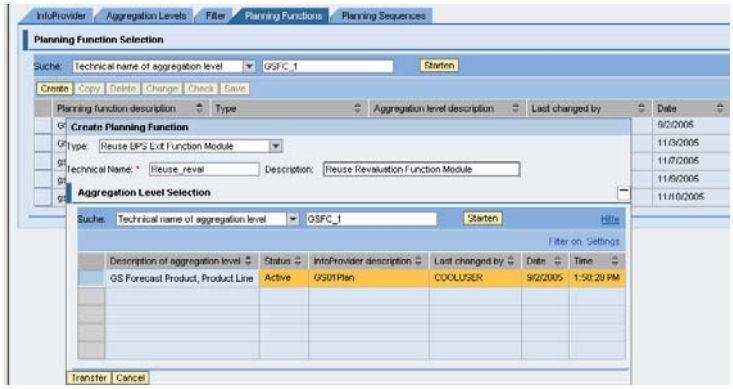


7. Once you are done you have to activate the planning function type. The type can now be used and you can create a planning function for that type.

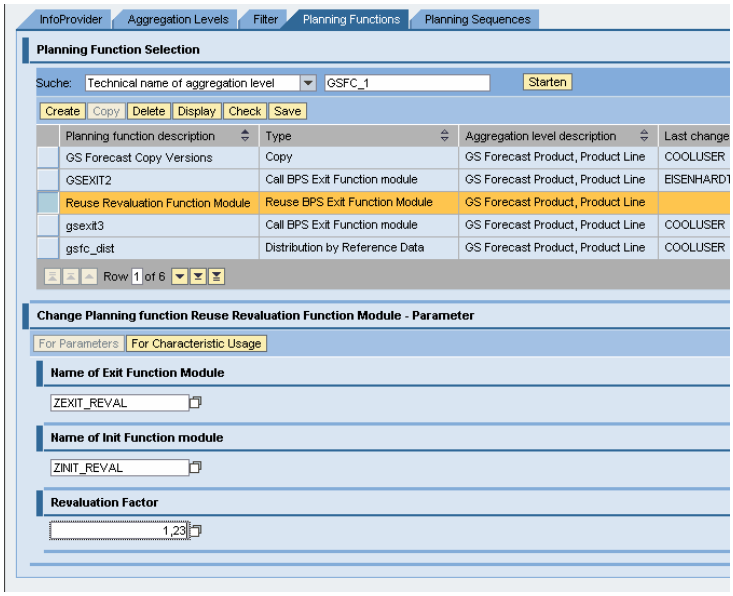


4.3 Create the Planning Function

8. Use the planning modeler to create a new planning function using the type we just defined. Choose the type of planning function and enter a name.



9. You can now enter the names of the function modules and the revaluation factor. You can test the new planning function now in a sequence in the planning modeler.



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