

## **Objective**

**The following section is intended to explain:**

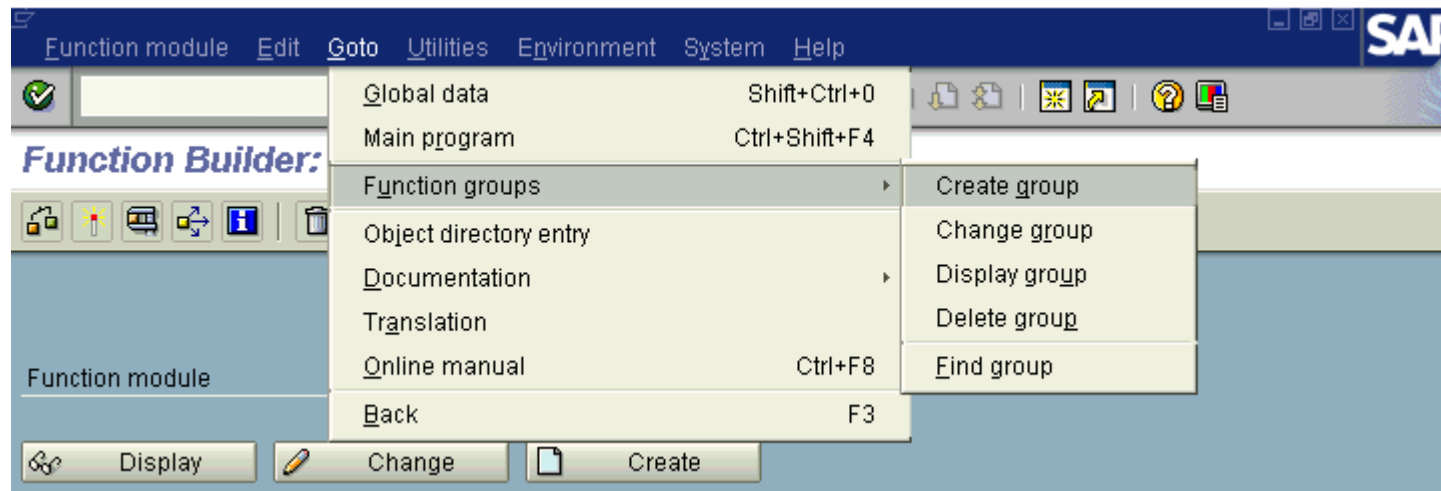
- **What function modules are**
- **Components of function modules**
- **Testing and releasing of function modules**

- **Function modules are special external subroutines(program type F)**
- **Function modules are classified in function groups and stored in the Function Library. Function groups act as containers for function modules that logically belong together.**
- **Function modules allow us to encapsulate and reuse global functions in the R/3 System.**
- **Function modules also play an important role in database updates and in remote communications between R/3 Systems or between an R/3 System and a non-SAP system.**
- **The R/3 System provides numerous predefined function modules that we can call from your ABAP/4 programs. We can also create your own function modules using Function Builder. (Transaction Code SE 37)**

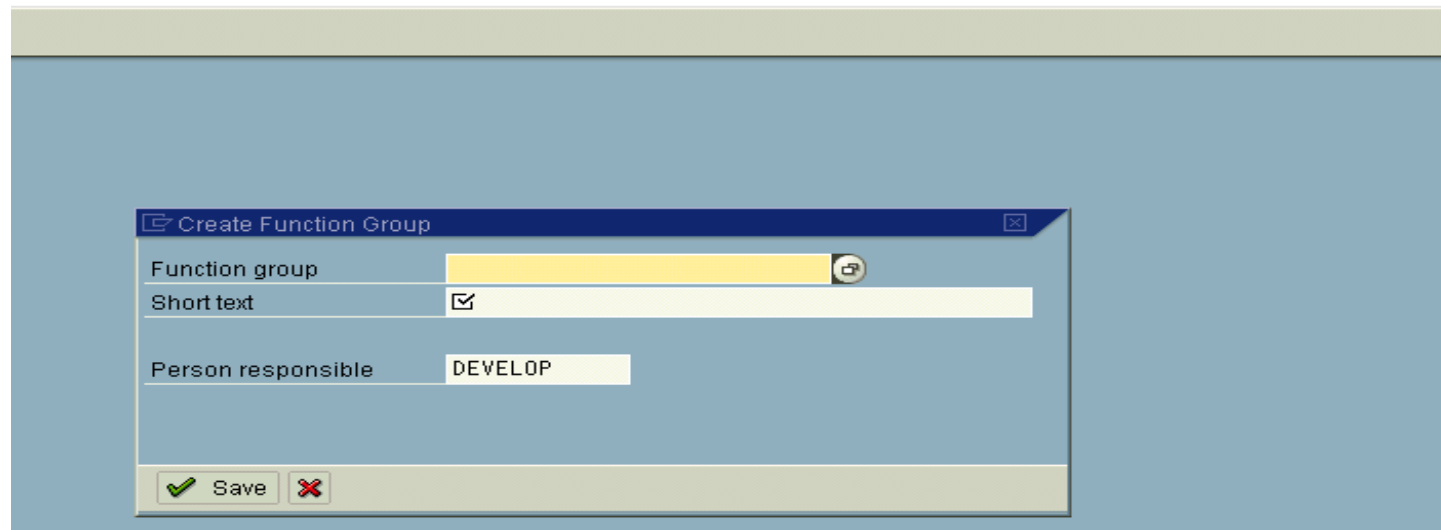
- **Subroutines are principally for local modularization while Function modules are for global modularization, that is, they are always called from a different program.**
- **Subroutines are defined in ABAP programs while Function modules are defined within function groups**
- **Function modules have clearly defined data interfaces to the calling program.**
- **We can test function modules in a stand-alone mode independent of the calling program.**

- **Function groups are containers for function modules.**
- **We cannot execute a function group. When we call a function module, the system loads the whole of its function group into the internal session of the calling program (if it has not already been loaded).**
- **Function group names are freely definable up to a maximum length of 26 alphanumeric characters.**
- **When we create a function group or function module in the Function Builder , the main program and include programs are generated automatically.**

## New function groups can be created from the menu Goto - Function Group - Create



SAP R/3

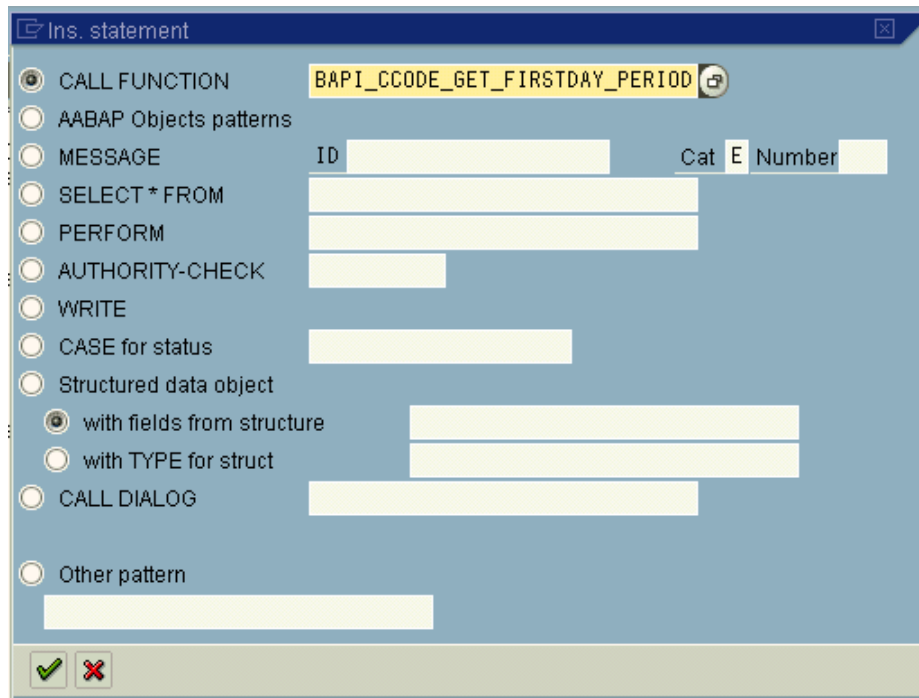


To program a function module, we must include our statements between the **FUNCTION** and **ENDFUNCTION** statements as follows:

**Syntax : FUNCTION <module>**  
**<statements>**  
**ENDFUNCTION**

---

**An existing Function Module can be inserted into ABAP Code using Edit - Pattern**



**Import:** Values transferred from the calling program to the function module. You cannot overwrite the contents of import parameters at runtime.

**Export:** Values transferred from the function module back to the calling program.

**Changing :** Values that act as import and export parameters simultaneously. The original value of a changing parameter is transferred from the calling program to the function module. The function module can alter the initial value and send it back to the calling program.

**Tables:** Internal tables that can be imported and exported. The internal table's contents are transferred from the calling program to the function module. The function module can alter the contents of the internal table and then send it back to the calling program. Tables are always passed by reference.

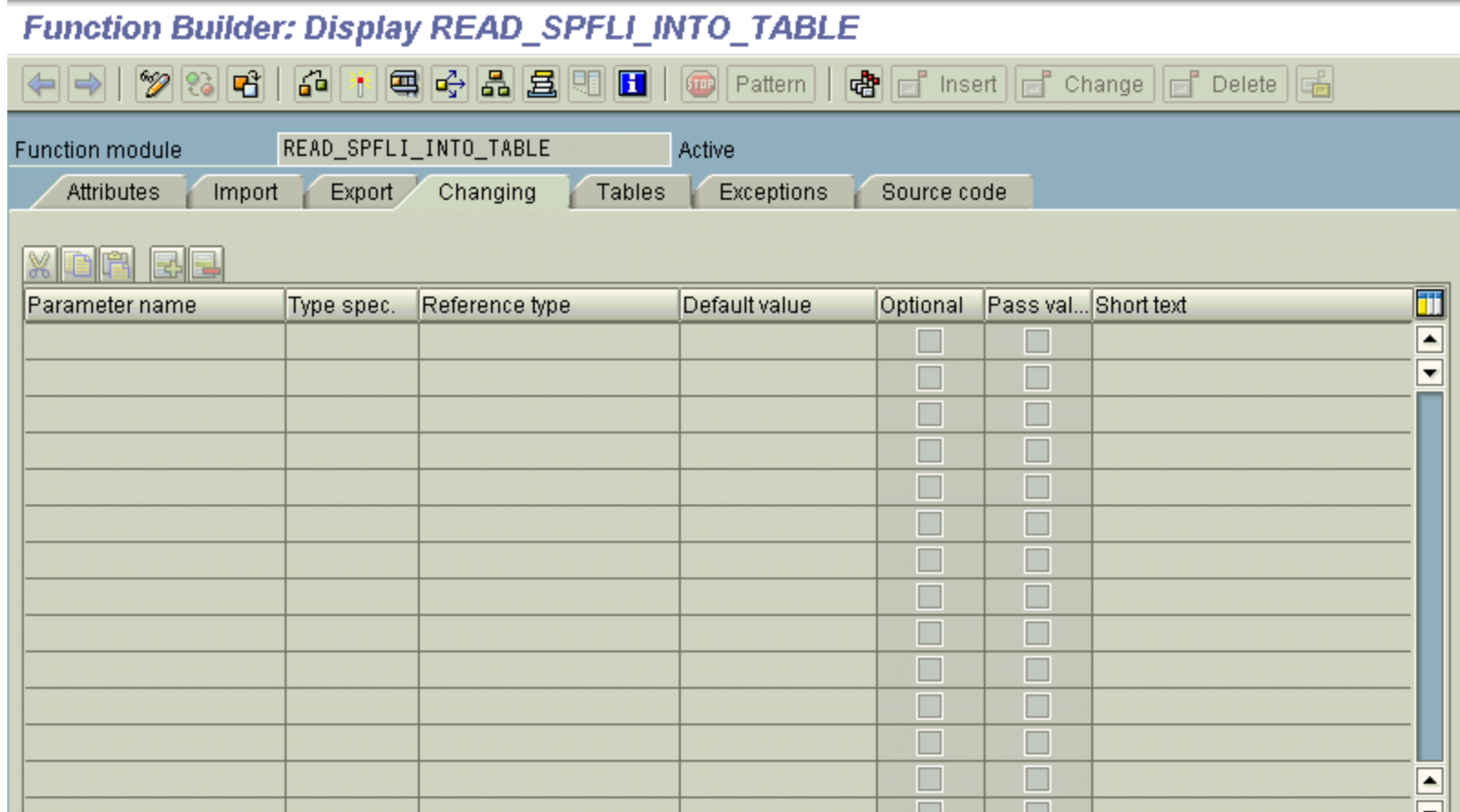
**Exceptions:** Error situations that can occur within the function module. The calling program uses exceptions to find out if an error has occurred in the function module. It can then react accordingly.







**Changing parameters :** Changing parameters are passed by reference or by value and result. Changing parameters act simultaneously as import and export parameters. They change the value passed to the function module and return it to the calling program.



**Tables parameters :We use these to pass internal tables. They are treated like CHANGING parameters. However, we can also pass internal tables with other parameters if you specify the parameter type appropriately.**

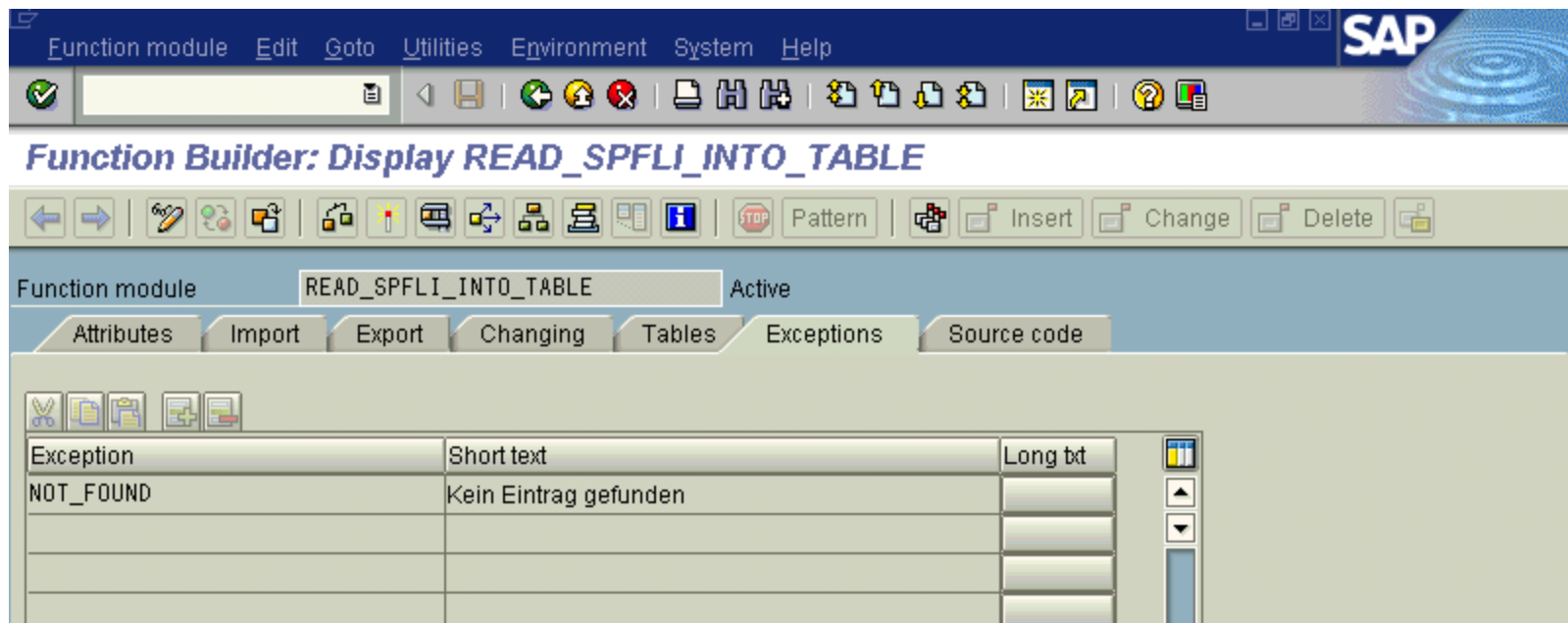
*Function Builder: Display BAPI\_SALESORDER\_CREATEFROMDAT1*

Function module: BAPI\_SALESORDER\_CREATEFROMDAT1 Active

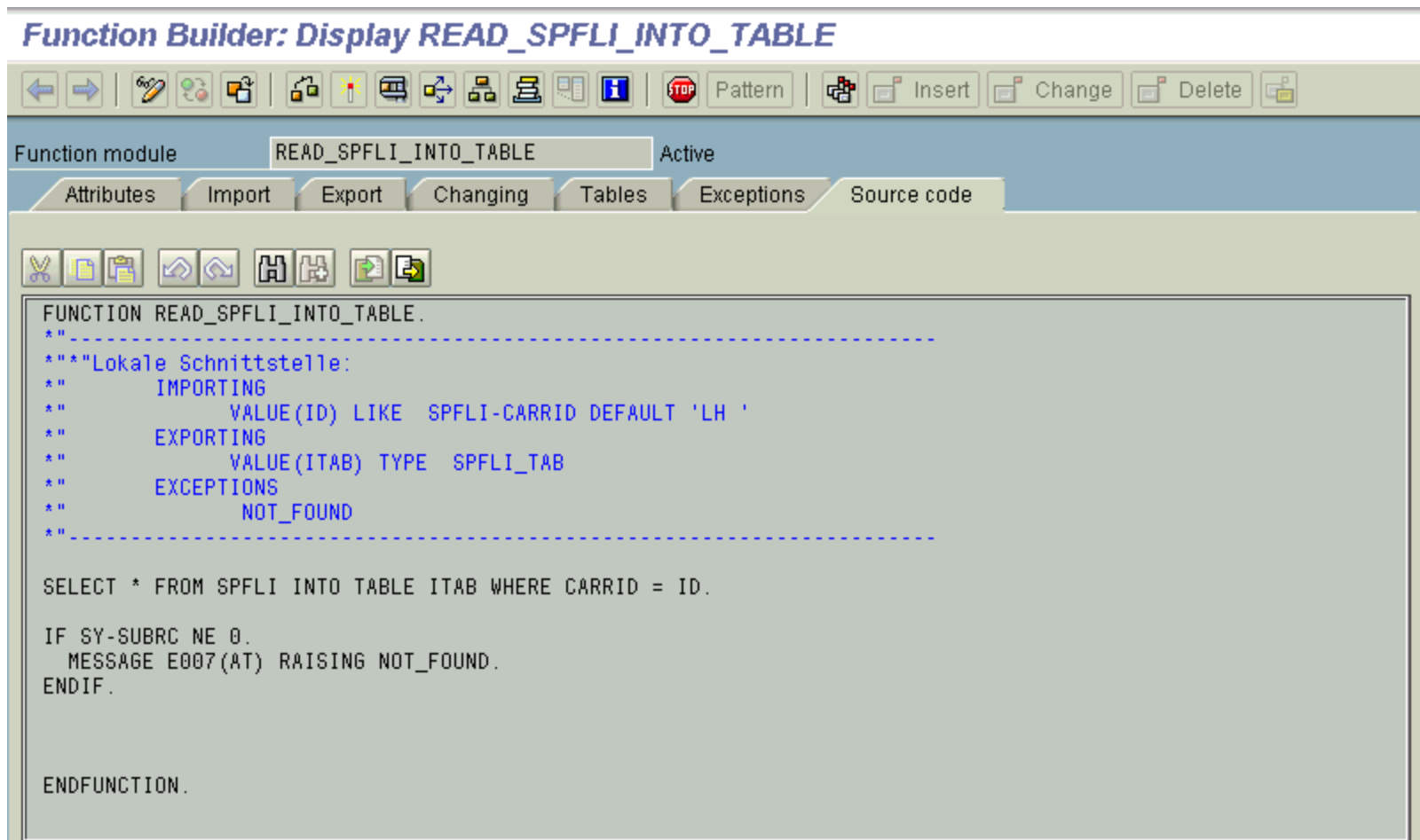
Attributes Import Export Changing Tables Exceptions Source code

Parameter name	Type spec.	Reference type	Optional	Short text	Long text
ORDER_ITEMS_IN	LIKE	BAPIITEMIN	<input type="checkbox"/>	Item data input	...
ORDER_PARTNERS	LIKE	BAPIPARTNR	<input type="checkbox"/>	Partners	...
ORDER_ITEMS_OUT	LIKE	BAPIITEMEX	<input checked="" type="checkbox"/>	Item data output	...
ORDER_CFGS_REF	LIKE	BAPICUCFG	<input checked="" type="checkbox"/>	Configuration: Reference data	...
ORDER_CFGS_INST	LIKE	BAPICUINS	<input checked="" type="checkbox"/>	Configuration: Instances	...
ORDER_CFGS_PART_OF	LIKE	BAPICUPRT	<input checked="" type="checkbox"/>	Configuration: Part-of specifications	...
ORDER_CFGS_VALUE	LIKE	BAPICUVAL	<input checked="" type="checkbox"/>	Configuration: Characteristic values	...
ORDER_CCARD	LIKE	BAPICCARD	<input checked="" type="checkbox"/>	Credit card data	...
ORDER_CFGS_BLOB	LIKE	BAPICUBLB	<input checked="" type="checkbox"/>		...
ORDER_SCHEDULE_EX	LIKE	BAPISDHEDU	<input checked="" type="checkbox"/>	Struture of VBEP with English Field N...	
			<input type="checkbox"/>		
			<input type="checkbox"/>		
			<input type="checkbox"/>		
			<input type="checkbox"/>		

- When creating function modules, we can define exceptions. The calling program determines whether and which exceptions it is to handle itself.
- We can assign the same error number to several exceptions.
- The OTHERS clause covers all exceptions not explicitly specified.



**This tab shows the source of the function module in the ABAP/4 editor. We can work with the source code in the same way as is done for normal ABAP/4 programs opened via forward navigation.**



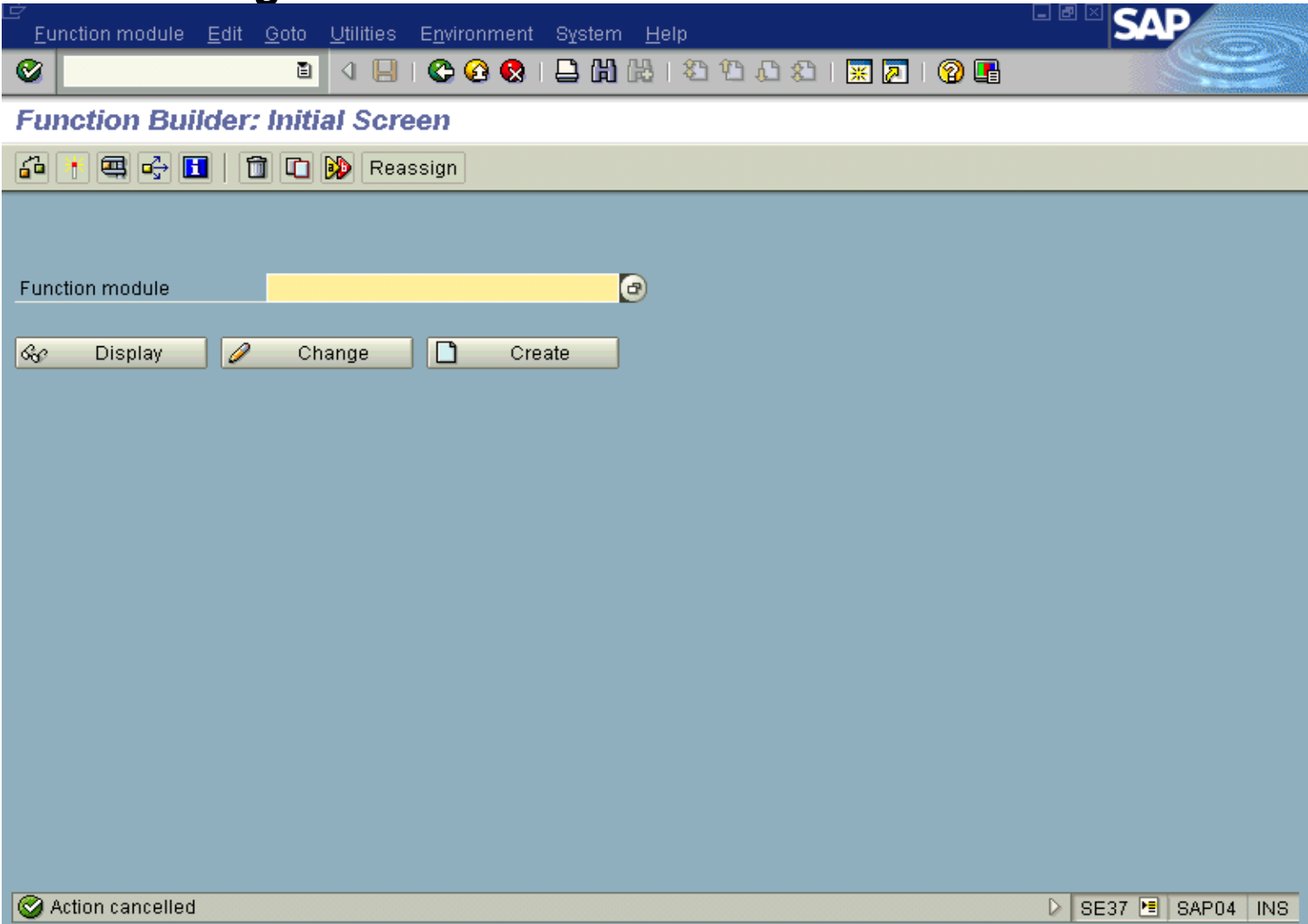
The screenshot displays the 'Function Builder: Display READ\_SPFLI\_INTO\_TABLE' window. The 'Source code' tab is active, showing the following ABAP code:

```
FUNCTION READ_SPFLI_INTO_TABLE.  
*-----  
*""Lokale Schnittstelle:  
*   IMPORTING  
*       VALUE(ID) LIKE SPFLI-CARRID DEFAULT 'LH '  
*   EXPORTING  
*       VALUE(ITAB) TYPE SPFLI_TAB  
*   EXCEPTIONS  
*       NOT_FOUND  
*-----  
  
SELECT * FROM SPFLI INTO TABLE ITAB WHERE CARRID = ID.  
  
IF SY-SUBRC NE 0.  
    MESSAGE E007(AT) RAISING NOT_FOUND.  
ENDIF.  
  
ENDFUNCTION.
```

- **Check whether a suitable function module already exists.  
If not, create one as follows.**
- **Create a function group, if no appropriate group exists yet.**
- **Create the function module.**
- **Define the function module interface by entering its parameters and exceptions.**
- **Write the actual ABAP code for the function module, adding any relevant global data to the TOP include.**
- **Activate the module.**
- **Test the module.**
- **Document the module and its parameters for other users.**
- **Release the module for general use.**

- **The CALL FUNCTION statement can pass import, export, and changing parameters either by value or by reference. Table parameters are always transferred by reference.**
- **If you declare the parameters with reference to ABAP Dictionary fields or structures, the system checks the type and length when the parameters are transferred. If the parameters from the calling program do not pass this check, the calling program terminates.**
- **At runtime, all function modules belonging to a function group are loaded with the calling program. As a result, you should plan carefully which functions really belong in a group and which do not. Otherwise, calling your function modules will unnecessarily increase the amount of memory required by the user.**

**The Function Builder allows us to create, test, and administer function modules in an integrated environment.**





### Function Builder: Initial Screen

Reassign

Create Function Module

Function module	READ_SPFLI_INTO_TABLE
Function group	DEMO_SPFLI
Short text	

Save

### Function Builder: Display READ\_SPFLI\_INTO\_TABLE

Function module  Active

Attributes Import Export Changing Tables Exceptions Source code

#### Classification

Function group  Demo. for Function Module Documentation  
Short text

#### Processing type

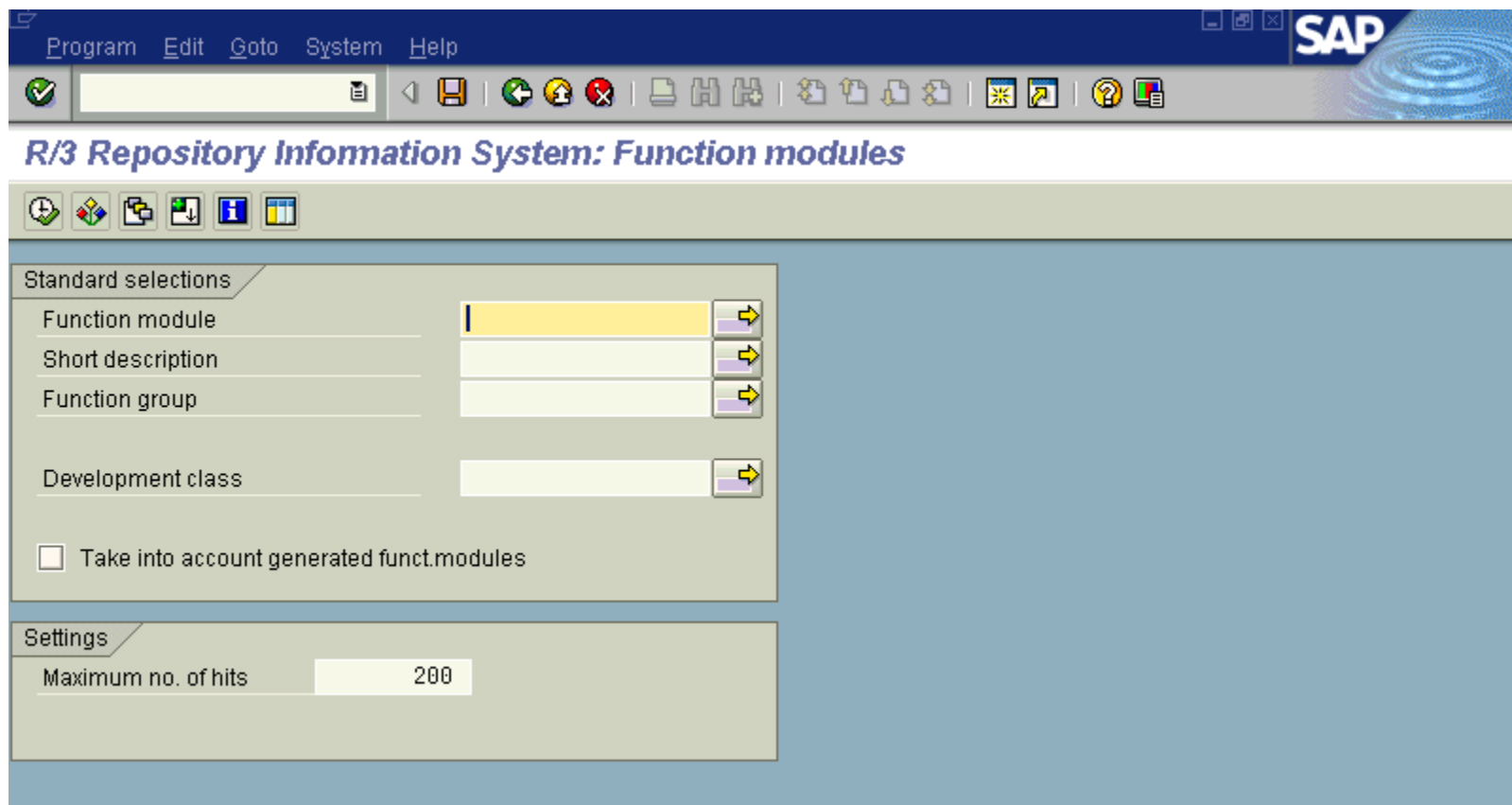
- Normal function module
- Remote-enabled module
- Update module
  - Start immed.
  - Immediate start, no restart
  - Start delayed
  - Coll.run

#### General data

Person responsible   
Last changed by   
Changed on   
Development class   
Program name   
INCLUDE name   
Original language   
Not released   
 Edit lock  
 Global

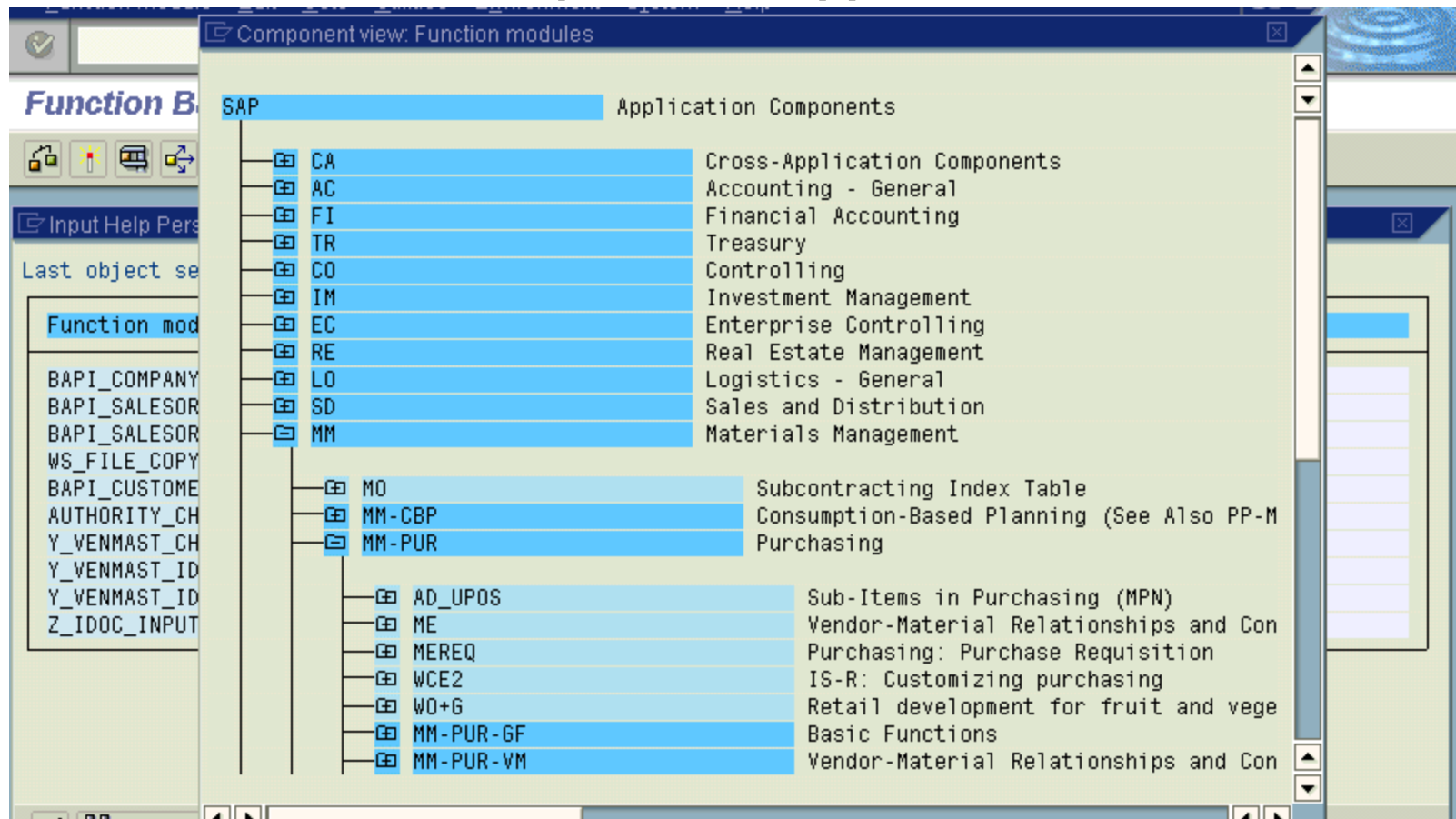
- **Normal Function Module** : Indicates that the function is a normal one
- **Remote-enabled Module** : Shows that the function is remote enabled
- **Update Module**
  - **Start Update Immediately** :The function module is processed immediately in the update task.
  - **Immediate Start, No restart** : The function module will be edited in the update task. It cannot be updated subsequently.
  - **Start Delayed** :The function module is processed in the update task as a low priority item. You use delayed update primarily for database changes that are not time-critical (e.g. statistical updates).
  - **Collective Run** :A number of similar function modules that previously used to run individually in the V2 update process can be grouped together and run collectively.

- **Using the Repository Information System**  
To search for a module, choose Find from the initial screen of the Function Builder. The system displays the standard Function Module search screen.



- **Using the Application Hierarchy**

**The Application Hierarchy provides an overview of all the applications in your R/3 system. You can use this hierarchy to display function modules associated with particular applications.**



**CALL FUNCTION <module>**

**[EXPORTING  $f_1 = a_1 \dots f_n = a_n$ ]**

**[IMPORTING  $f_1 = a_1 \dots f_n = a_n$ ]**

**[CHANGING  $f_1 = a_1 \dots f_n = a_n$ ]**

**[TABLES  $f_1 = a_1 \dots f_n = a_n$ ]**

**[EXCEPTIONS  $e_1 = r_1 \dots e_n = r_n$  [ERROR\_MESSAGE =  $r_E$ ]**

**[OTHERS =  $r_o$ ] ].**

```
PROGRAM CALL_FUNCTION.  
DATA: TEXT(10) TYPE C VALUE  
      '0123456789',  
      TEXT1(6) TYPE C,  
      TEXT2(6) TYPE C.  
PARAMETERS POSITION TYPE I.  
CALL FUNCTION  
'STRING_SPLIT_AT_POSITION'  
  EXPORTING  
    STRING = TEXT  
    POS = POSITION  
  IMPORTING  
    STRING1 = TEXT1  
    STRING2 = TEXT2  
EXCEPTIONS  
STRING1_TOO_SMALL = 1  
STRING2_TOO_SMALL = 2  
POS_NOT_VALID = 3  
OTHERS = 4.
```

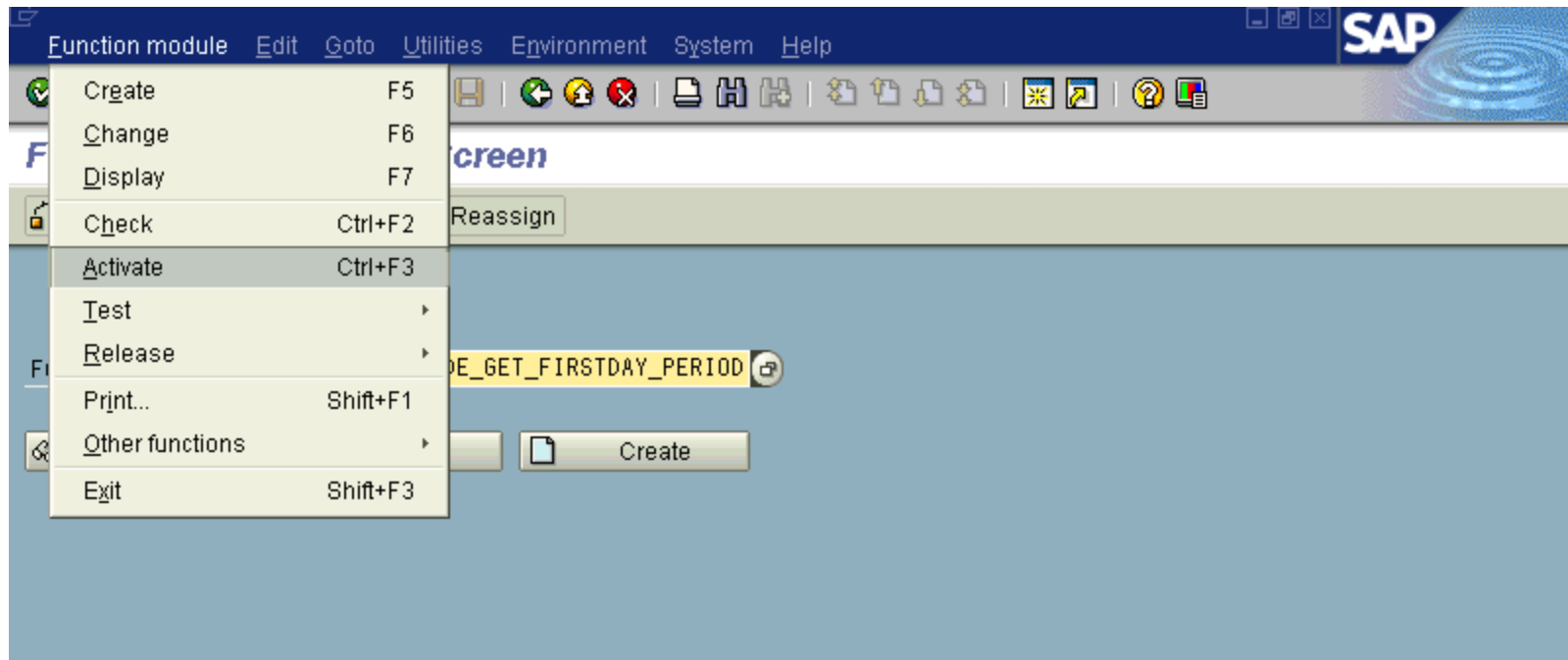
```
CASE SY-SUBRC.  
WHEN 0.  
WRITE: / TEXT, / TEXT1, / TEXT2.  
WHEN 1.  
WRITE 'Target field 1 too short!'.  
WHEN 2.  
WRITE 'Target field 2 too short!'.  
WHEN 3.  
WRITE 'Invalid split position!'.  
WHEN 4.  
WRITE 'Other errors!'.  
ENDCASE.
```

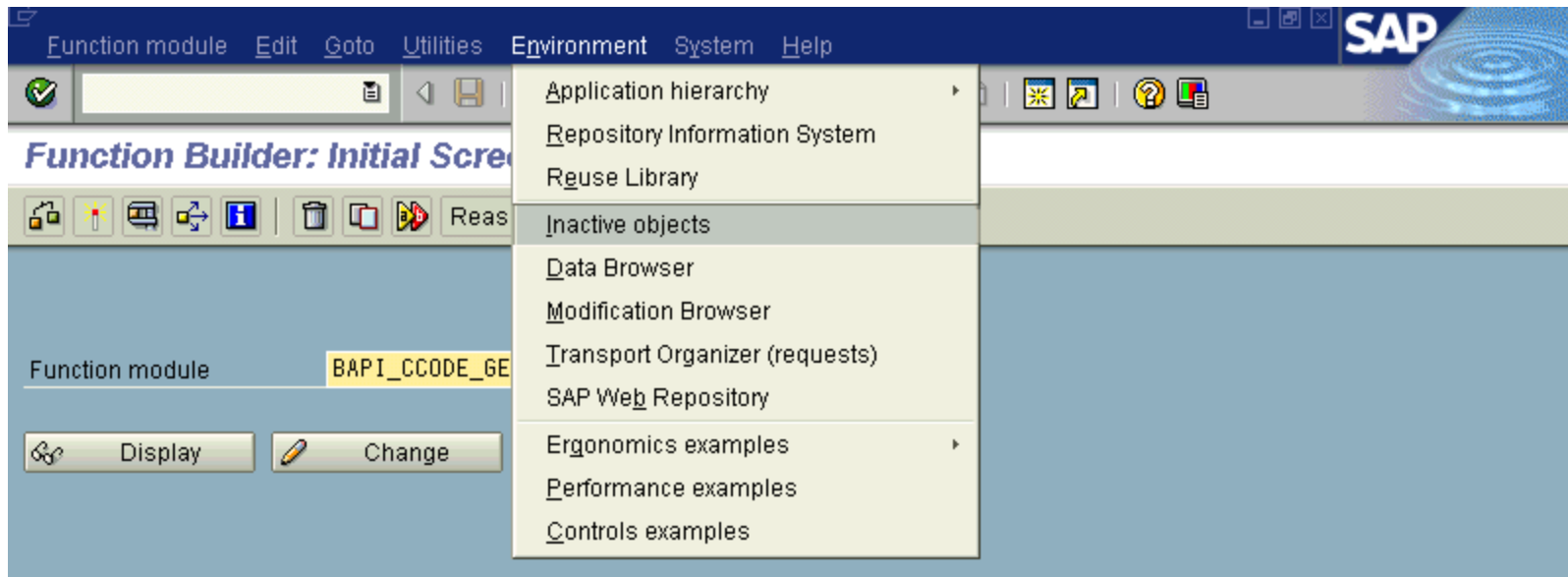
**The documentation for the function module is done in the Function Builder. There are two kinds of documentation - parameter documentation, and full function module documentation.**

- The parameter documentation must provide users with information about the different parameters and exceptions.**
- Function module documentation contains important detailed information about the task of the function module. A detailed documentation will help us to understand the function module without having to examine its source code.**

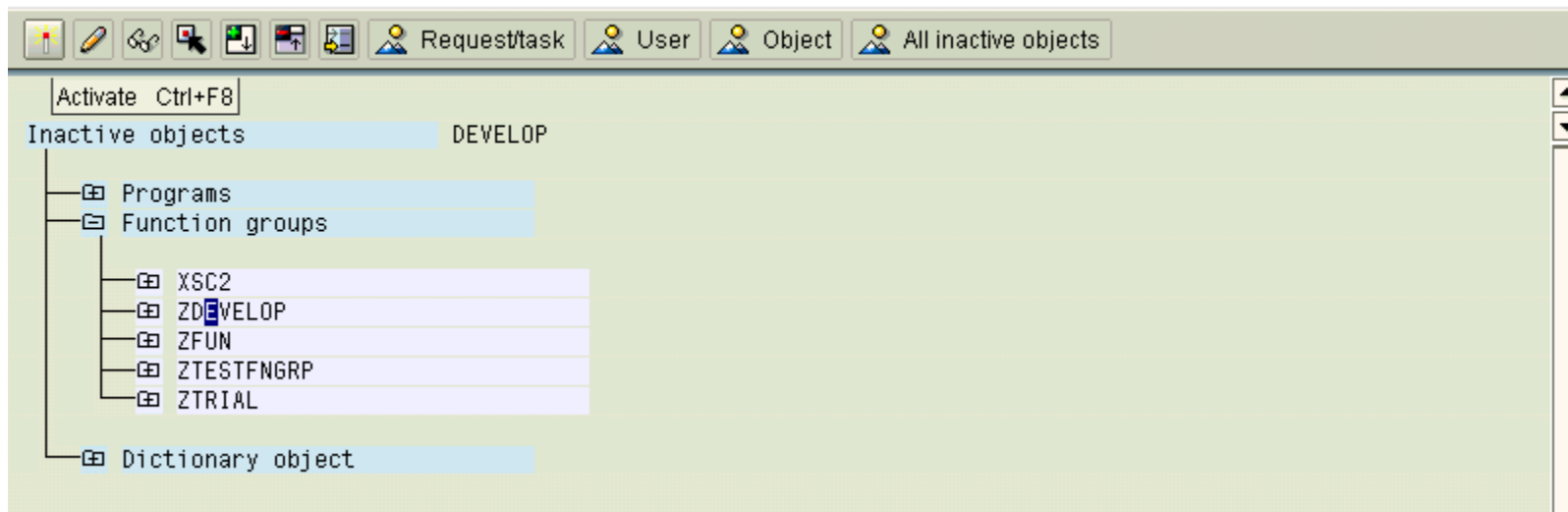


**Function Module can be activated from the menu as shown below.**





### Overview: Inactive Objects for User DEVELOP



- We can test function modules without having to include them in a program using the Function Builder. When we test a function module, the system displays any exceptions. The system also identifies the time required to execute the module in microseconds.

## Example Function : BAPI\_CCODE\_GET\_FIRSTDAY\_PERIOD

Import Parameters: Company Code, Fiscal Period, Fiscal Year

The screenshot shows the SAP Function Builder interface. The title bar reads 'Function modules Edit Goto Utilities System Help' and the SAP logo is visible in the top right. Below the title bar is a toolbar with various icons. The main window title is 'Test Function Module: Initial Screen'. Below this, there are buttons for 'Debugging' and 'Test data directory'. The main area contains the following text:

```

Test for function group      0002
Function module             BAPI_CCODE_GET_FIRSTDAY_PERIOD
Upper/lower case           
RFC target sys:           

```

Below the text is a table with two columns: 'Import parameters' and 'Value'.

Import parameters	Value
COMPANYCODEID	0001
FISCAL_PERIOD	04
FISCAL_YEAR	2001

The function **BAPI\_CCODE\_GET\_FIRSTDAY\_PERIOD** when executed with values given in the previous screen gives the following output. It also shows the time required for executing the function module.

The screenshot shows the SAP Test Function Module: Result Screen. The title bar includes 'Function modules', 'Edit', 'Goto', 'Utilities', 'System', and 'Help'. The main content area displays the following information:

```

Test for function group      0002
Function module             BAPI_CCODE_GET_FIRSTDAY_PERIOD
Upper/lower case           

Runtime:                   103 Microseconds


RFC target sys:

```

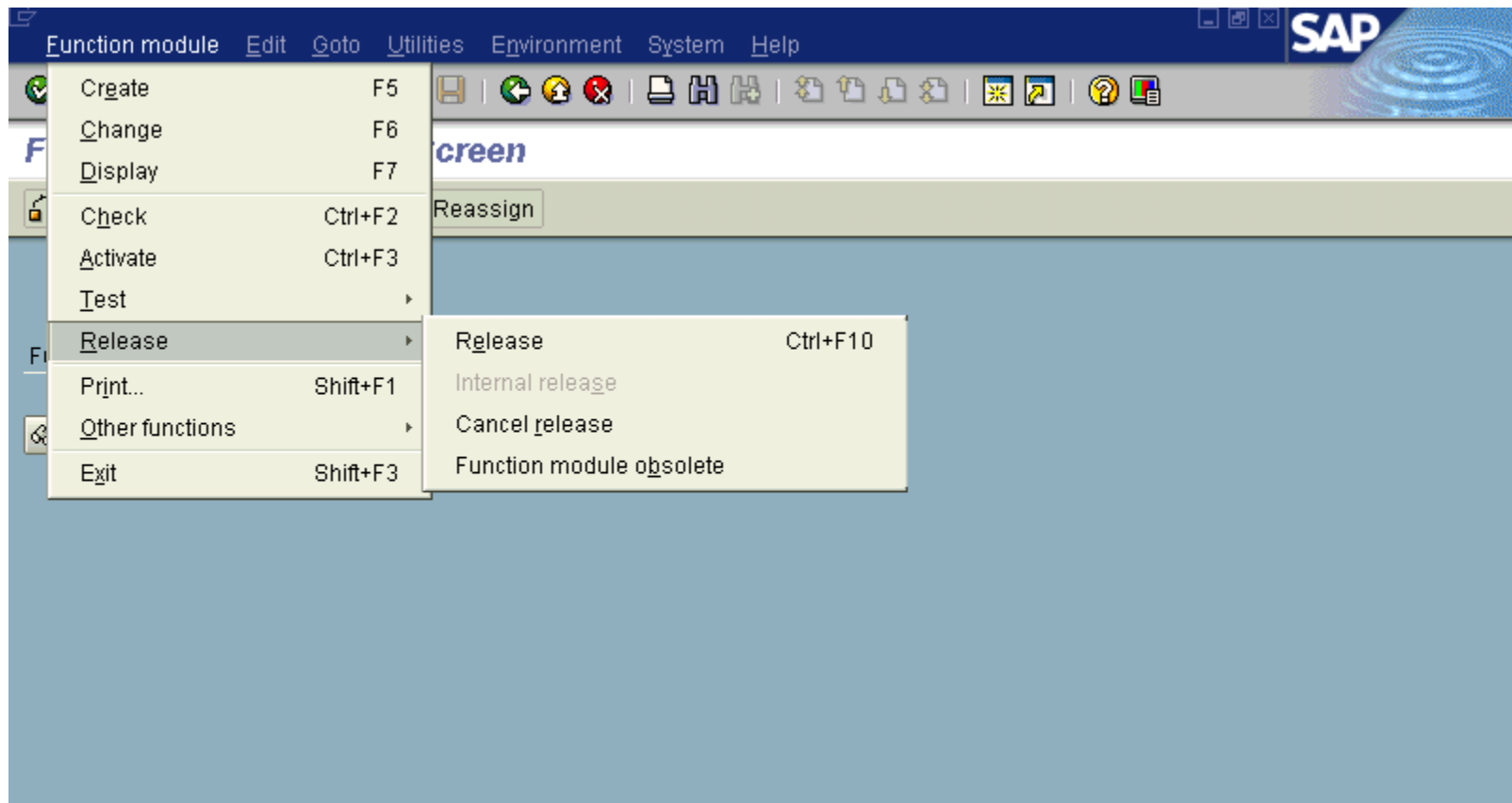
Below the text, there are two tables. The first table shows import parameters and the second table shows export parameters.

Import parameters	Value
COMPANYCODEID	0001
FISCAL_PERIOD	04
FISCAL_YEAR	2001

Export parameters	Value
FIRST_DAY_OF_PERIOD	01.04.2001
RETURN	 000

**Releasing a function module is a purely administrative gesture with no effect on the function or its usability. Releasing a function module signals that a developer has tested it. When a function module is released its documentation is released for translation and appears in the relevant translator's worklist.**



## **Summary**

**This section explained :**

- **The creation of function modules**
- **Components of function modules**
- **Releasing and activation of function modules**