Reuse of BW-BPS Planning Functions in BI-Integrated Planning

Applies to

SAP BW-BPS projects getting upgrade to SAP BI-IP. Developers who wish to incorporate business logic through exit planning functions type. For more information, visit the Business Intelligence homepage.

Summary

This document gives a step by step process to build an Exit planning function type in SAP Integrated Planning.

Author: Pankaj Patil

Company: Accenture

Created on: 21st April 2010

Author Bio

Pankaj Patil is a SAP BI-IP consultant at Accenture Services Private Limited, India. He has an elaborate experience of 3 years in ABAP customizations done within SAP BI-IP implementations.

Table of Contents

Applicable scenario

Exit functions can be used to perform specific planning activities that otherwise is not possible to implement through standard SAP delivered objects.

In BW-BPS, there are planning functions developed, which have customized logic written through function modules. This paper will describe the reuse of same function modules in BI-IP without any interference of BW-BPS.

Prerequisites

Fundamentals of SAP Business Intelligence, SAP Integrated Planning. Good knowledge of ABAP language. Customizations build in function modules are completely coded in ABAP.

Introduction

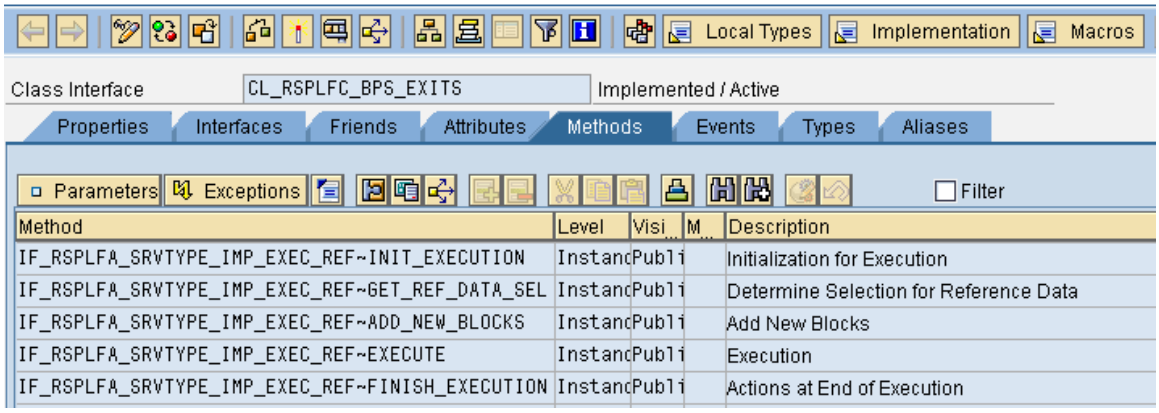
In BW-BPS, we can define exit planning functions through a predefined planning function type. While creating this kind of planning function, the developer has to specify 2 types of function modules. These 2 function modules then become the part of planning function. These are conventionally termed as: Init and Exit function modules.

However BI-IP does not provide the option of defining exit planning function directly on the planning modeler. In order that this planning functions exit type appear in planning function tab, we have to make use of a predefined SAP delivered wrapper class CL_RSPLFC_BPS_EXITS.

Wrapper Class "CL_RSPLFC_BPS_EXITS"

The class CL_RSPLFC_BPS_EXITS is delivered as an implementation of a planning function type in BI Integrated Planning. Implementation of this class gives an added advantage in BI-IP. Placeholders in form of 'INIT_EXECUTION' and 'FINISH_EXECUTION' methods give developer an option to do flexible logic building. Also no extra programming is necessary to read the reference data. Therefore, an additional thinking can always be done to either reuse a BW-BPS planning function as would be described below or rebuild the same logic in BI-IP all again.

Class Builder: Display Class CL_RSPLFC_BPS_EXITS



Method	Level	Visi...	M...	Description
IF_RSPLFA_SRVTYPE_IMP_EXEC_REF~INIT_EXECUTION	Instanc	Publi		Initialization for Execution
IF_RSPLFA_SRVTYPE_IMP_EXEC_REF~GET_REF_DATA_SEL	Instanc	Publi		Determine Selection for Reference Data
IF_RSPLFA_SRVTYPE_IMP_EXEC_REF~ADD_NEW_BLOCKS	Instanc	Publi		Add New Blocks
IF_RSPLFA_SRVTYPE_IMP_EXEC_REF~EXECUTE	Instanc	Publi		Execution
IF_RSPLFA_SRVTYPE_IMP_EXEC_REF~FINISH_EXECUTION	Instanc	Publi		Actions at End of Execution

Some important attributes of the class are:
P_T_CHARSEL,
P_T_CHARNM,
P_T_KEYFNM,
P_C_NAME_EXIT_PARAM,
P_C_NAME_INIT_PARAM,
P_C_NAME_AREA_PARAM

The class on recognizing these parameters passes them to the above mentioned 2 function modules.

The class forms an interface and fetches the data from backend. An important point to note is that the fetched data is converted into tables that are used in the interfaces of the BPS function modules. After the data is changed in function modules, the tables are converted back to BI-IP format. Therefore BPS planning functions are not utilized; rather the underlying function modules are called.

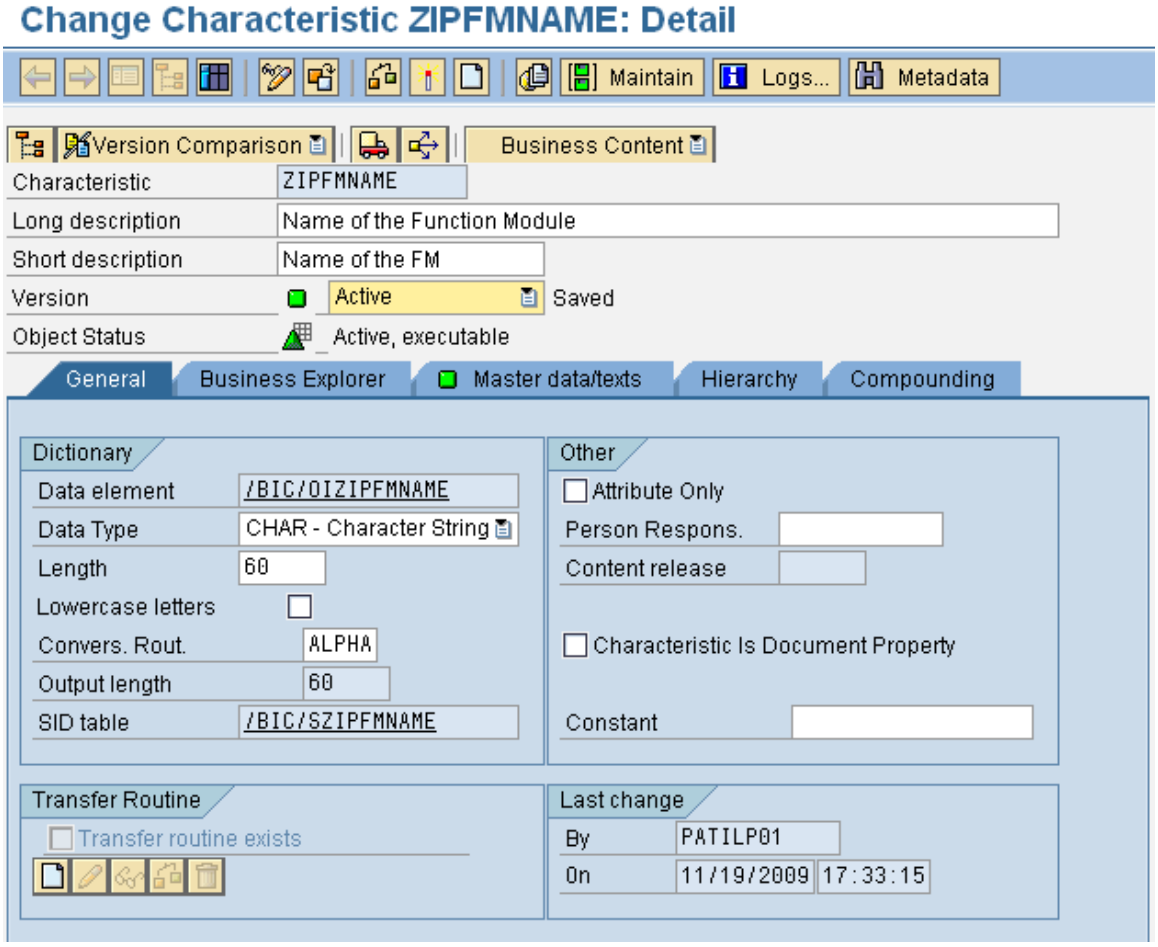
The fact that BPS data buffer is not used though may lead to certain limitations in reusing the logic build in function modules. Some examples of possible issues that can arise are:

1) BPS exits needing information from BPS variables.
2) Information is required to be fetched from planning level or planning package.
3) Reference data required to read through BPS buffer.

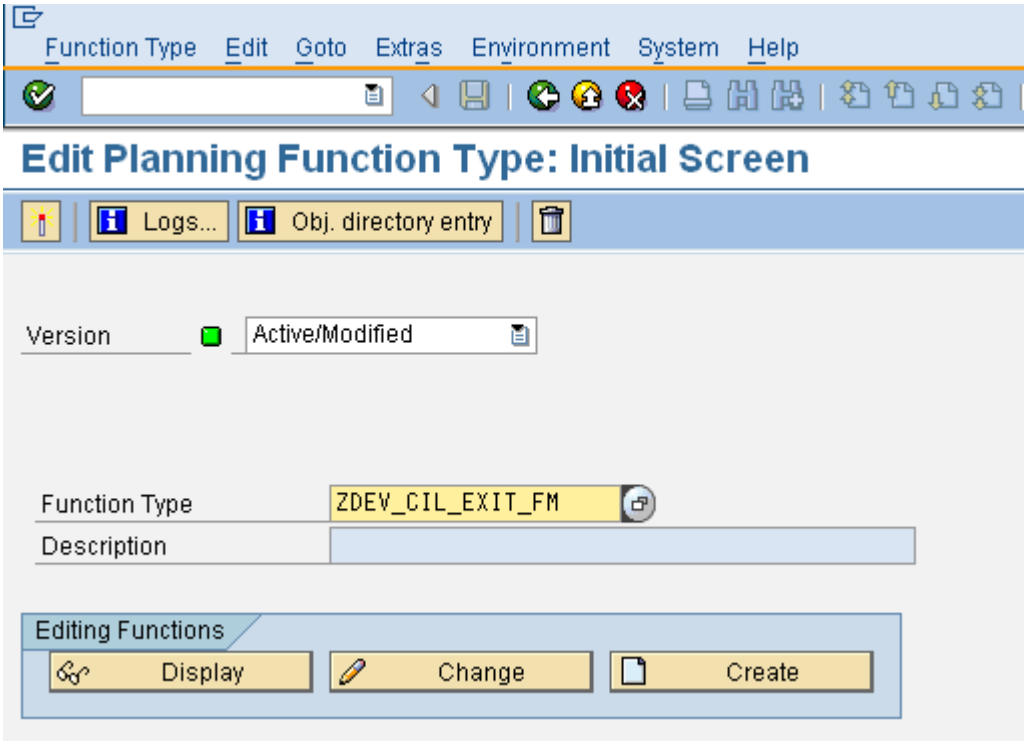
Step By Step procedure

To start with, Exit planning function type will have 2 parameters that will take in entries for 2 function modules. In order that these parameters appear as part of planning function, an infoobject will require to be created. This infoobject will act as parameter cells that will hold the name of 2 function modules in exit planning function.

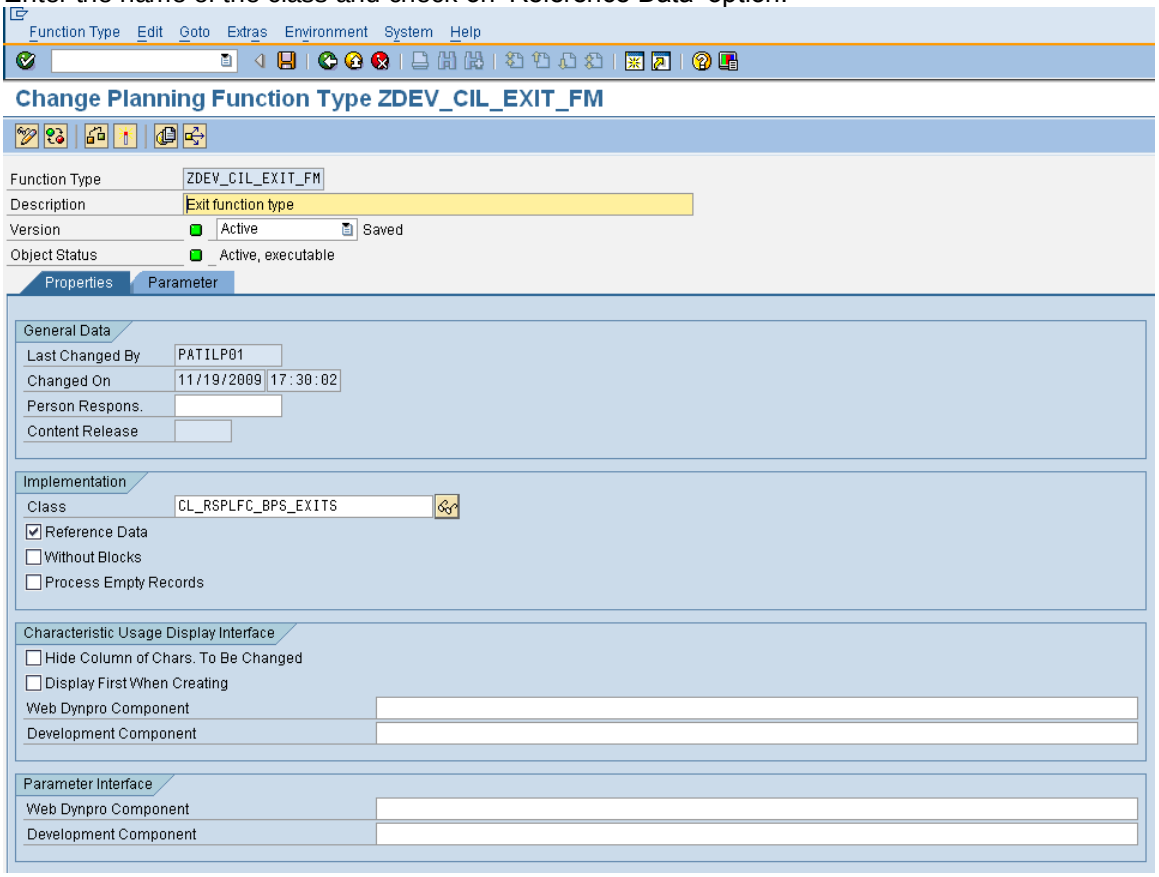
Infoobject of data type character string (CHAR) and length 60 is created as in the below snapshot.



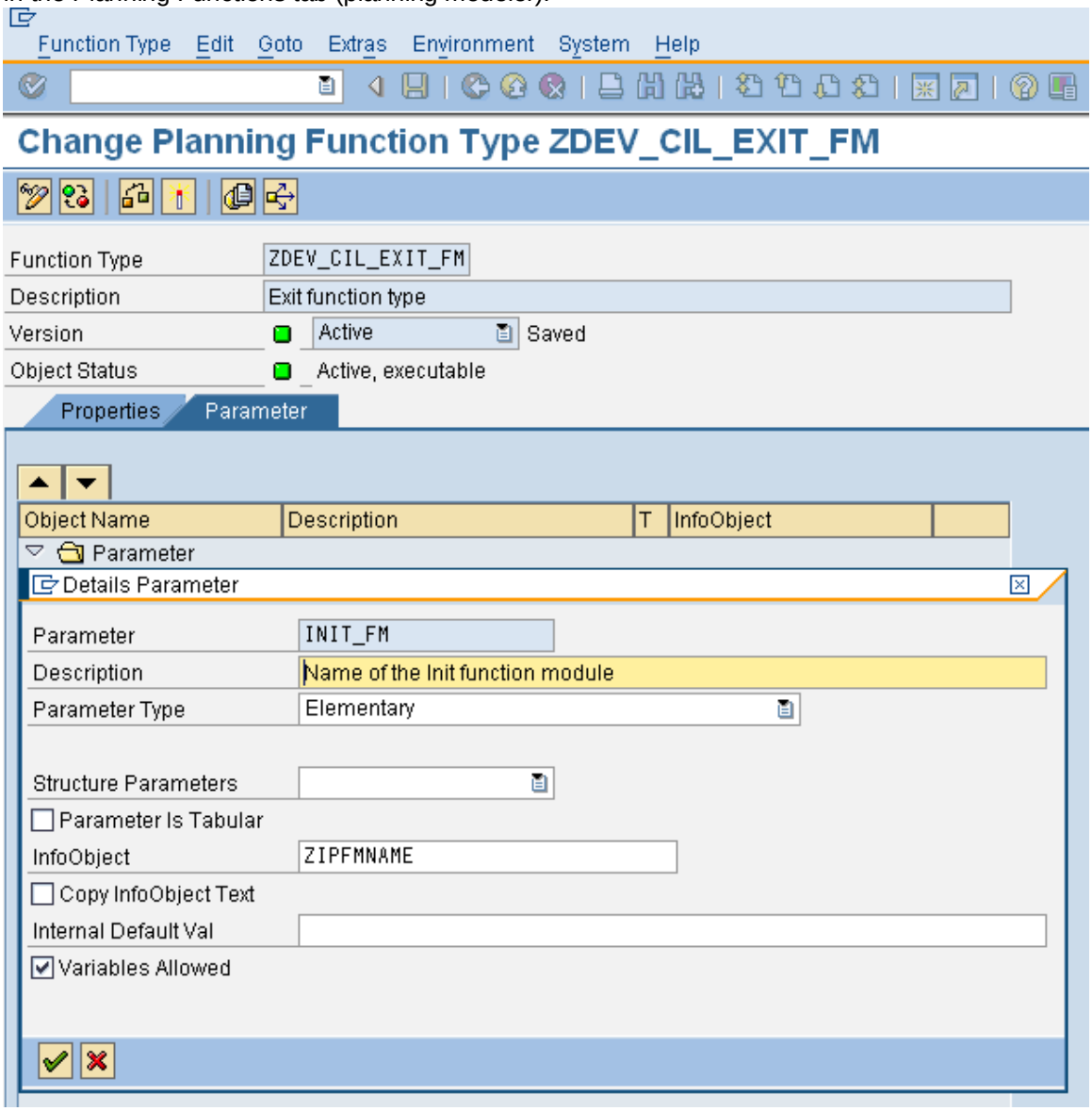
Now create a planning function type in transaction RSPLF1 that can utilize the functionality of the wrapper class.



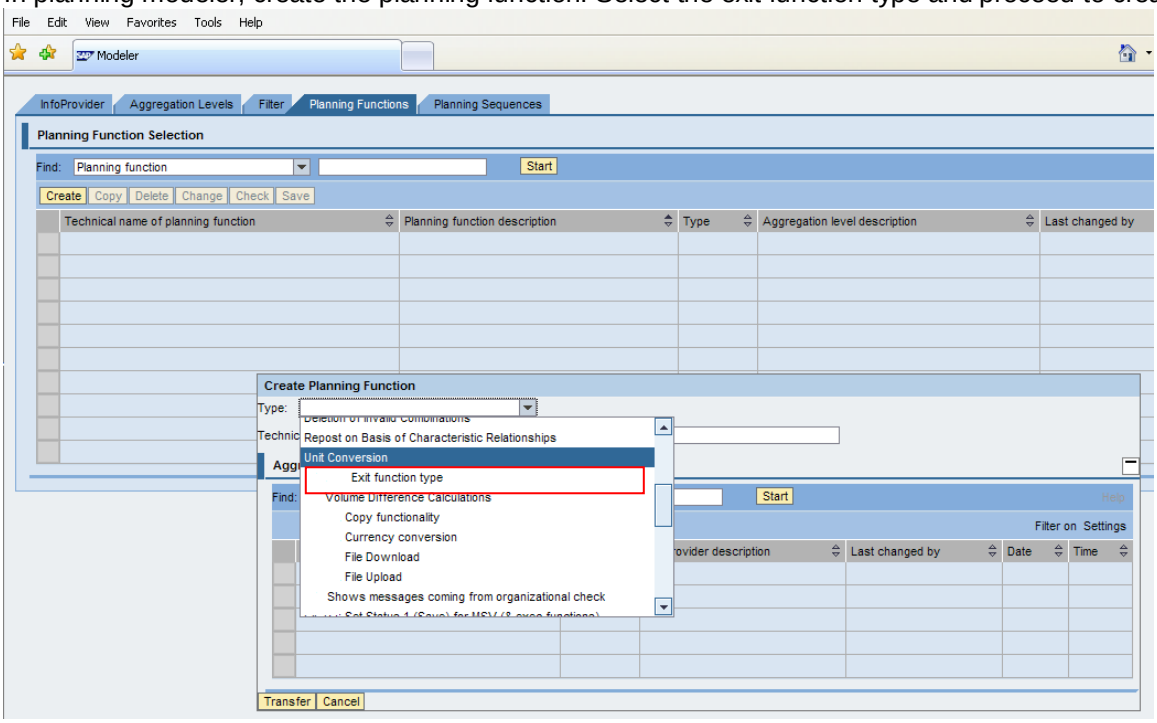
Enter the name of the class and check on 'Reference Data' option.



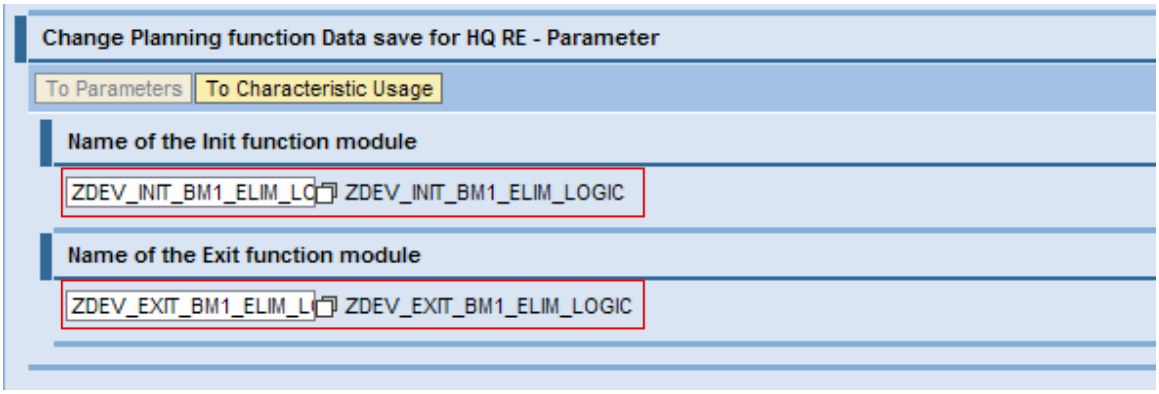
Now click on the 'parameter' tab. Create 2 parameters – 1 each for the exit and init function modules. Choose the parameter type to be 'Elementary' and enter the name of above created infoobject. After the parameters are created, activate the planning function type. The new planning function type will now appear in the Planning Functions tab (planning modeler).



In planning modeler, create the planning function. Select the exit function type and proceed to create one.



Names of 2 Function modules (Exit and Init) can be put in the newly created planning function. Note that these function modules can be same modules used in BW-BPS.



Function Modules

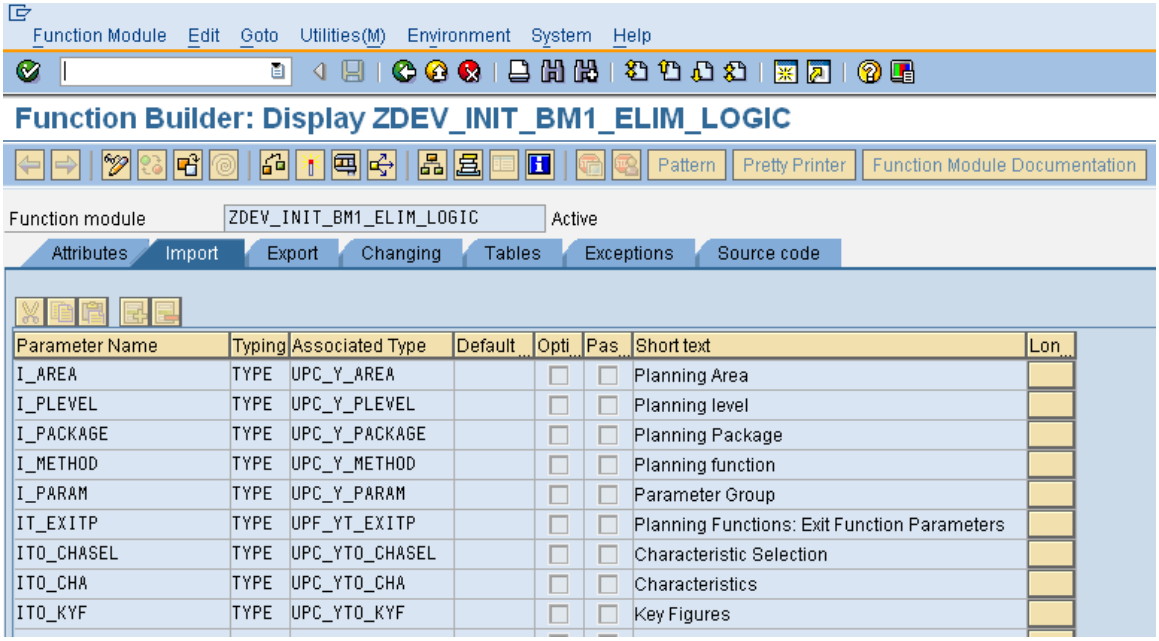
Init module

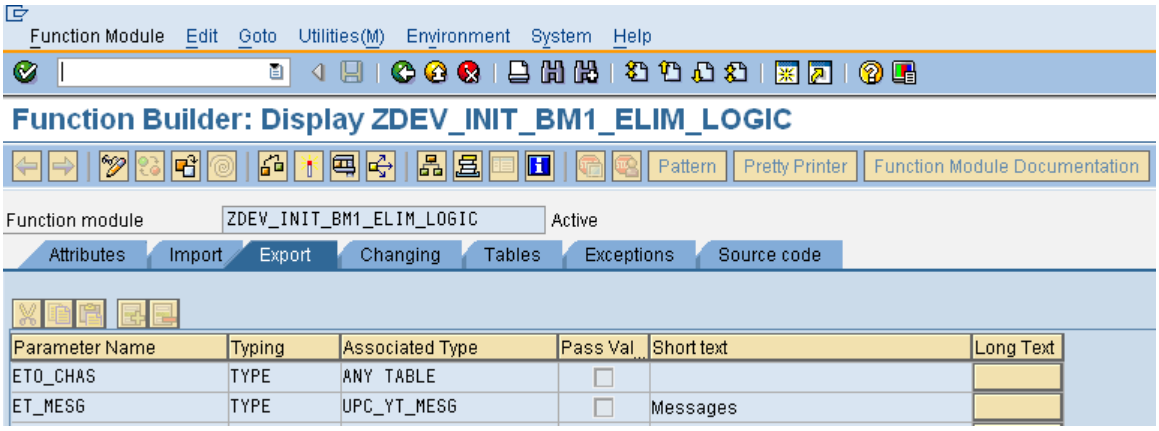
Used for initialization. Particularly useful when new data records are to be created. It is optional and called only once per execution before the exit module is called.

Interface of module (Import/Export parameters):

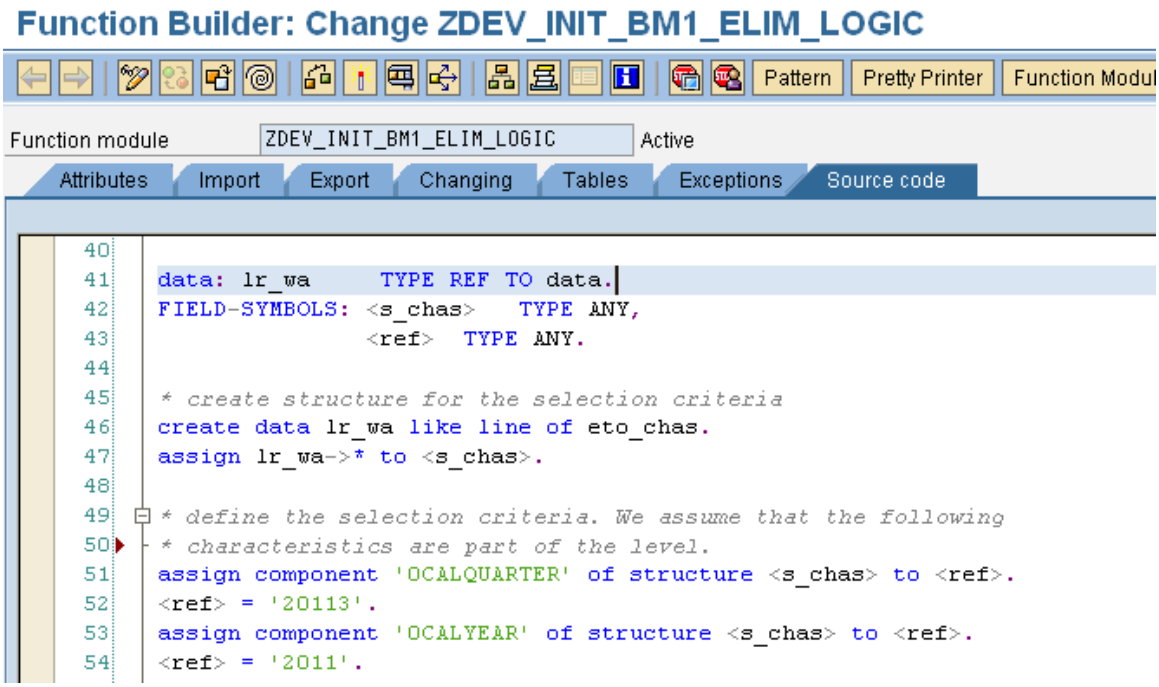
In the import tab, the parameter I_AREA will contain the name of data provider. Generally, infocube name and simple aggregation levels can be included.

IT_EXITP is another such parameter table that can contain any additional elementary parameter defined in the planning function type. Note these parameters are apart from the once that are defined for INIT and EXIT modules.



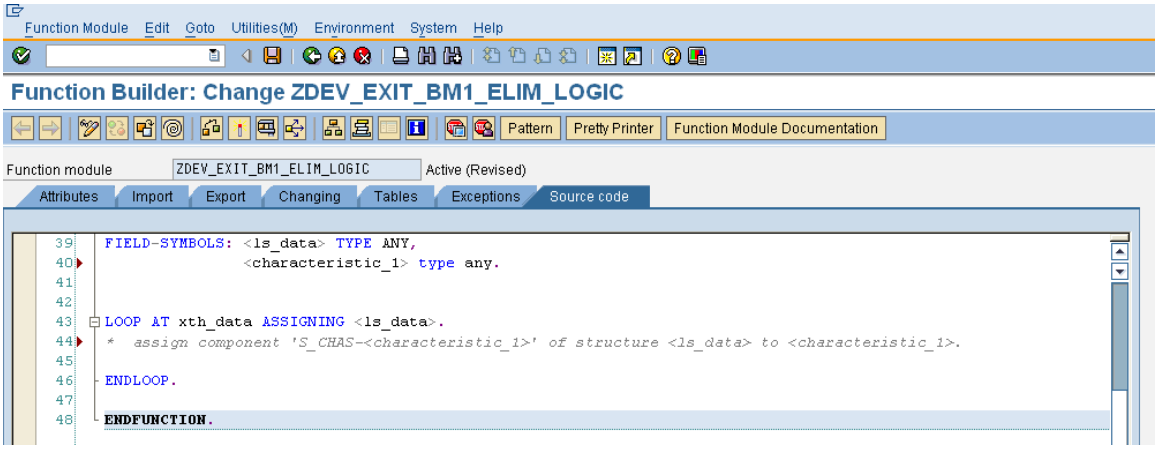


Sample coding to restrict the data with specific values:



Exit module

It contains the custom-defined logic and is mandatory to be defined. It can be called multiple times as the data to change can be distributed in several smaller subsets of partitions depending on the characteristics that are marked under 'changed' section.
A difference in interface parameters between INIT and EXIT is the table XTH_DATA.
Exit module contains XTH_DATA. The table buffer can be efficiently used to code the business specific logic.



For references and more details on how to code the 2 function modules, please refer SAP delivered function group UPFX. It has an overview of options that are possible with exit functions. The sample function modules mentioned in the function group can be used to code custom-defined logic. Below link gives more details about Exit functions.

http://help.sap.com/saphelp_nw70ehp1/helpdata/en/81/b54aa832e911d4b2c10050dadfb23f/frameset.htm

Additionally the exit functions can be combined with different planning functions to achieve a computationally larger functionality. These multiple planning functions can be then sequenced in planning sequences.

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.