ABAP Part I

Lesson 12: Modularization Techniques

Function Modules

Lesson Objectives

After completing this lesson, participants will be able to -

- Function Modules

Modularization

ABAP contains the following kinds of procedures:

- Subroutines
  - 1.Subroutines are principally for local modularization, that is, they are generally called from the program 2.in which they are defined. You can use subroutines to write functions that are used repeatedly within a program. You can define subroutines in any ABAP program.
- Function Modules
  - 1.Function modules are for global modularization, that is, they are always called from a different program.
  - 2.Function modules contain functions that are used in the same form by many different programs. They are important in the R/3 System for encapsulating processing logic and making it reusable.
  - 3.Function modules must be defined in a function group, and can be called from any program.

Function Modules : Overview

Function modules are

- General-purpose ABAP/4 routines that anyone can use
- Are Reusable
- Are stored in Function Groups
- Have special screen used for defining parameters-parameters
- SE37 is used to build function Modules

Function Group

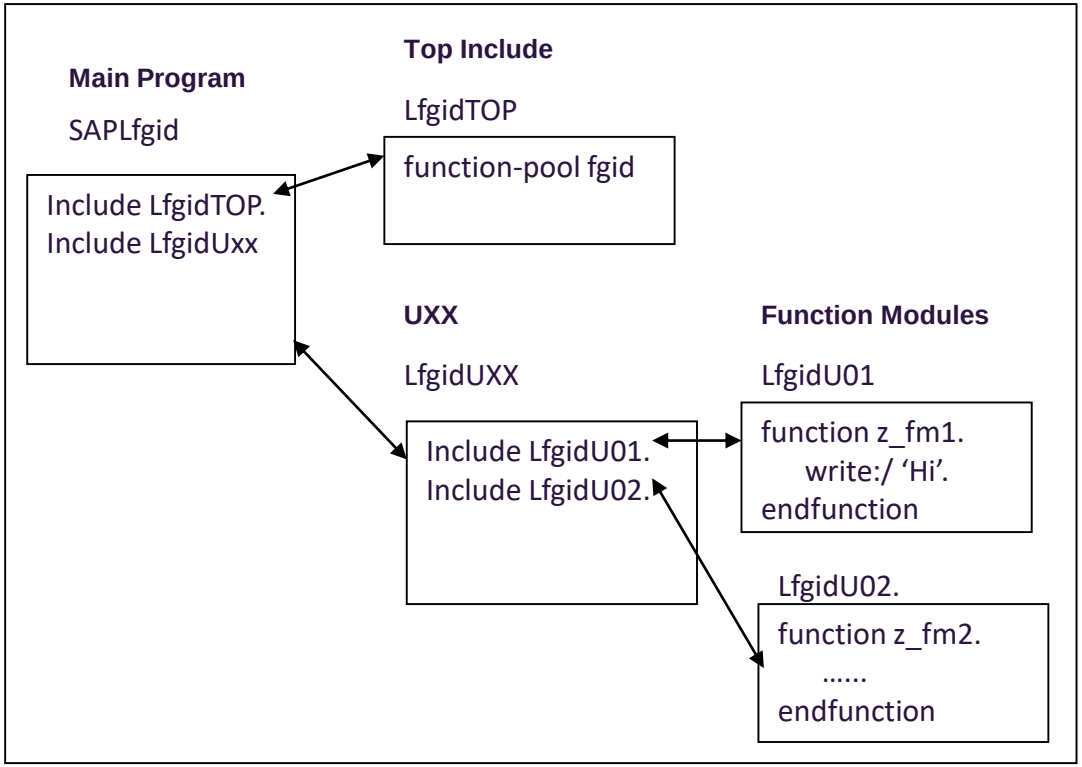
A function group is a collection of logically related functions that share a common program context, such as global variables, at runtime.

Every Function belongs to a function group.

Function Group

When you create a function module, the system will first ask you for a function group ID.
This ID tells the system where to store the function module.
When you supply the ID, and if it doesn't yet exist, the system creates:

- A main program
- A top include
- A UXX include
- A function module include

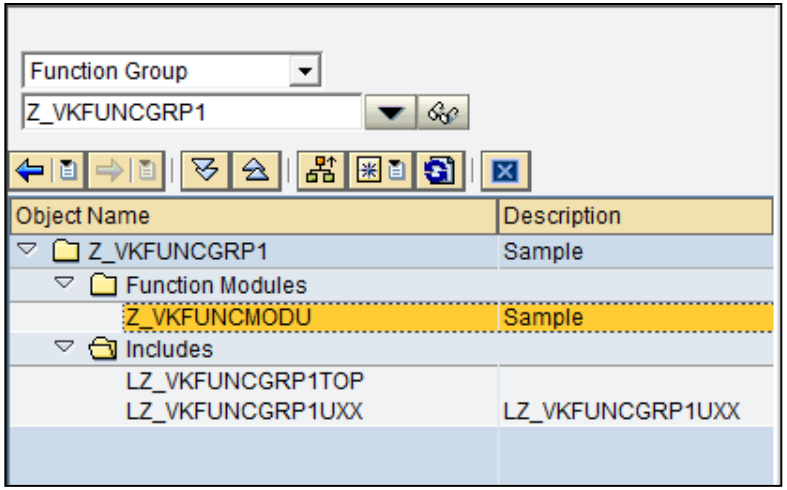


Function Group

The name of the main program will be saplfgid, where fgid is your four-character function group ID.

The system automatically places two include statements into it:

- include lfgidtop.
- include lfgiduxx.



Defining the Function Module Interface

To define parameters, you must go to one of two parameter definition screens:

- Import/Export Parameter Interface
- Table Parameters/Exception Interface

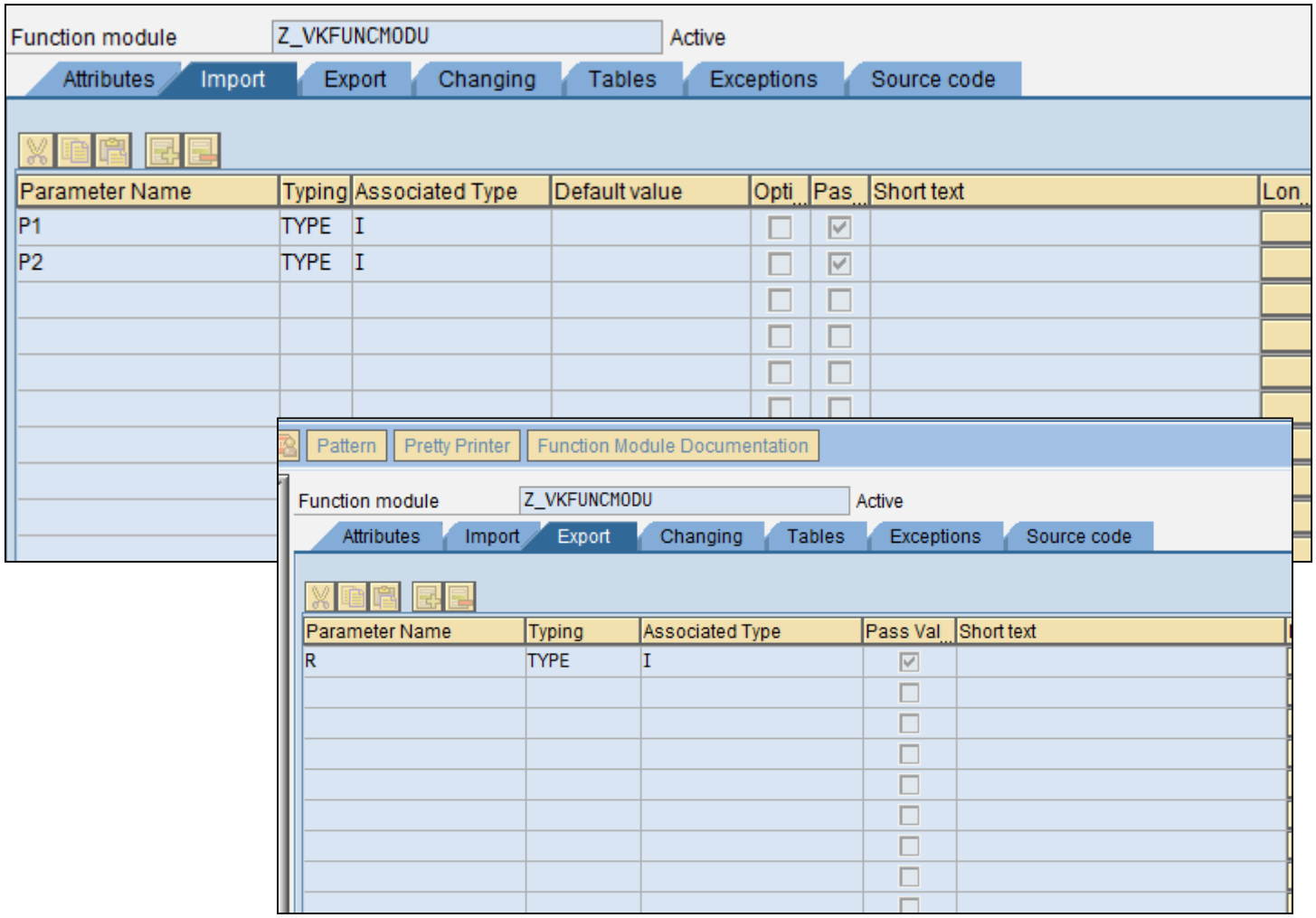
Import

- Values transferred from the calling program to the function module.
- The corresponding formal parameters in the function module are defined under IMPORTING
- You must specify values to any function module import parameters with no default assigned in the interface definition.
- You cannot overwrite the contents of import parameters at runtime

Export

- Values transferred from the function module back to the calling program.
- The assignment of actual parameters to export parameters is up to the user.

Import/Export Parameters



Passing Parameters to a Function Module

The methods for passing parameters to function modules are very similar to those for passing parameters to external subroutines.

By default:

- Import and export parameters are passed by value.
- Changing parameters are passed by value and result
- Internal tables are passed by reference

Calling Function Modules

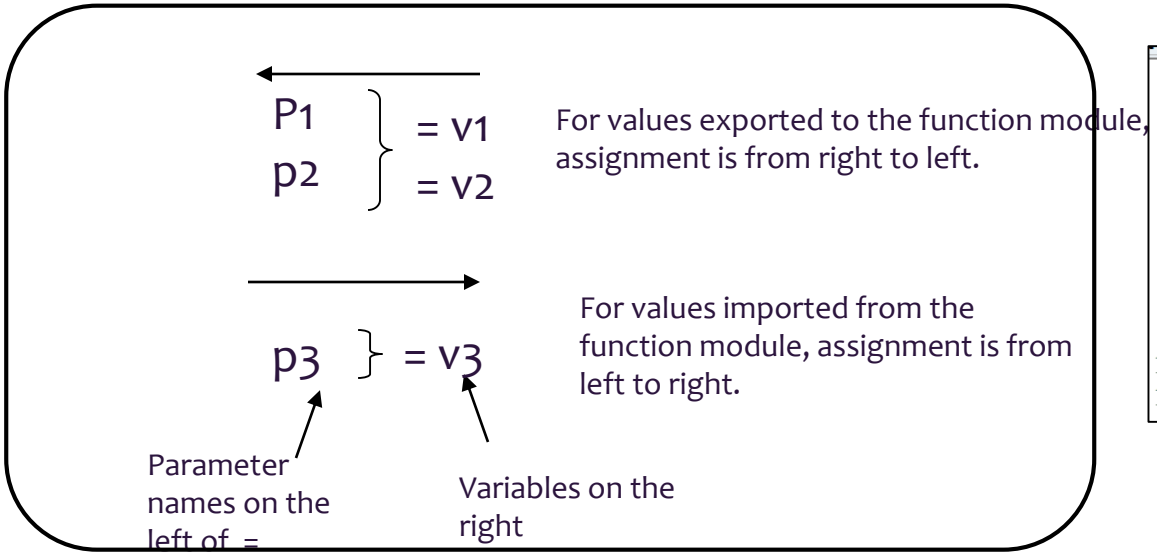
```
call function 'FUNC'
[exporting   p1 = v1 ... ]
 [importing   p2 = v2 ... ]
 [changing    p3 = v3 ... ]
 [tables      p4 = it ... ]
 [exceptions  x1 = n [others = n]].
```

Parameters passed to and received from a function module

data: v1, v2, ,v3.

call function ‘Z_XXX’

exporting




```
1   REPORT Z.
2   PARAMETERS: NUM1 TYPE I,
3               NUM2 TYPE I.
4   DATA RES TYPE I.
5   CALL FUNCTION 'Z104329FMADD'
6     EXPORTING
7       N1              = NUM1
8       N2              = NUM2
9    IMPORTING
10     R               =  RES.     .
11  WRITE RES.
```

Defining Global Data for Function Modules

Two types of *global data* can be defined for a function module:

- Data definitions placed in the top include are accessible from all function modules and subroutines within the group.
  - This type of global data is persistent across function module calls.
- Data definitions within the interface are known only within the function module.
  - If the interface is defined as a *global interface*, the parameter definitions are also known within all subroutines that the function module calls.
  - This type of global data is not persistent across function module calls.

Defining Subroutines in a Function Group

You can call internal and external subroutines from within function modules.

Setting the Value of sy-subrc on Return

Normally, after returning from a function module, the system automatically sets the value of value of *sy-subrc* to zero.

Use one of the following two statements to set *sy-subrc* to a non-zero value:

- raise
- message ... raising

Using the message ... raising Statement

The *message ... raising* statement has two modes of operation:

- If the exception named after raising is not handled by the call function statement and others is not coded, a message is issued to the user.
- If the exception named after raising is handled by the *call function* statement, a message is not issued to the user.
- Instead, control returns to the call function statement and the exception is handled the same way as for the raise statement.

Summary

In this lesson, you have learnt:

- Function Modules

Review Question

Question 1. The _____ statement terminates the subroutine and goes directly to the end-of-selection event.

Question 2: Parameter names that appear on the form statement are called _____ parameters.

Review Question

Question 1. Function modules are organized into _____.

Question 2: The _____ statement is used to exit the function module and set the value of sy-subrc on return.

Question 3: *Tables* parameters are always passed by _____.