Ishnoor Singh Sethi Sep 24, 2021 3 min read

BAPI - Choosing The Right Strategy

Updated: May 31, 2022

Business Application Programming Interface (BAPI)

SAP BAPI (Business Application Programming Interface) is a standard interface to the business object models in SAP products.

BAPIs are the primary method through which customer code and third-party applications interact with SAP products. BAPIs wrap the internal layers of SAP's business object model to ensure that all business logic, validations and authorisation checks are executed properly when accessing or changing business objects.

BAPIs are implemented as function modules that call SAP internal code. Depending on which set of BAPIs is being used, they may call business object models defined using the Business Objects Processing Framework (BOPF) or legacy models defined using programs, tables and function modules.

Types of BAPIs —

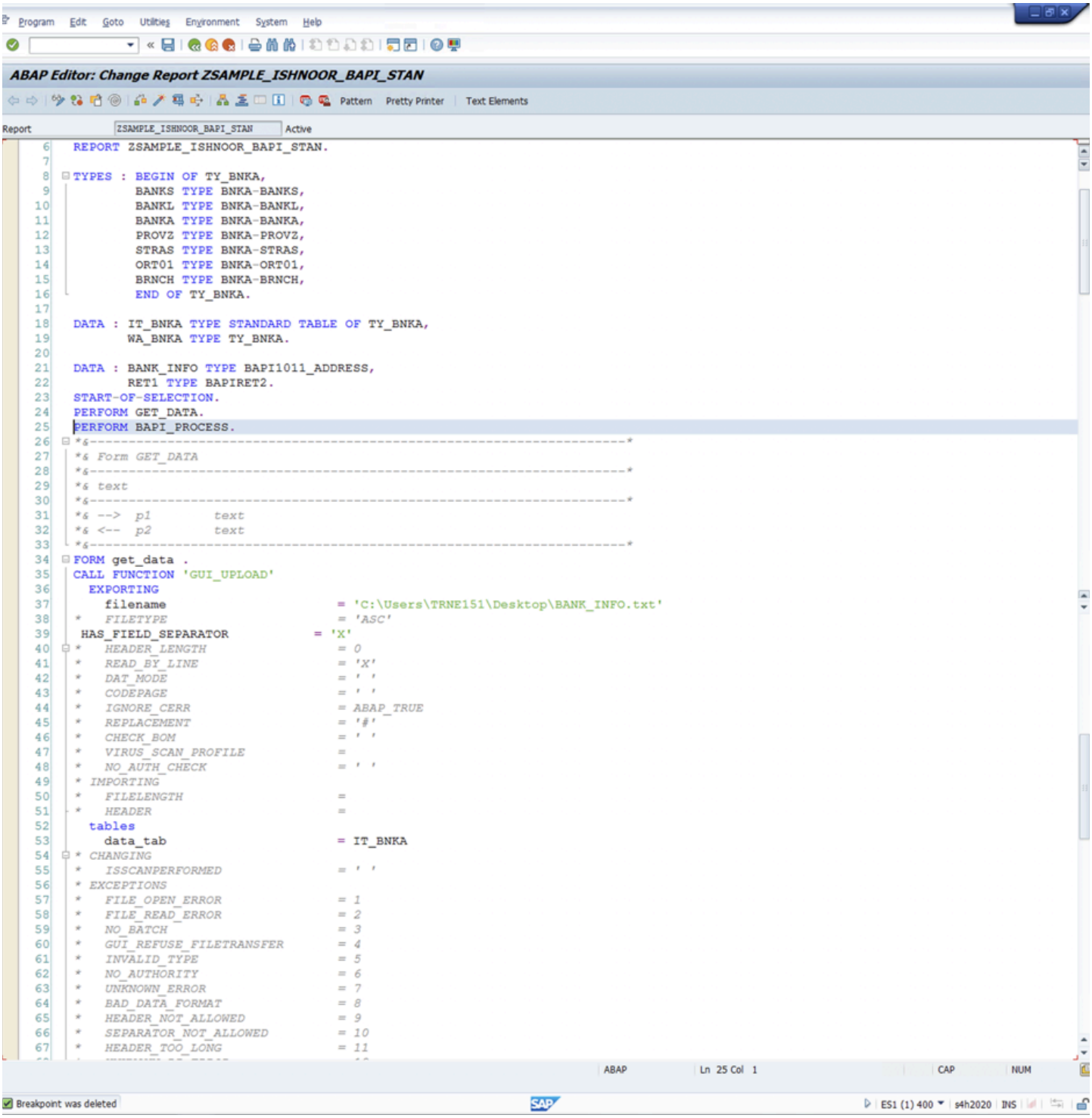
1. Standard BAPI — BAPI created by SAP labs.

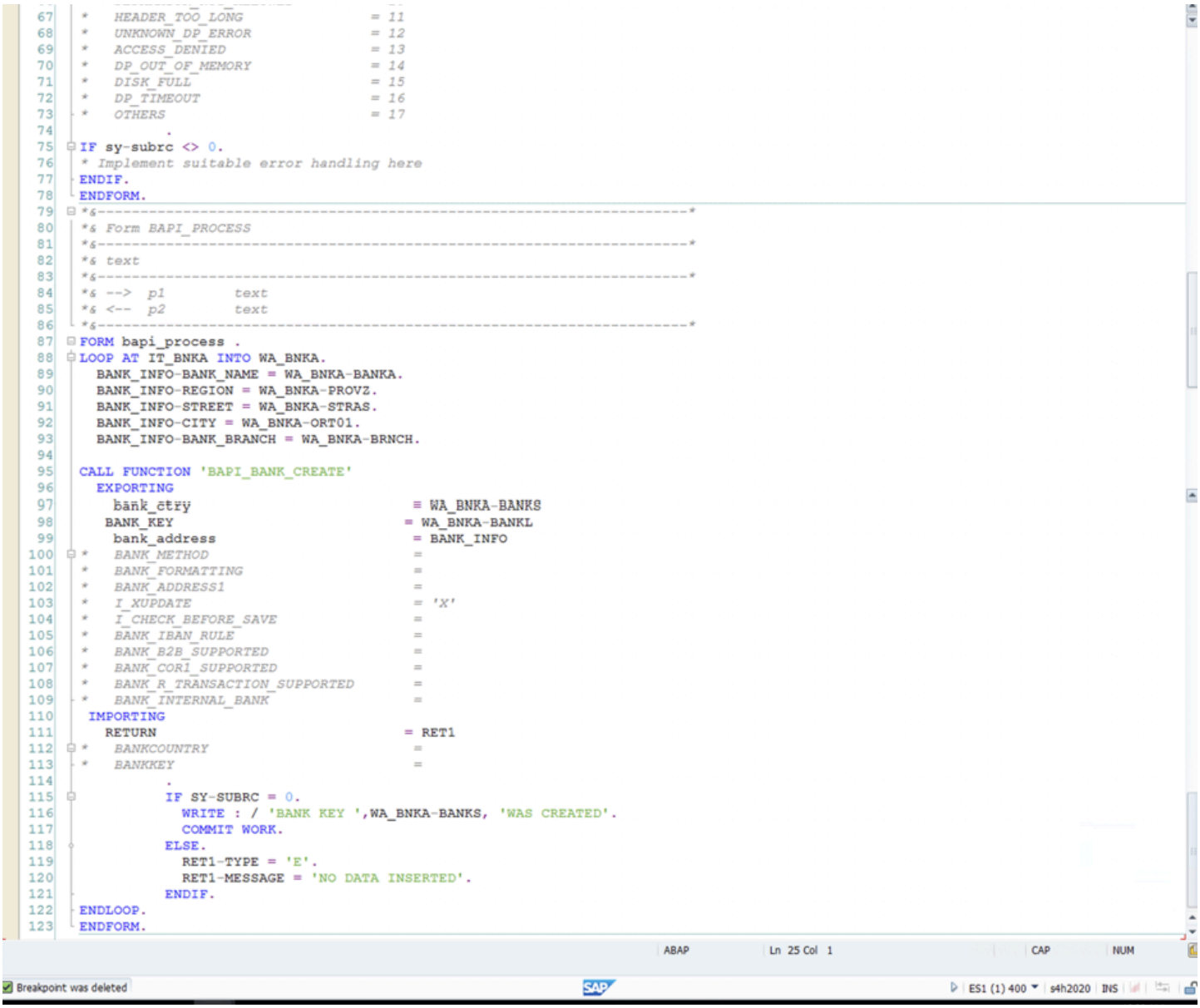
List of BAPIs in SAP ABAP — https://wiki.scn.sap.com/wiki/display/ABAP/List+of+BAPI%27s

2. Custom BAPI — BAPI created by users

Uploading data to standard table using standard BAPI (BAPI_BANK_CREATE) .

Coding Screen —





```
*   HEADER_TOO_LONG                = 11
*   UNKNOWN_DP_ERROR               = 12
*   ACCESS_DENIED                  = 13
*   DP_OUT_OF_MEMORY               = 14
*   DISK_FULL                      = 15
*   DP_TIMEOUT                     = 16
*   OTHERS                         = 17
          .
IF sy-subrc <> 0.
* Implement suitable error handling here
ENDIF.
ENDFORM.
*&---------------------------------------------------------------------*
*& Form BAPI_PROCESS
*&---------------------------------------------------------------------*
*& text
*&---------------------------------------------------------------------*
*& -->  p1        text
*& <--  p2        text
*&---------------------------------------------------------------------*
FORM bapi_process .
LOOP AT IT_BNKA INTO WA_BNKA.
  BANK_INFO-BANK_NAME = WA_BNKA-BANKA.
  BANK_INFO-REGION = WA_BNKA-PROVZ.
  BANK_INFO-STREET = WA_BNKA-STRAS.
  BANK_INFO-CITY = WA_BNKA-ORT01.
  BANK_INFO-BANK_BRANCH = WA_BNKA-BRNCH.

CALL FUNCTION 'BAPI_BANK_CREATE'
  EXPORTING
    bank_ctry                     = WA_BNKA-BANKS
   BANK_KEY                      = WA_BNKA-BANKL
    bank_address                  = BANK_INFO
*   BANK_METHOD                   =
*   BANK_FORMATTING               =
*   BANK_ADDRESS1                 =
*   I_XUPDATE                     = 'X'
*   I_CHECK_BEFORE_SAVE           =
*   BANK_IBAN_RULE                =
*   BANK_B2B_SUPPORTED            =
*   BANK_CORI_SUPPORTED           =
*   BANK_R_TRANSACTION_SUPPORTED  =
*   BANK_INTERNAL_BANK            =
 IMPORTING
   RETURN                        = RET1
*   BANKCOUNTRY                   =
*   BANKKEY                       =
          .
          IF SY-SUBRC = 0.
            WRITE : / 'BANK KEY ',WA_BNKA-BANKS, 'WAS CREATED'.
            COMMIT WORK.
          ELSE.
            RET1-TYPE = 'E'.
            RET1-MESSAGE = 'NO DATA INSERTED'.
          ENDIF.
ENDLOOP.
ENDFORM.
```

ABAP Ln 25 Col 1 CAP NUM

Breakpoint was deleted ES1 (1) 400 s4h2020 INS

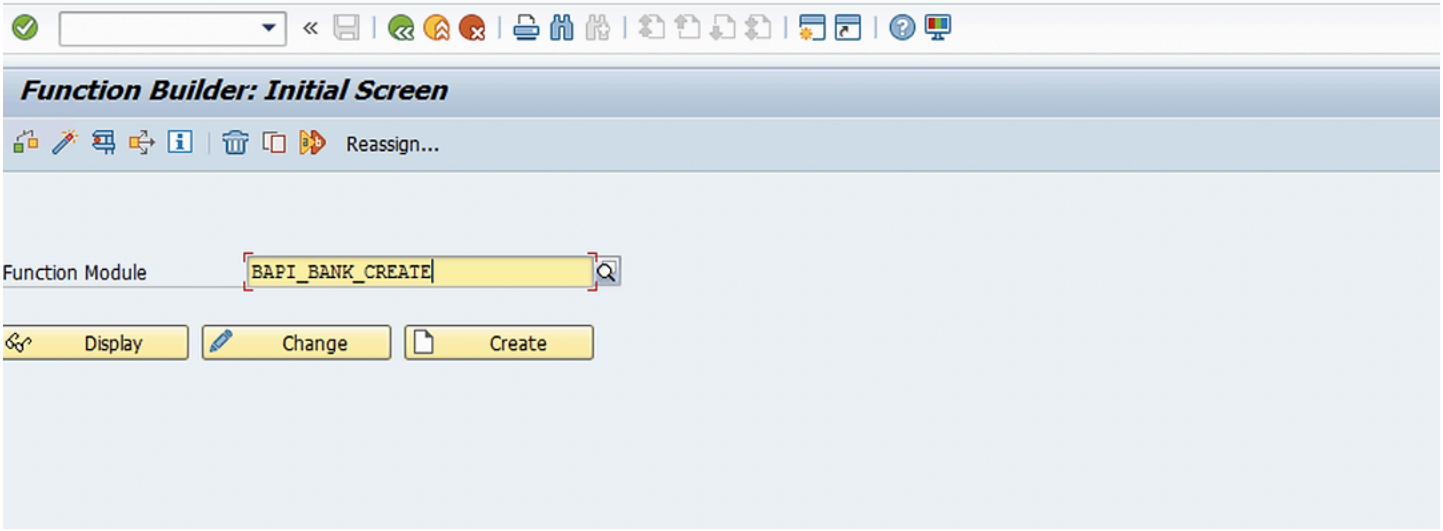
Key points in BAPI programs —

```
DATA : BANK_INFO TYPE BAPI1011_ADDRESS,
       RET1 TYPE BAPIRET2.
```

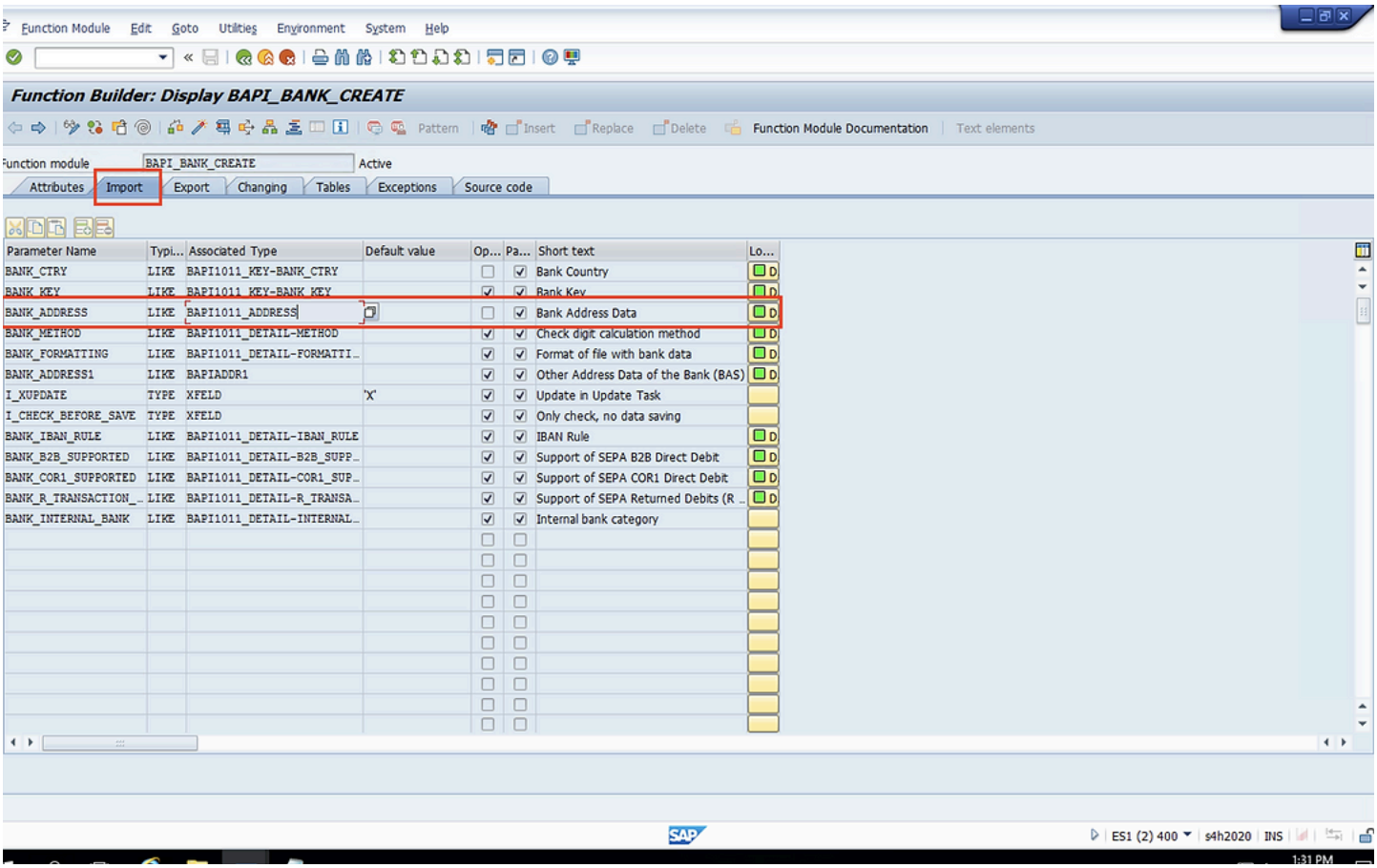
We got these two fields from the function that we called i.e “BAPI_BANK_CREATE”.

Steps to know particular data type for variables —

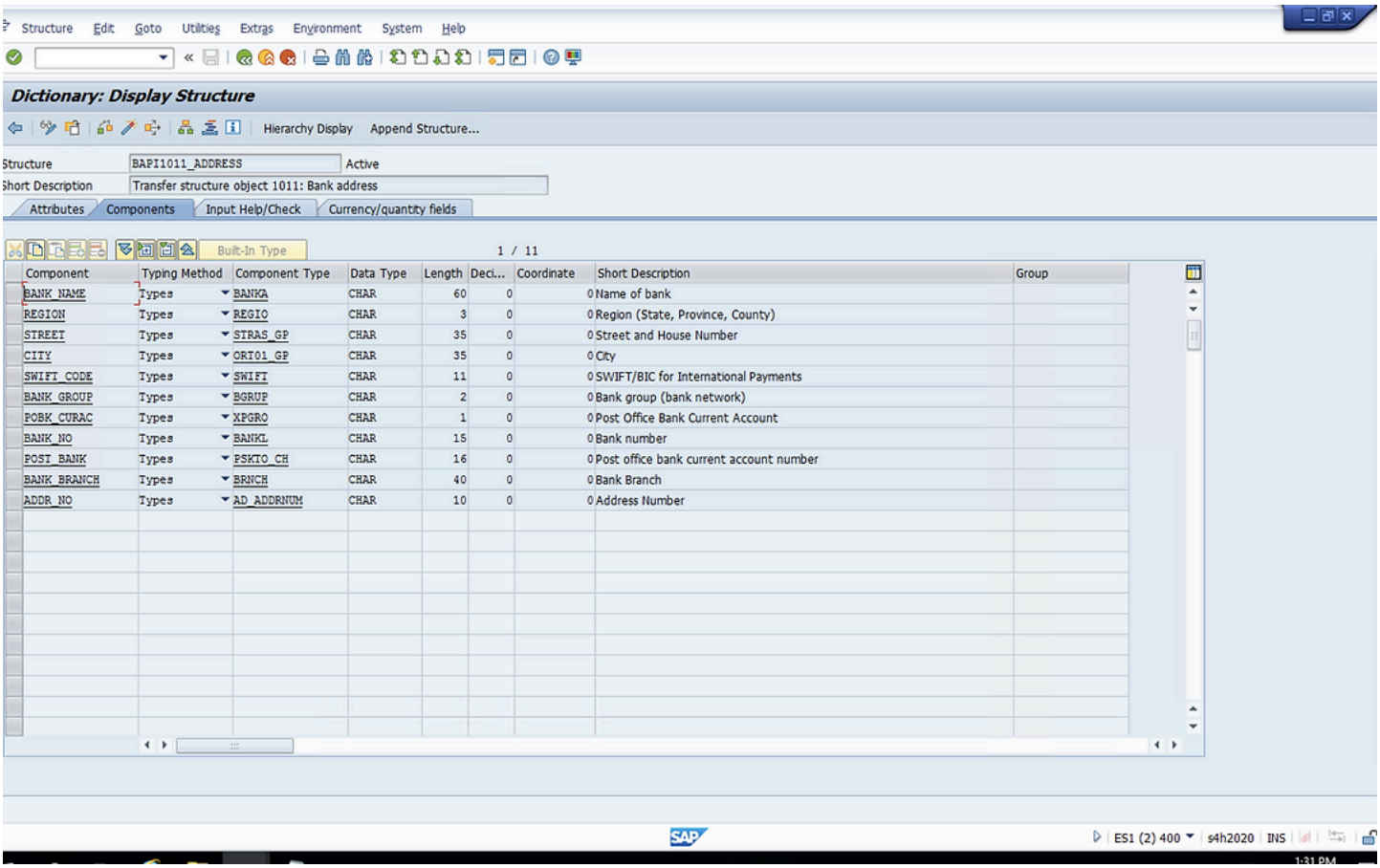
Step 1 - On “SE37” screen type the desired function module and click on display



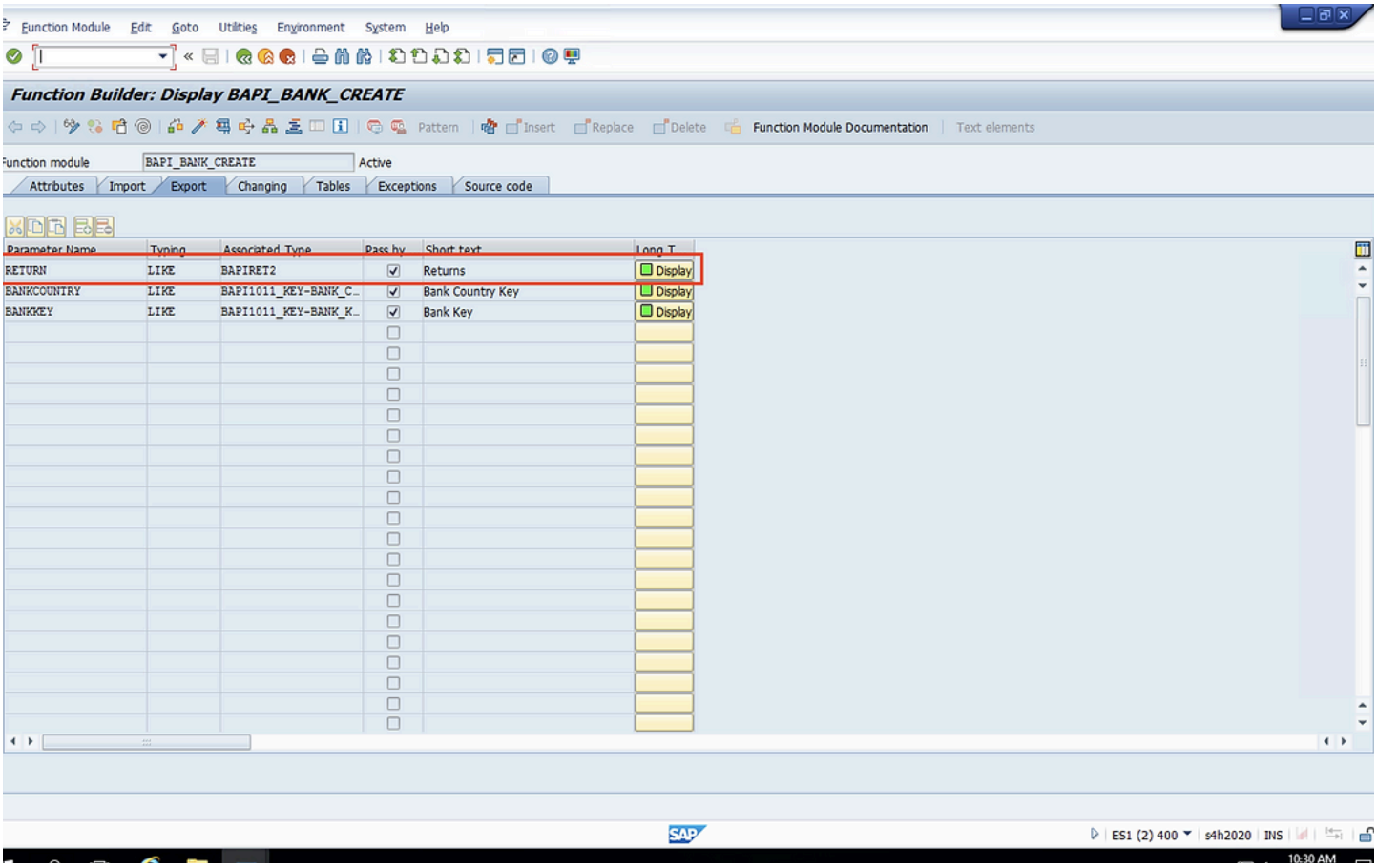
Step 2 - Click on import button and you will see the importing parameters. Search for the desired data type or structure. In this case Bank_Address is the required field.



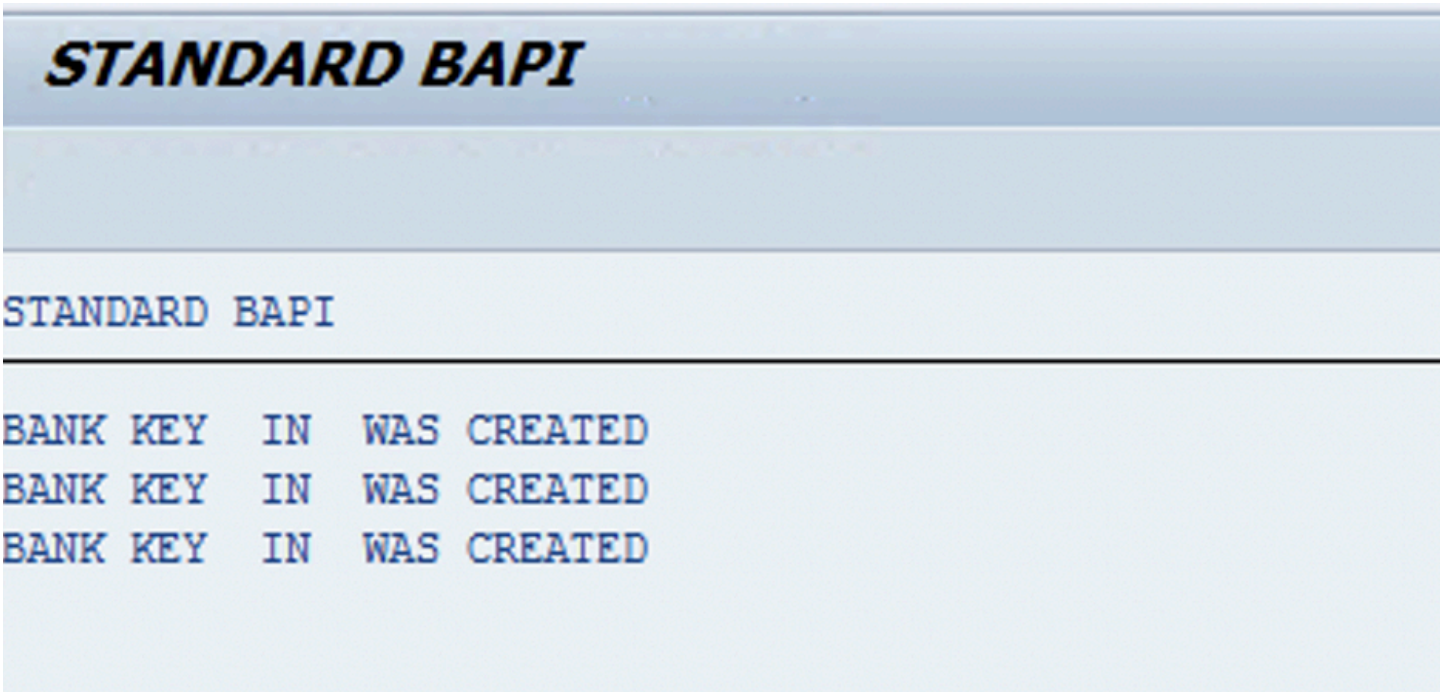
Step 3 - Double click on the Bank_Address and you will see the components in this structure. A variable of this structure type will contain all the components of the structure.

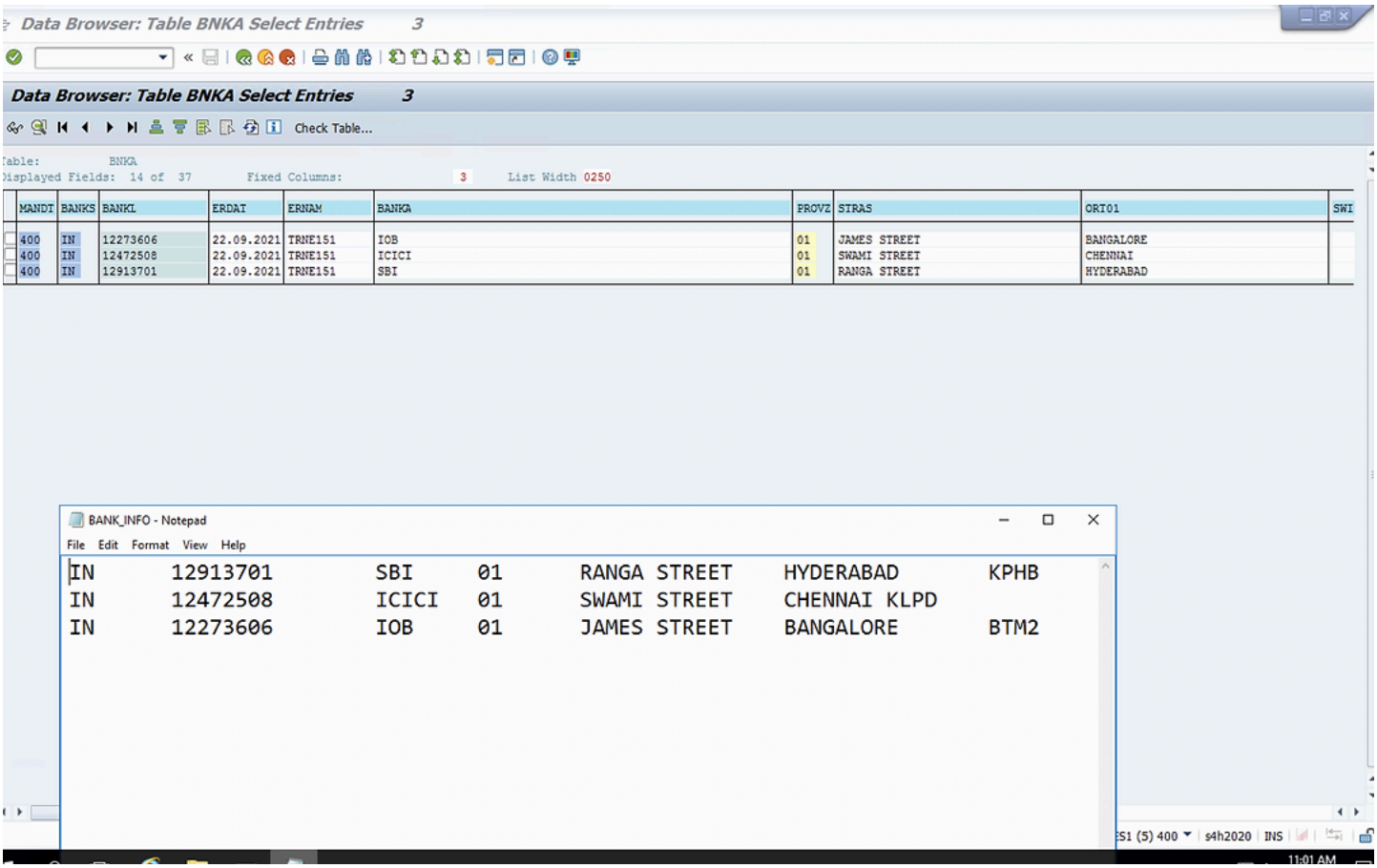


Step 4 - Now go back and click on export button and get the return data type.



Output —





Remote Function Call (RFC)

Communication between applications of different systems in the SAP environment includes connections between SAP systems as well as between SAP systems and non-SAP systems. Remote Function Call (RFC) is the standard

SAP interface for communication between SAP systems. RFC calls a function to be executed in a remote system. There is now a whole series of different RFC variants, each of which has different properties and is used for a specific purpose.

Custom BAPI

Step 1 - Create import and export structure using tcode “SE11”.

Step 2 - Create Function Module using import and export structure using tcode “SE37”.

Step 3 - After creation of Function Module convert it into RFC and release that RFC.

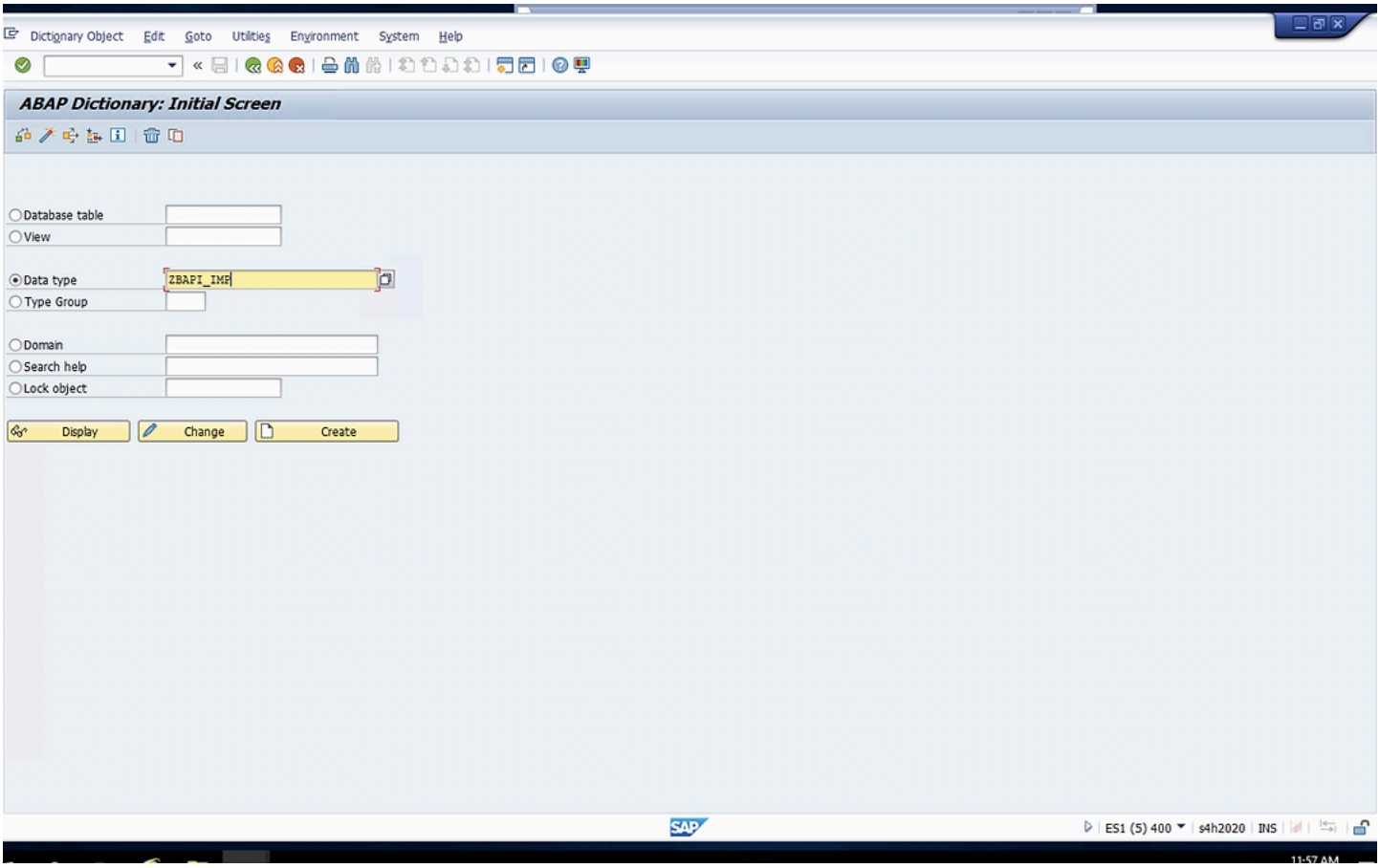
Step 4 - Create business object using tcode “SWO1”.

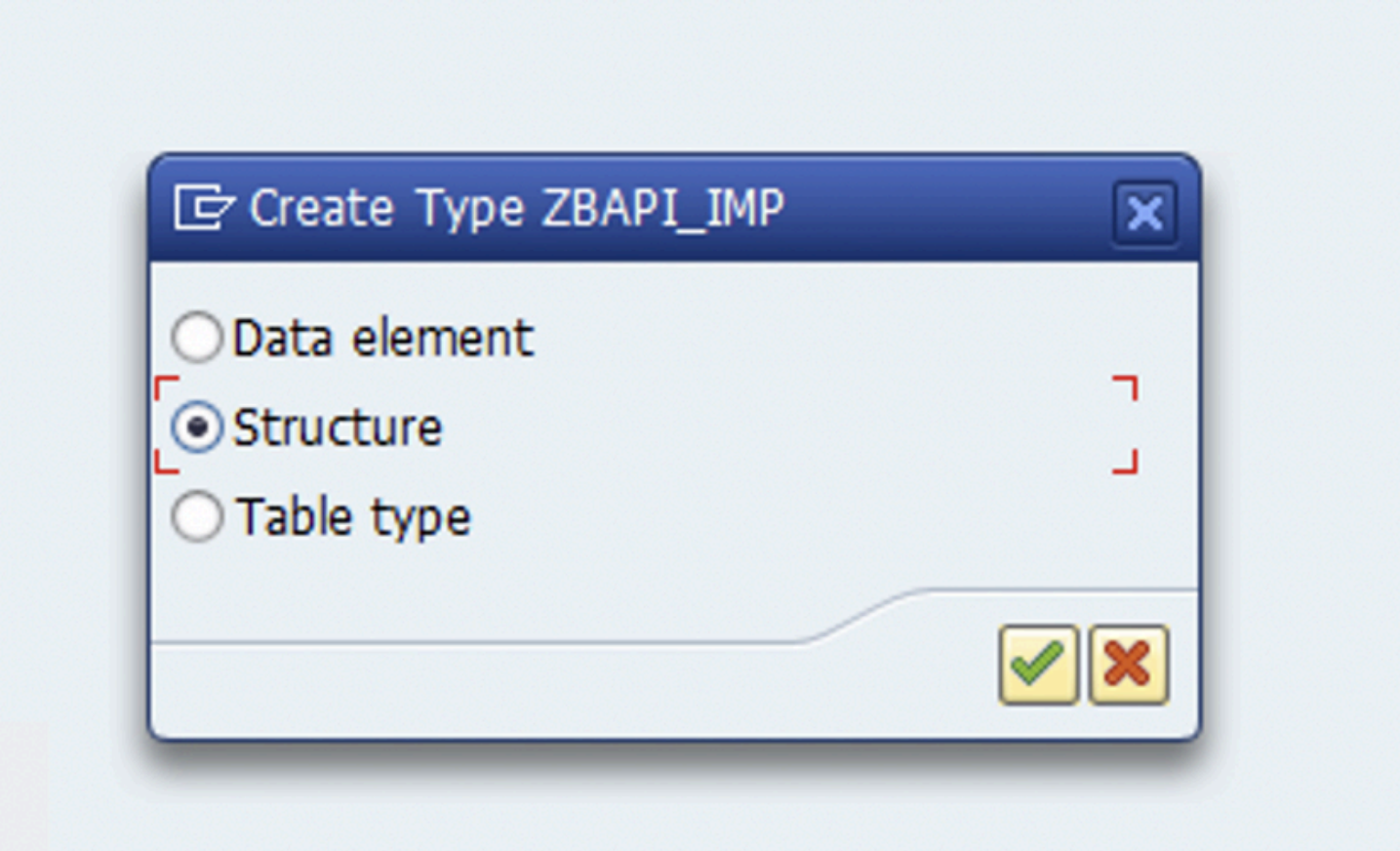
Step 5 - After BAPI creation we have to add RFC Function module into business object.

Step 6 - At last release business object components and component types and then finally release Business Object.

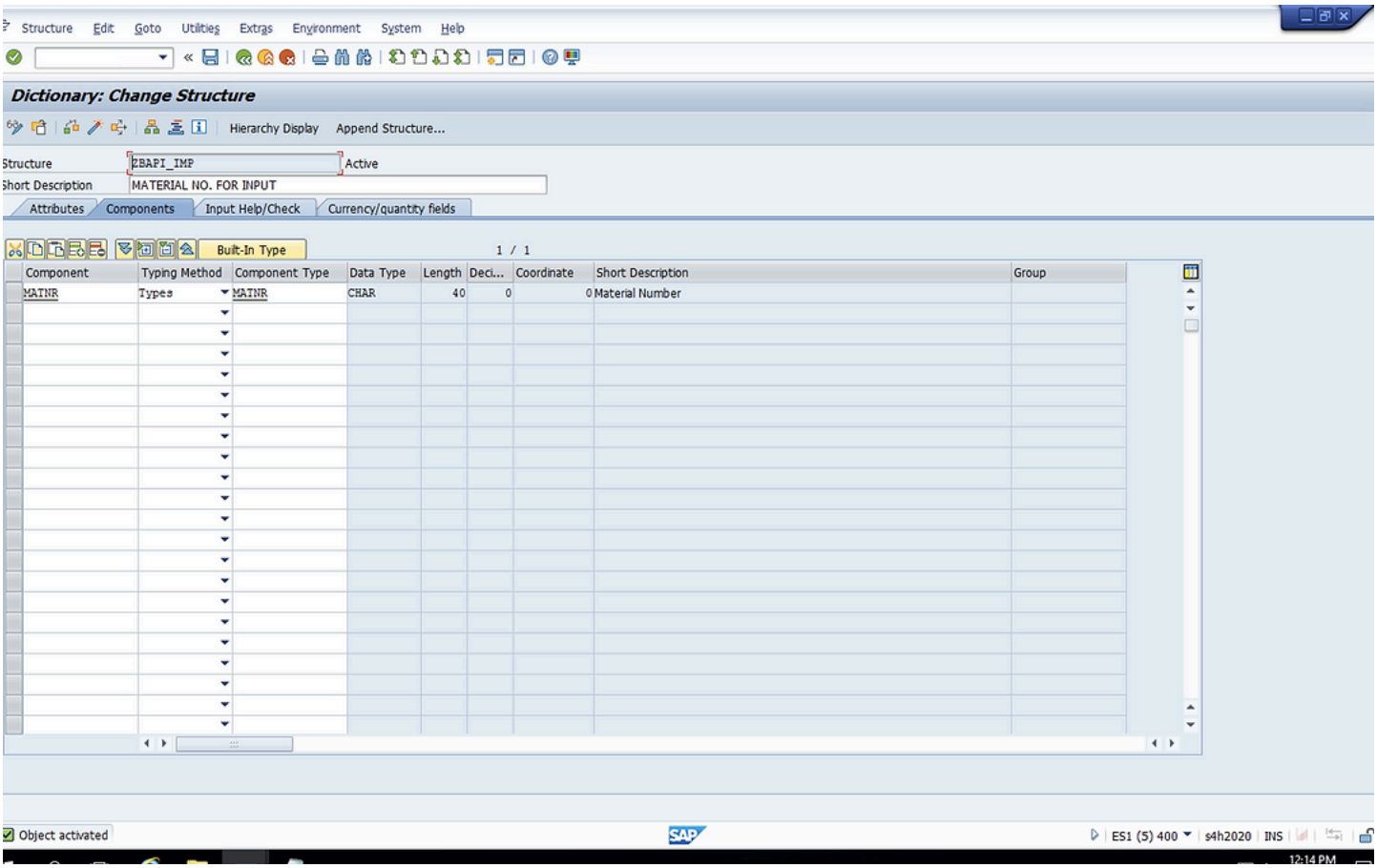
Step 1 - Create import and export structure using tcode “SE11”.

Import structure —

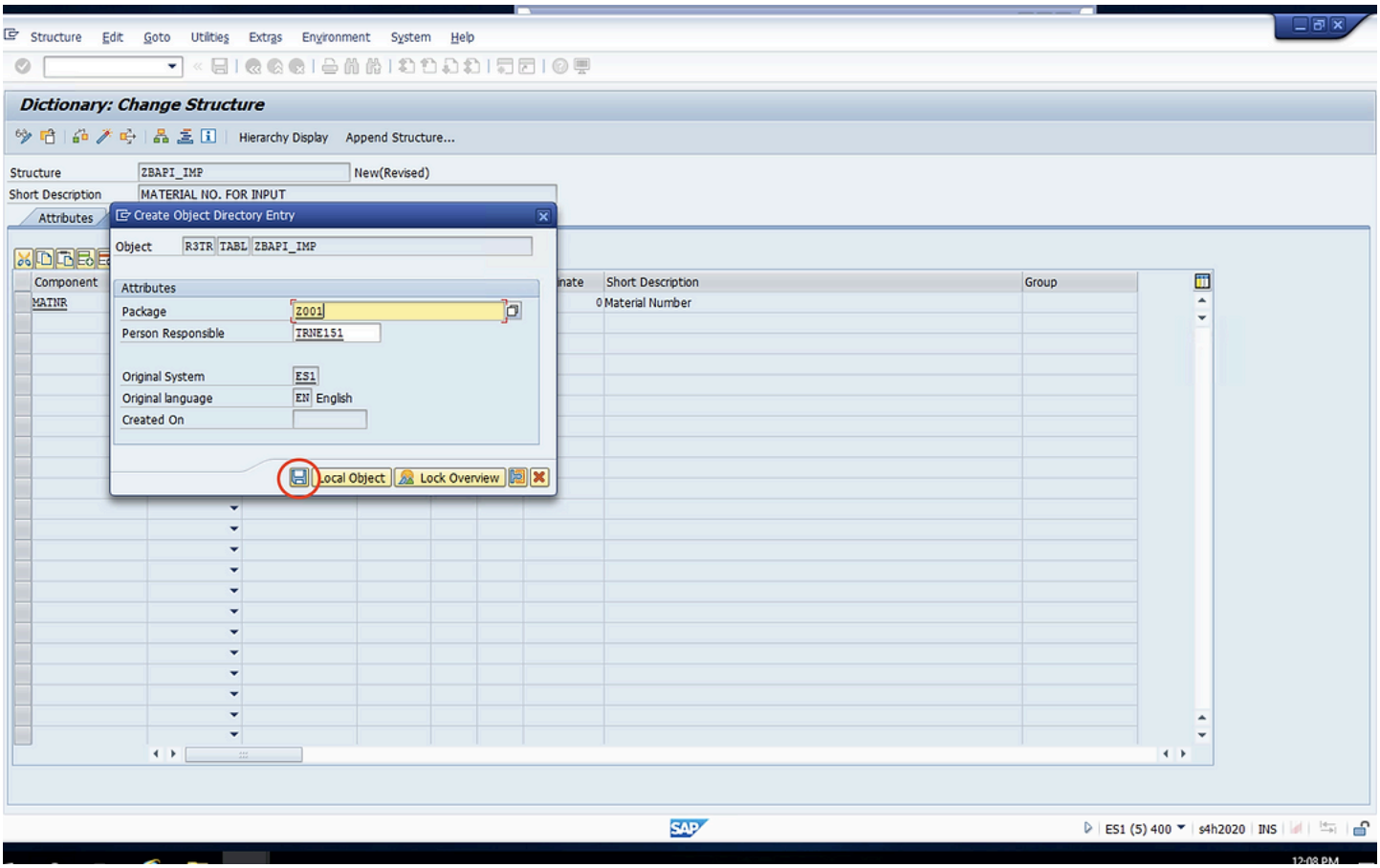




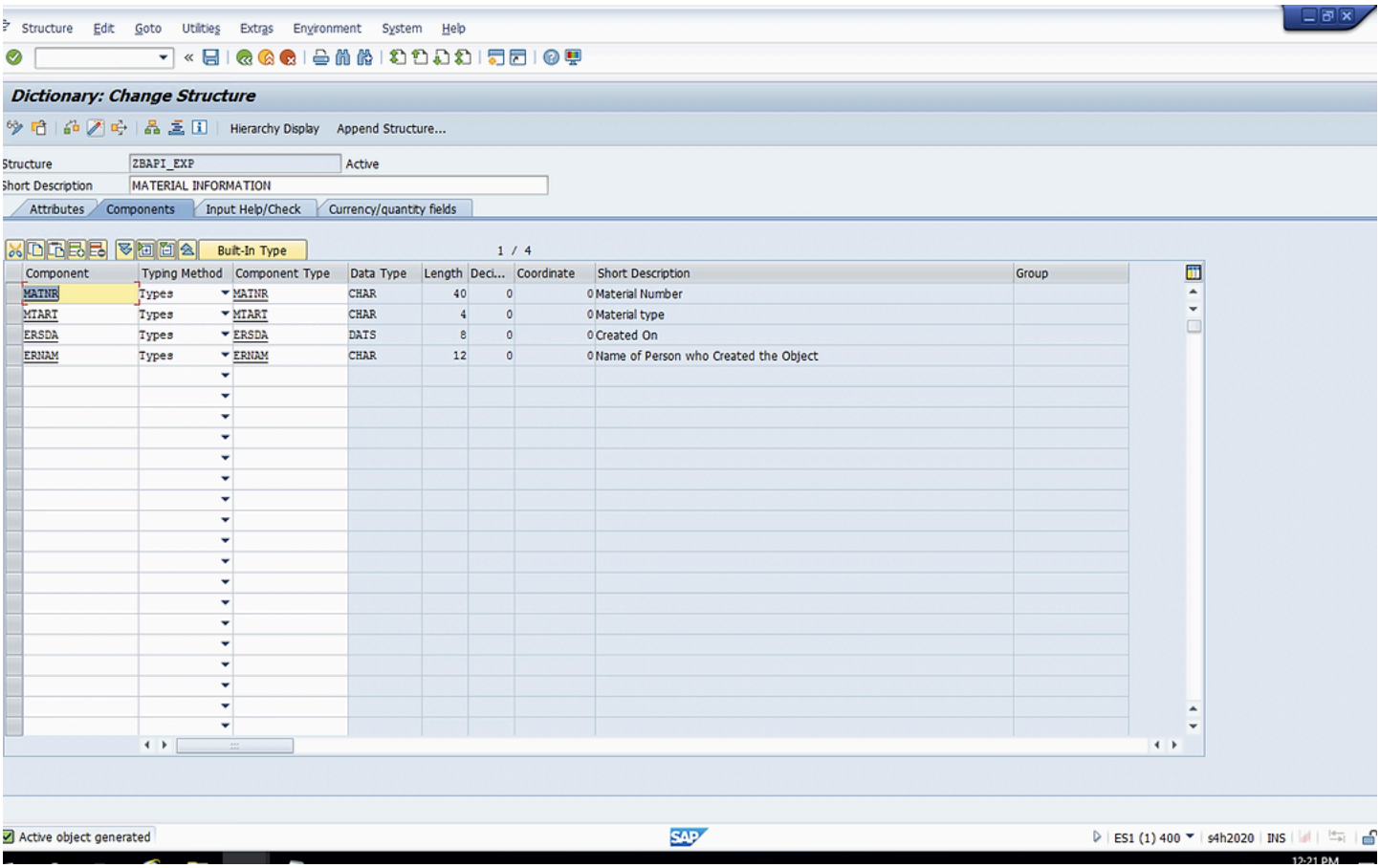
Enter required fields.



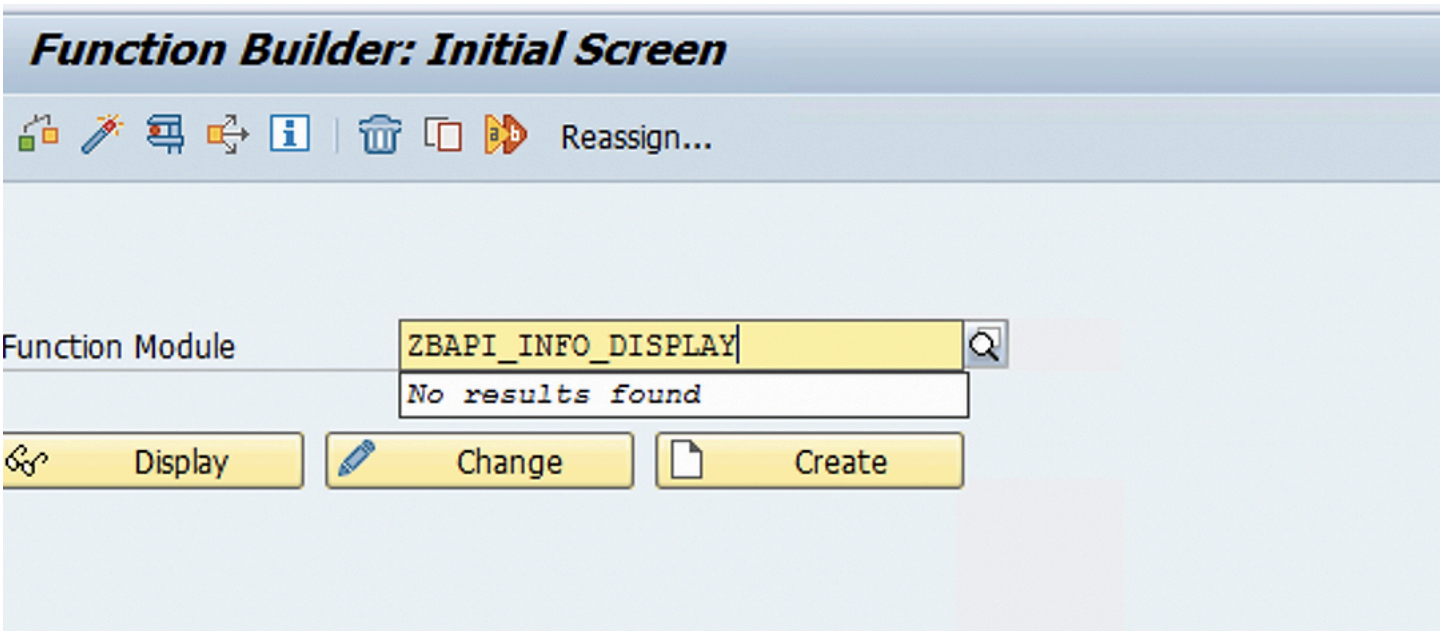
Save the structure in package.

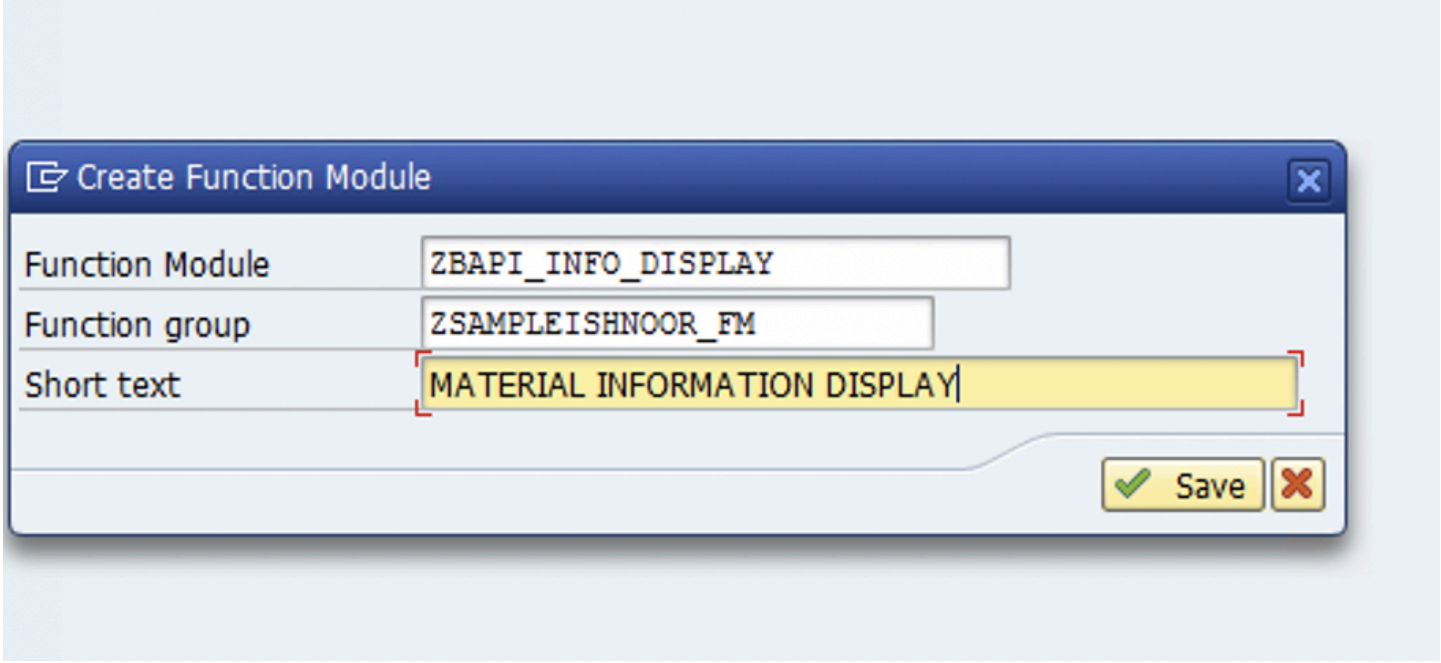


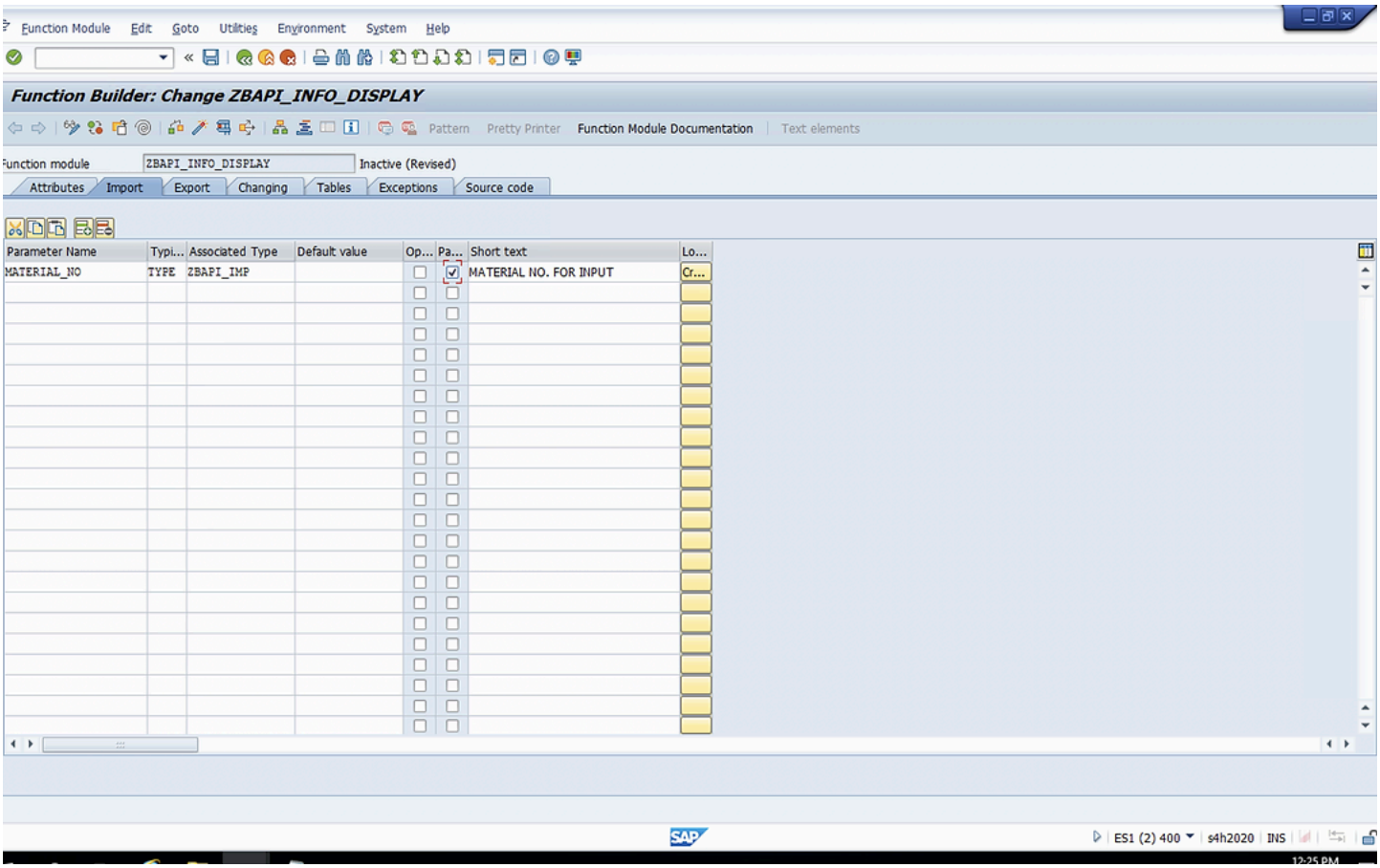
Do same for Exporting structure —

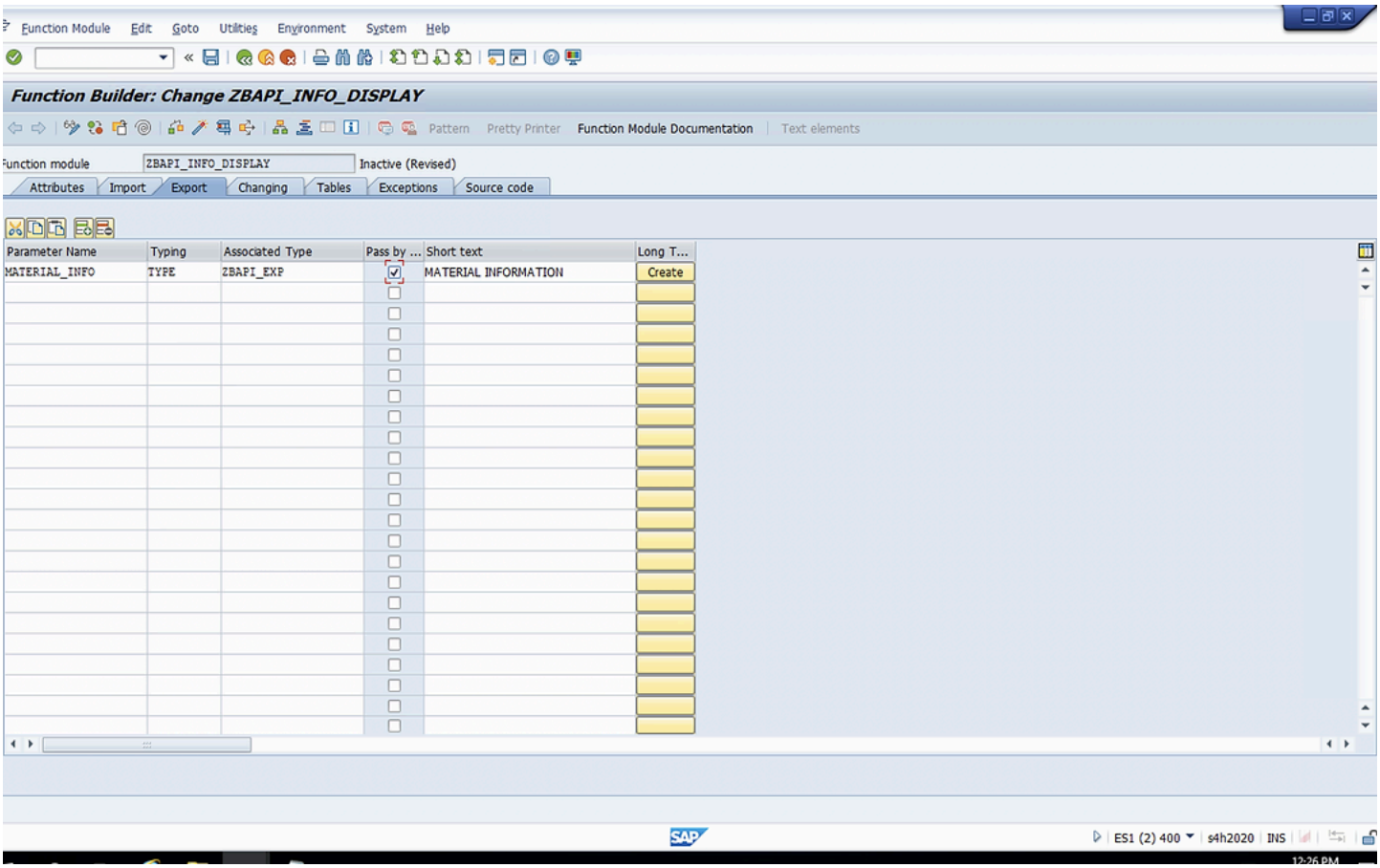


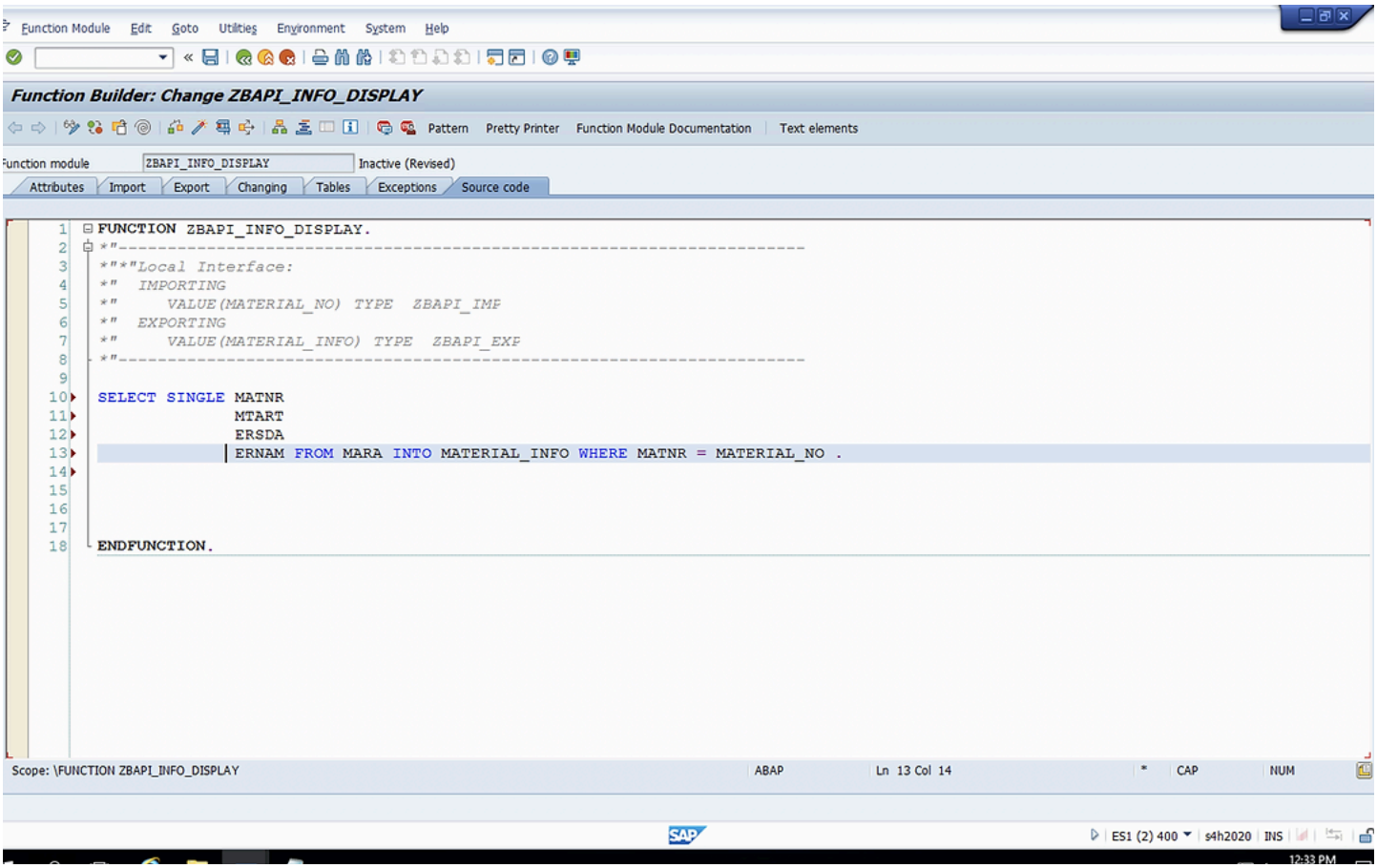
Step 2 - Create Function Module using import and export structure using tcode "SE37".




Create Function Module
Function Module
ZBAPI_INFO_DISPLAY
Function group
ZSAMPLEISHNOOR_FM
Short text
MATERIAL INFORMATION DISPLAY
Save

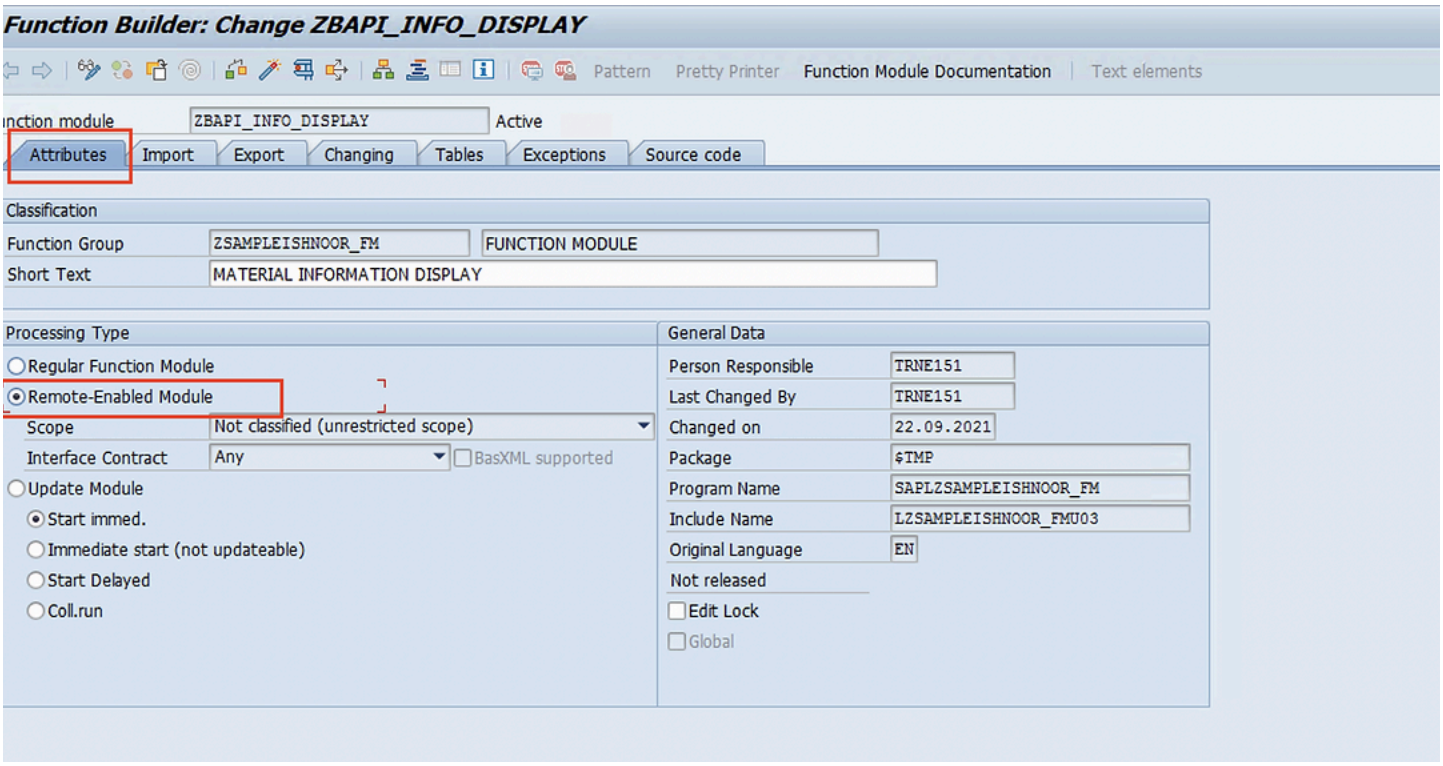

Function Module Edit Goto Utilities Environment System Help
Function Builder: Change ZBAPI_INFO_DISPLAY
Pattern Pretty Printer Function Module Documentation Text elements
Function module ZBAPI_INFO_DISPLAY Inactive (Revised)
Attributes Import Export Changing Tables Exceptions Source code
Parameter Name Typi... Associated Type Default value Op... Pa... Short text Lo...
MATERIAL_NO TYPE ZBAPI_IMP MATERIAL NO. FOR INPUT Cr...
ES1 (2) 400 s4h2020 INS



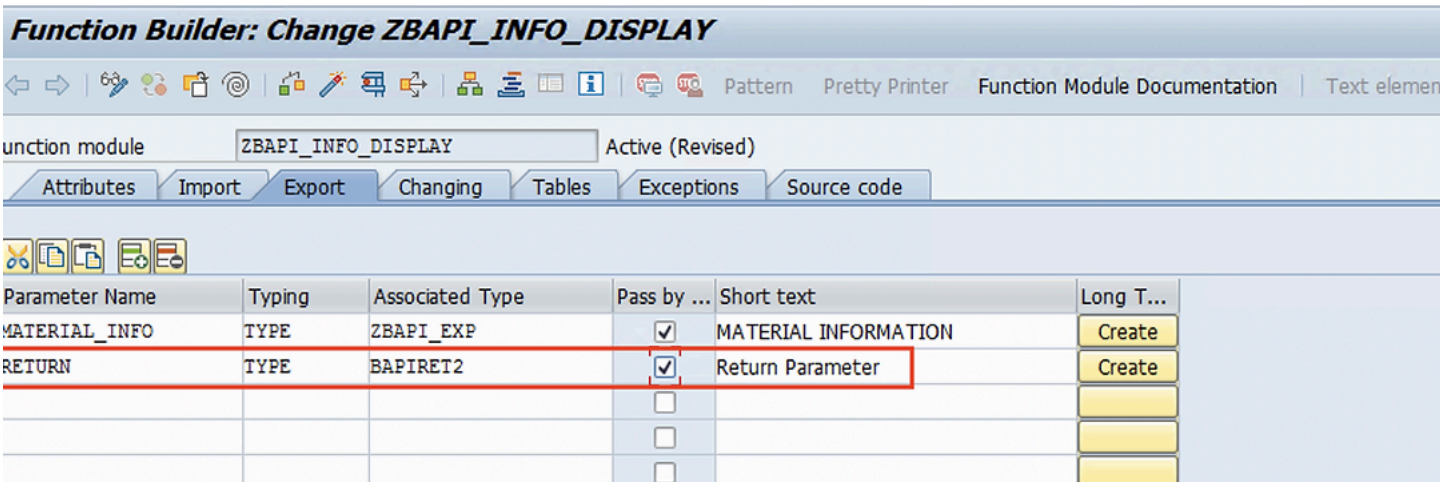


Step 3 - After creation of Function Module convert it into RFC and release that RFC.

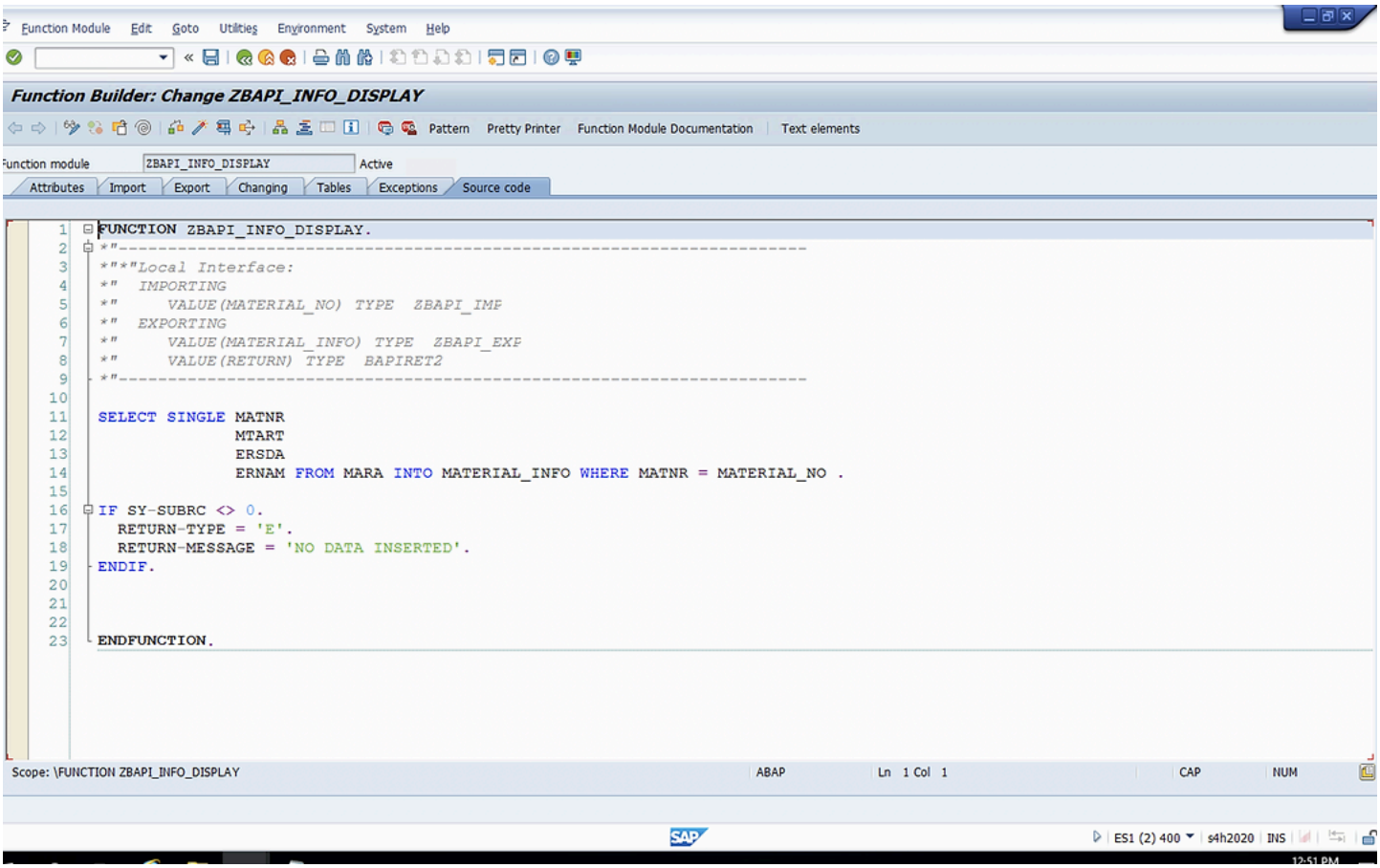
To convert FM into RFC click on attribute button and select Remote Enabled Module radio button



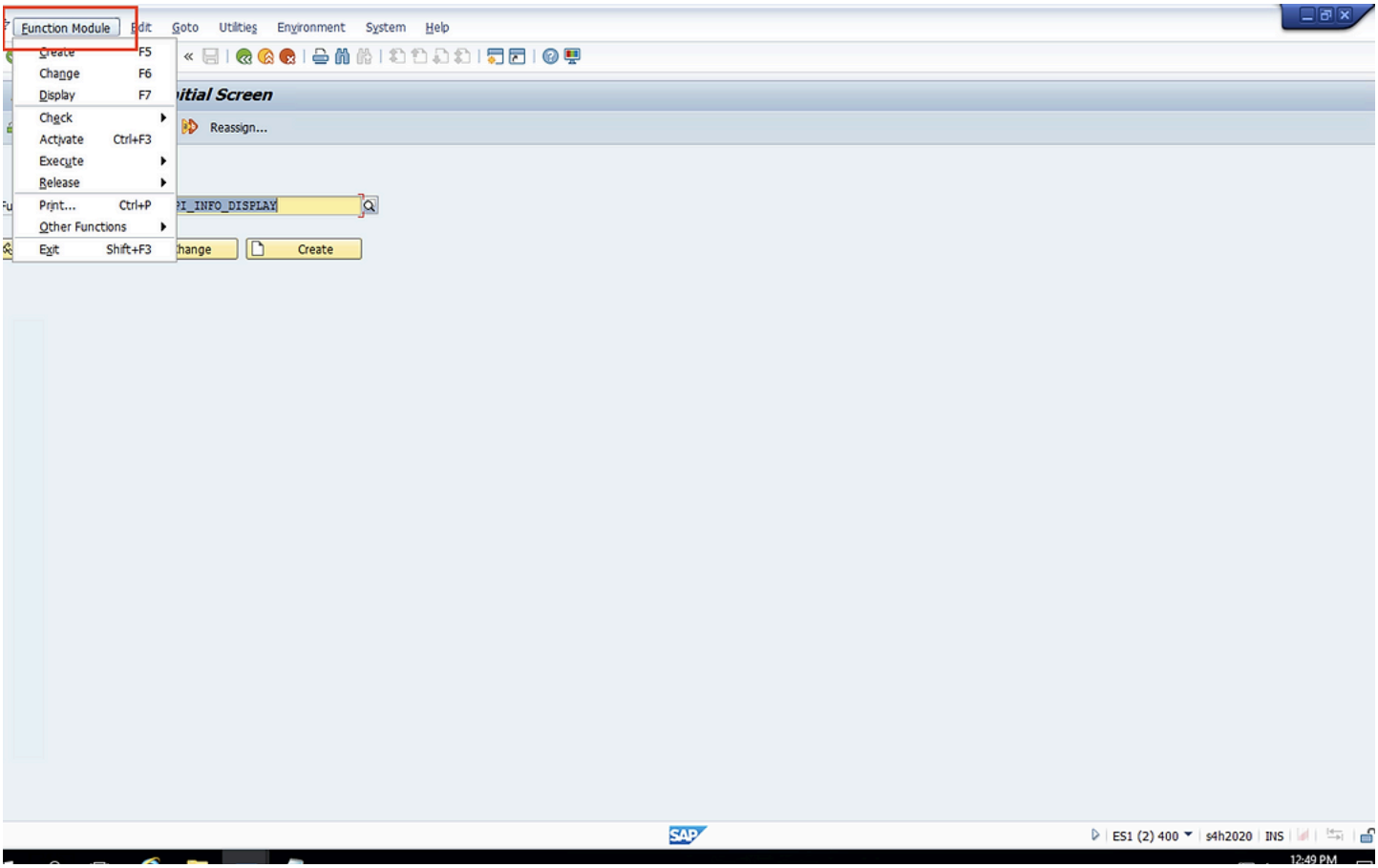
Enter a return type variable as BAPI has a return type



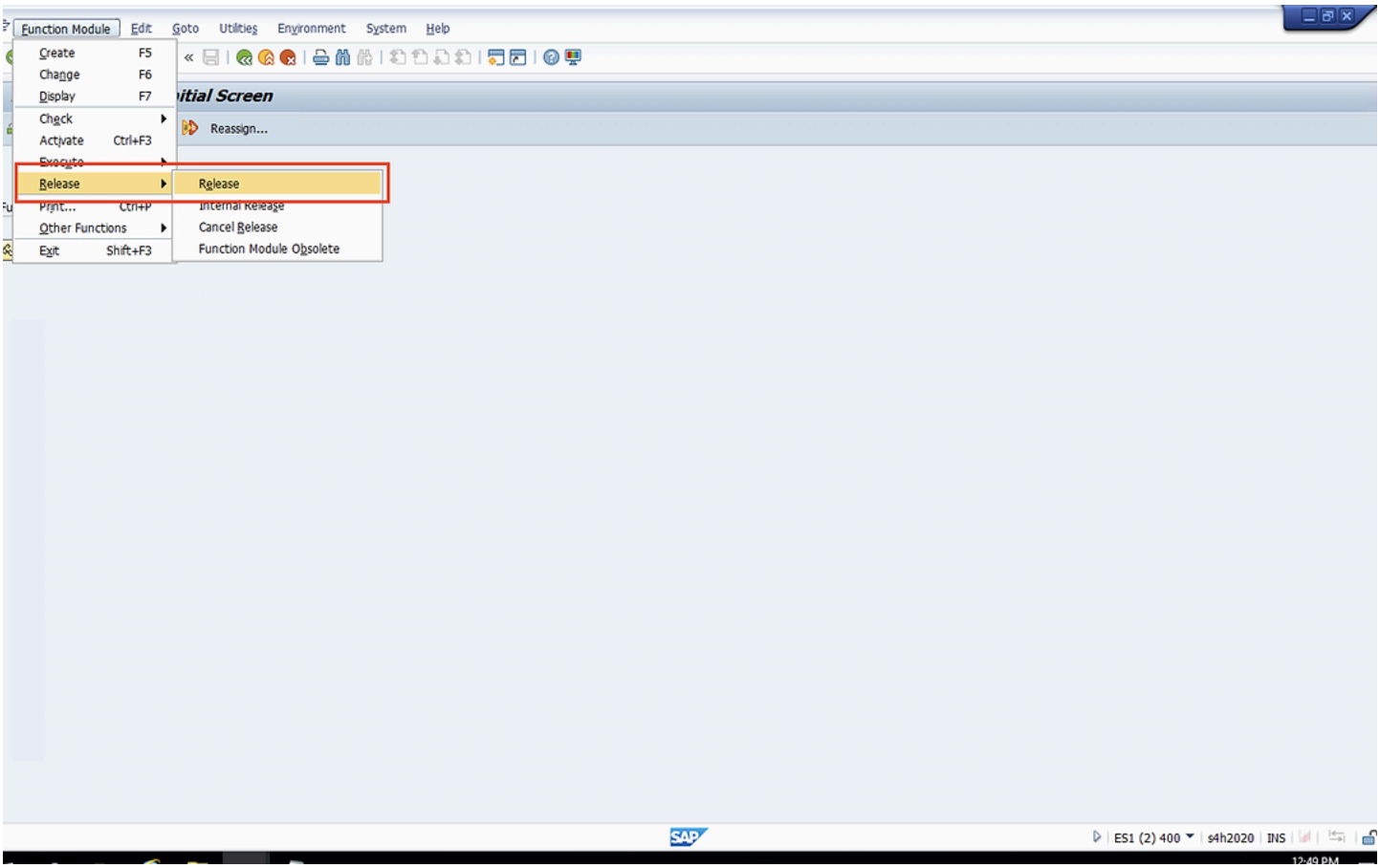
Source Code —



To release click on SE37 screen type the name of the FM you want to release and then click on “Function Module” option in the top menu.

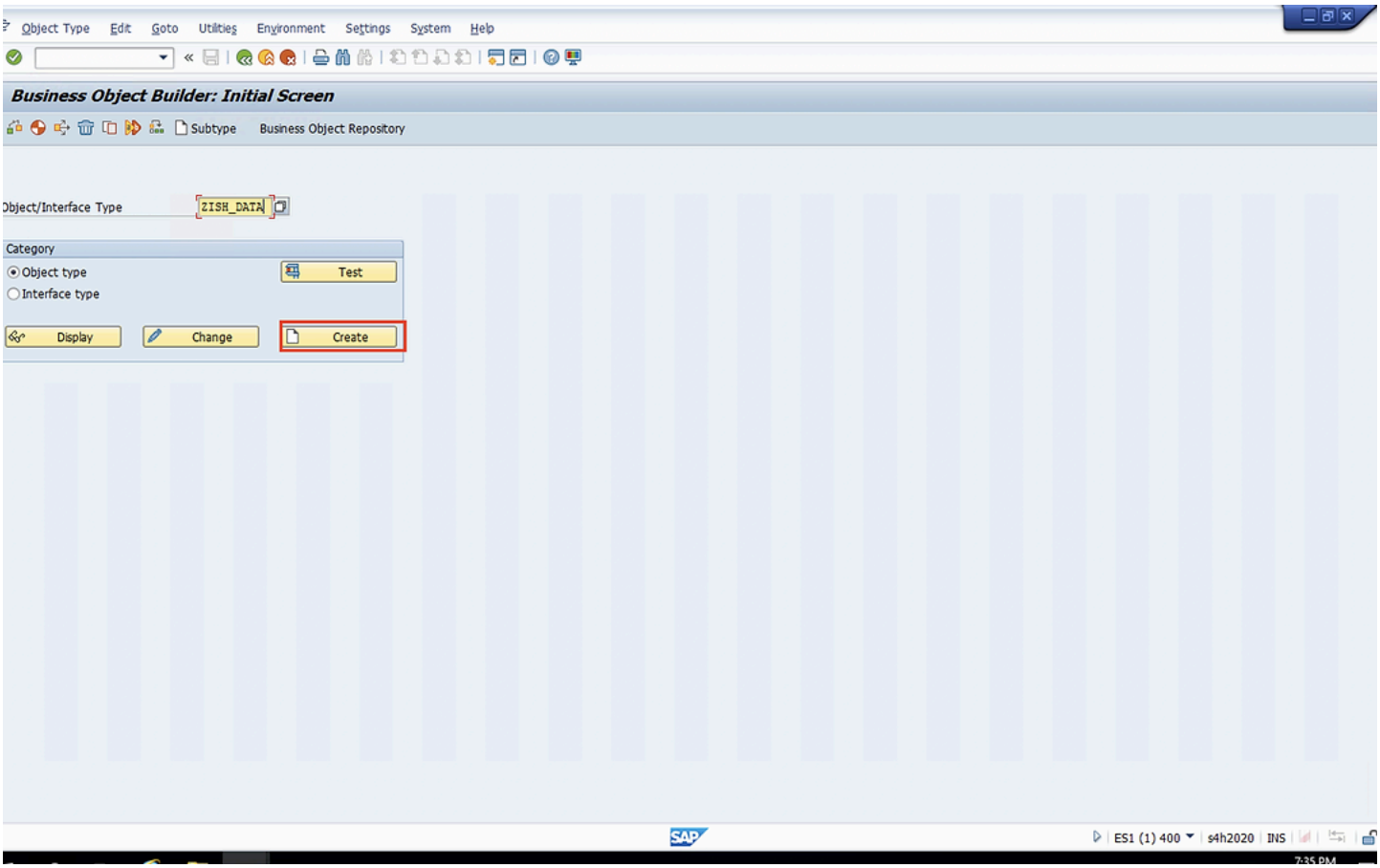


After that click on release and release.

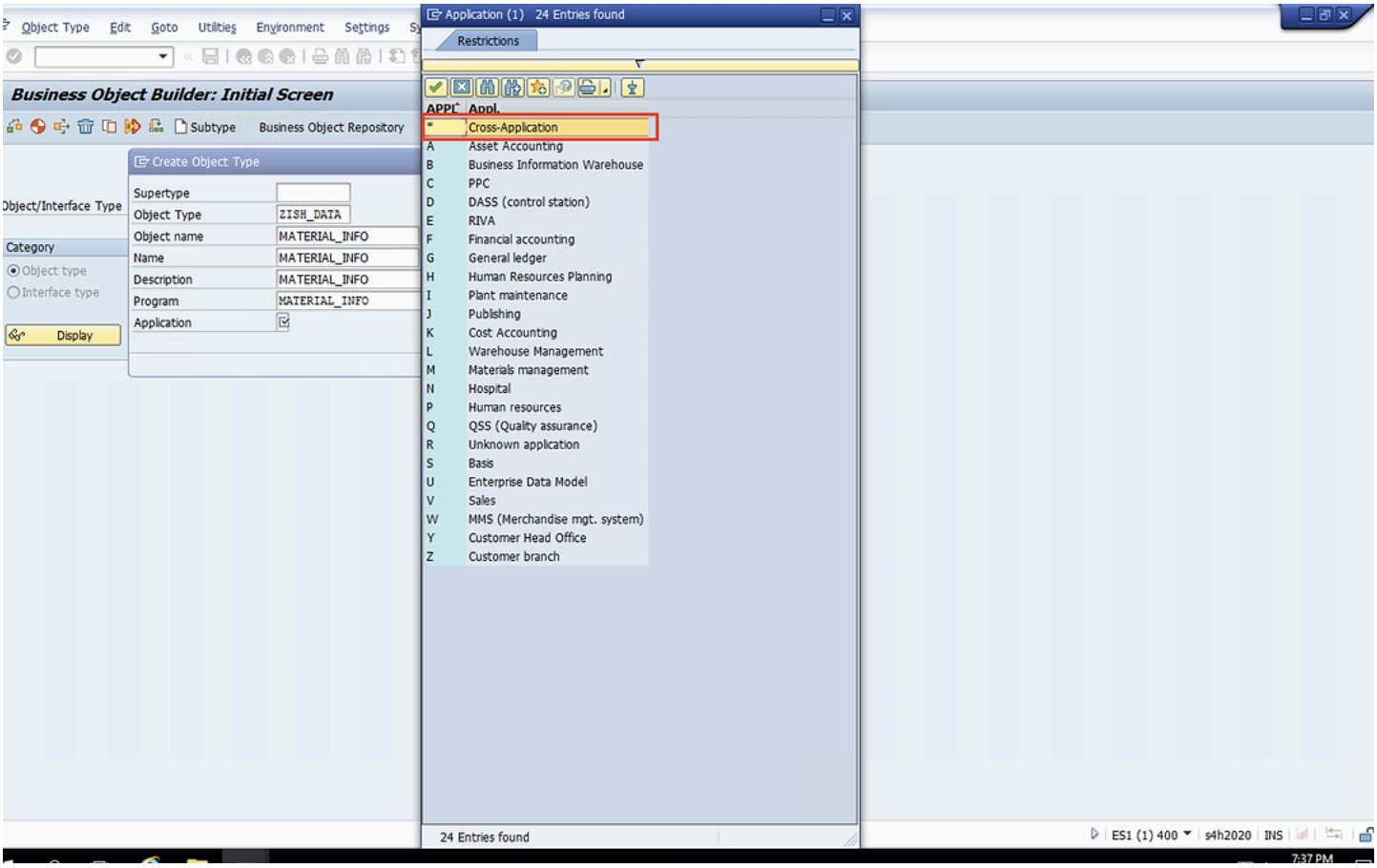


Step 4 - Create business object using tcode "SWO1".

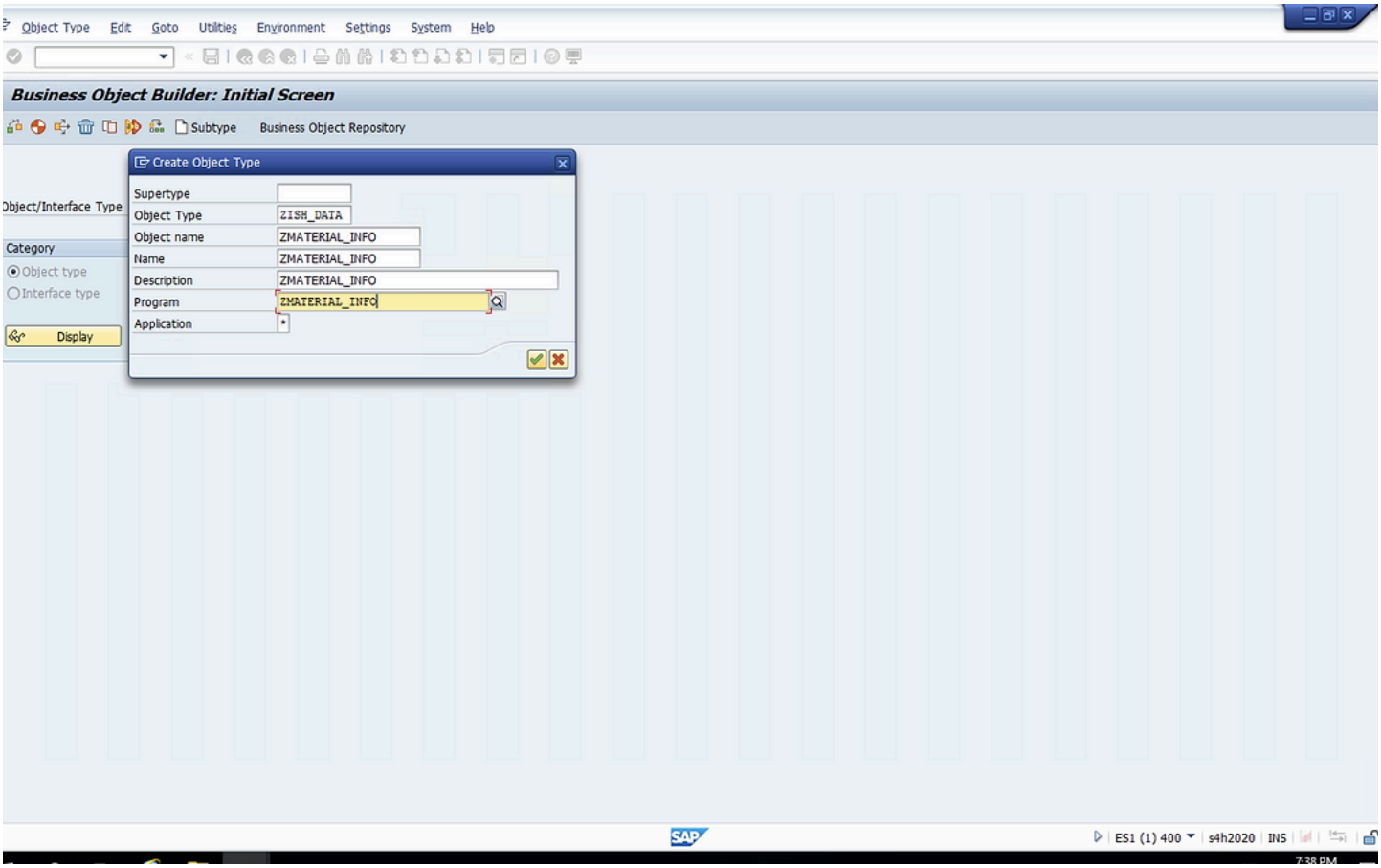
Write name of the business object and click on create.



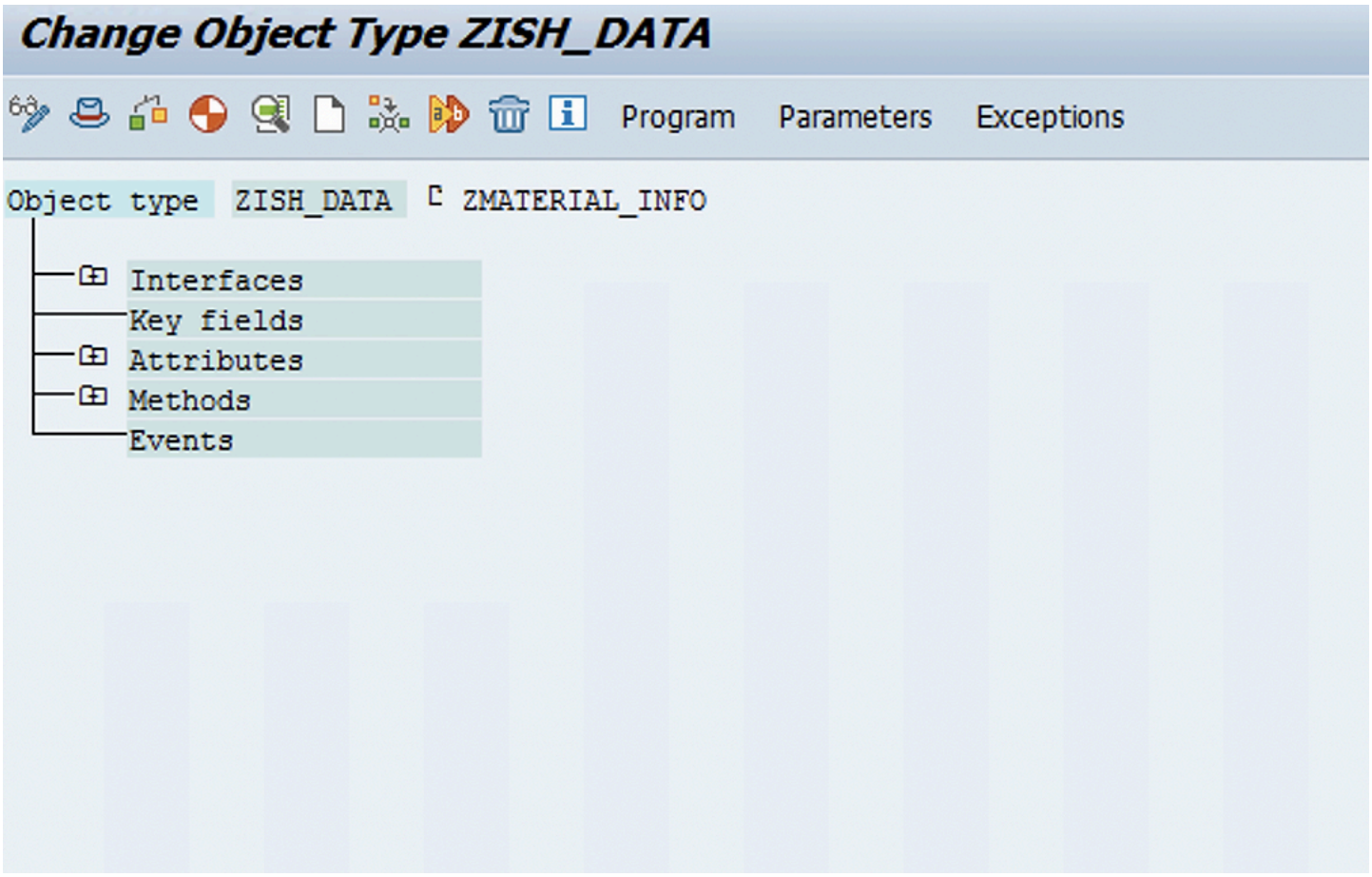
Fill the fields and for application field BAPI is always Cross-application



After filling click on the green tick and save it in package.

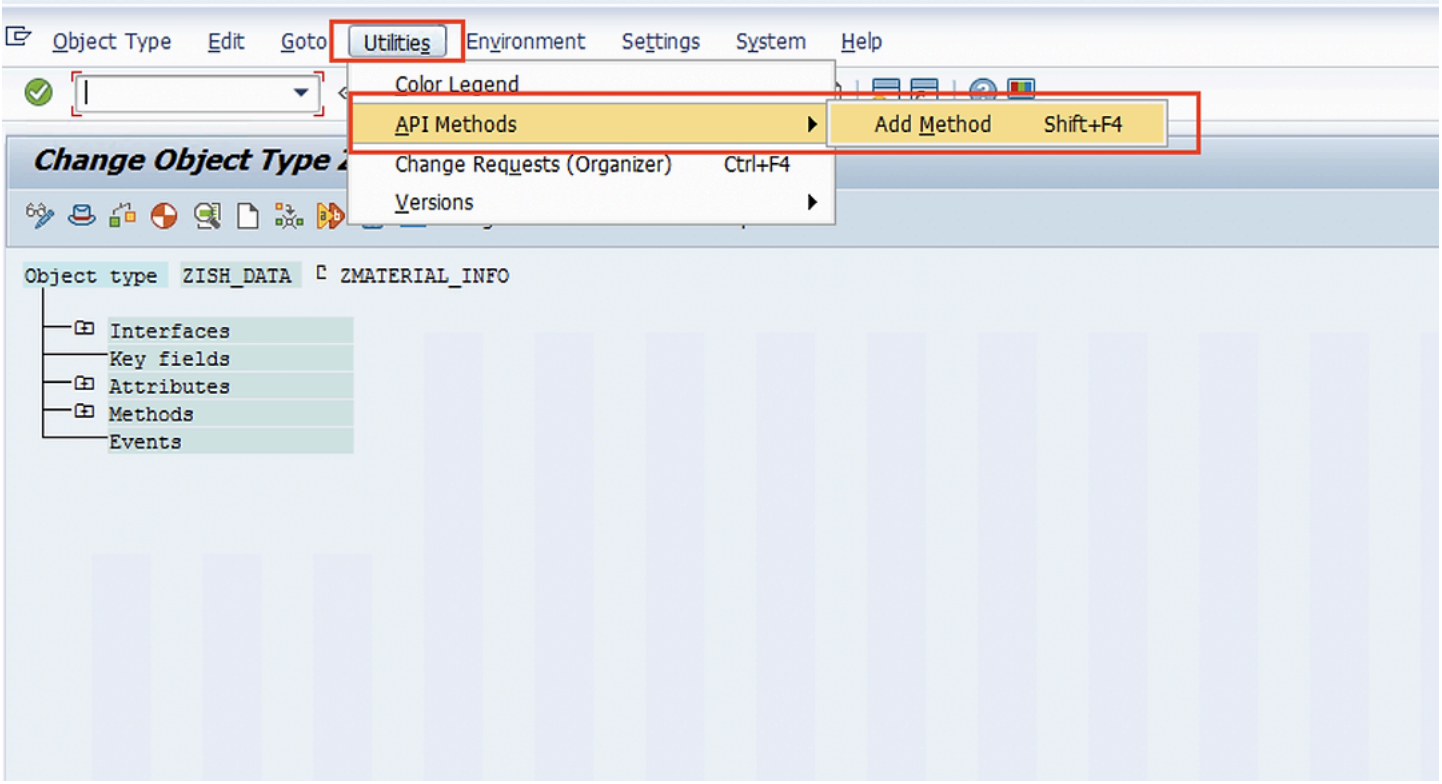


Business Object is formed

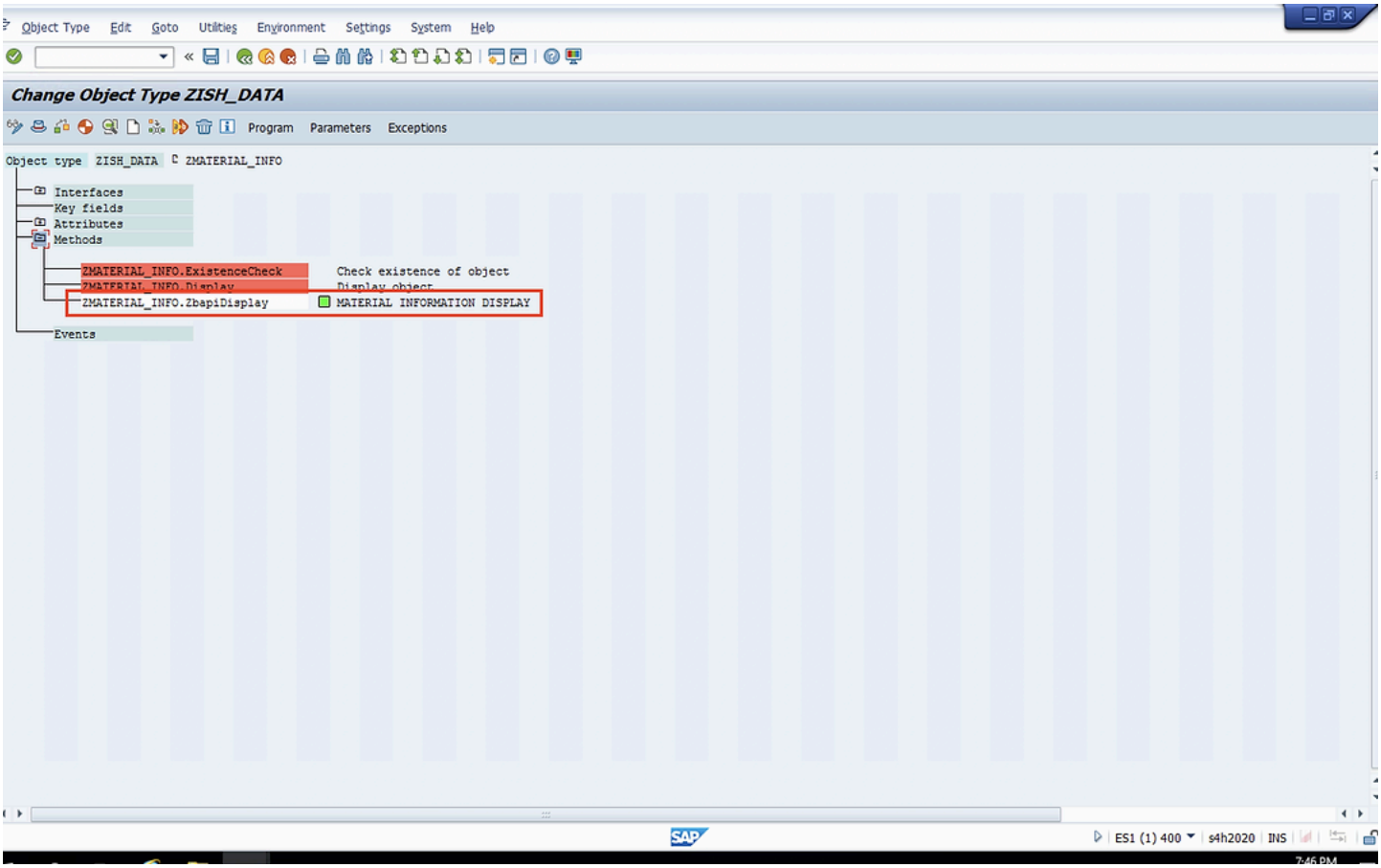


Step 5 - After BAPI creation we have to add RFC Function module into business object.

Click on utilities in the top menu then API methods and add methods.

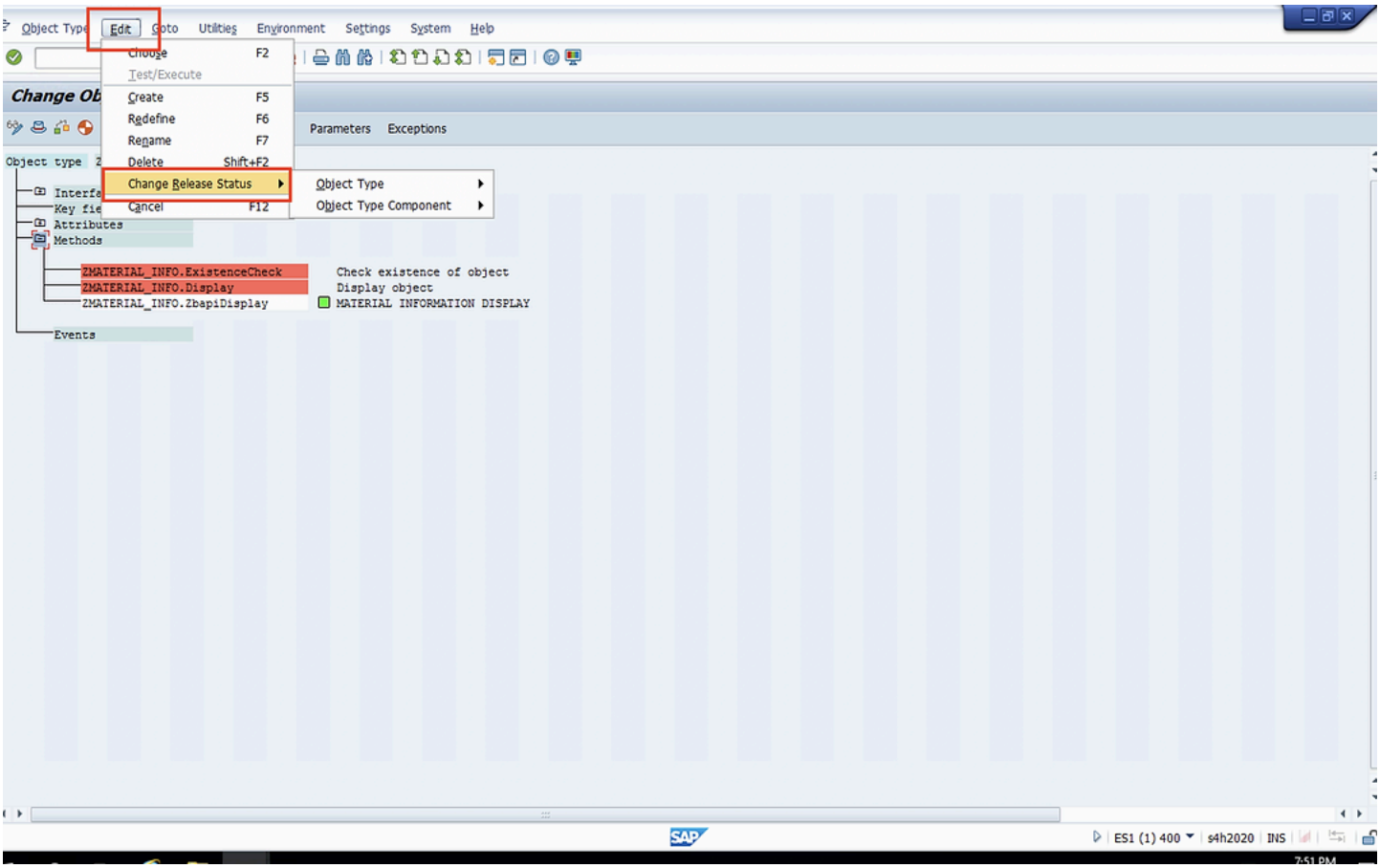


After this type the function module created for this program and click on enter. Function module will successfully be entered in the business object.

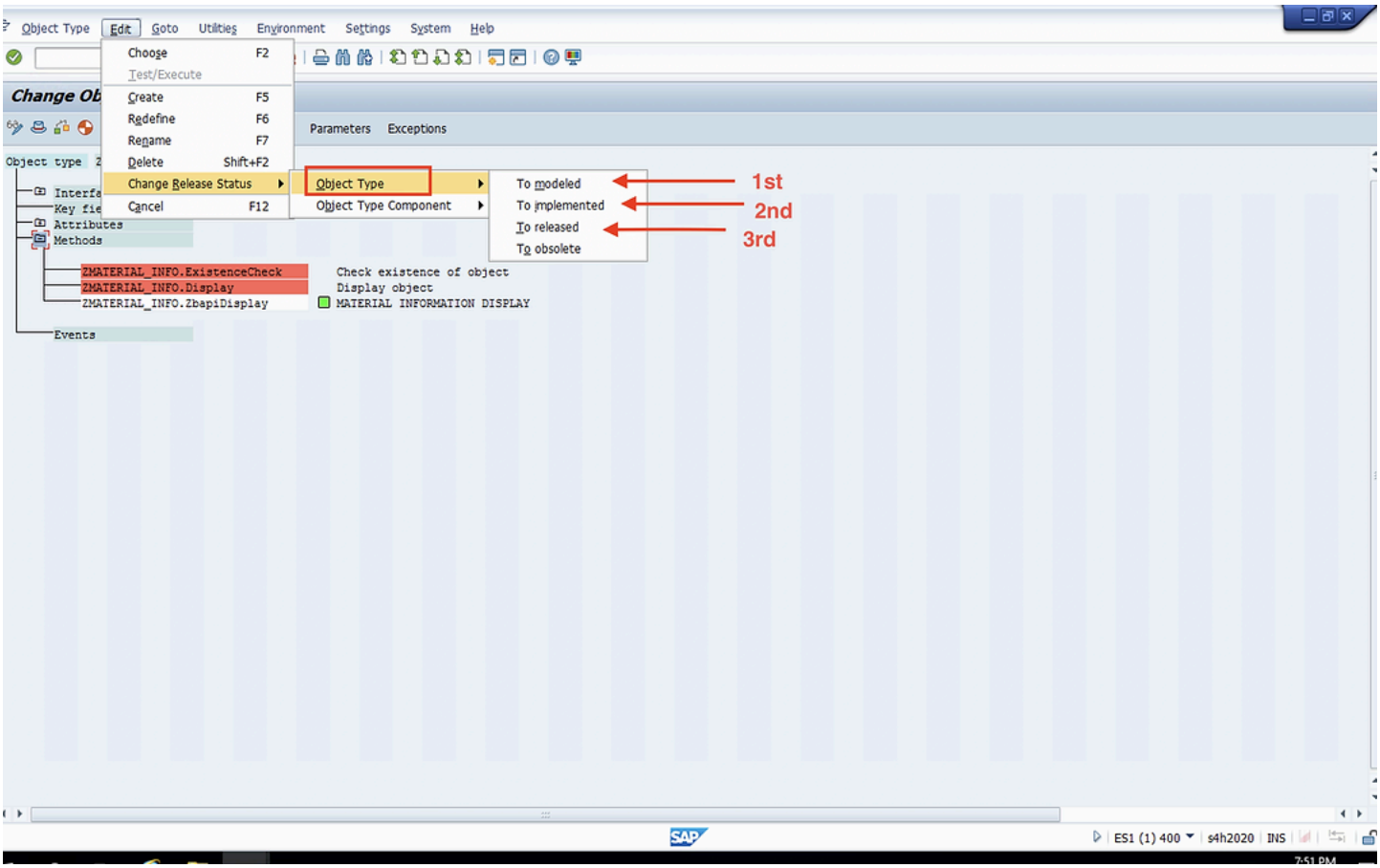


Step 6 - At last release business object components and component types and then finally release Business Object.

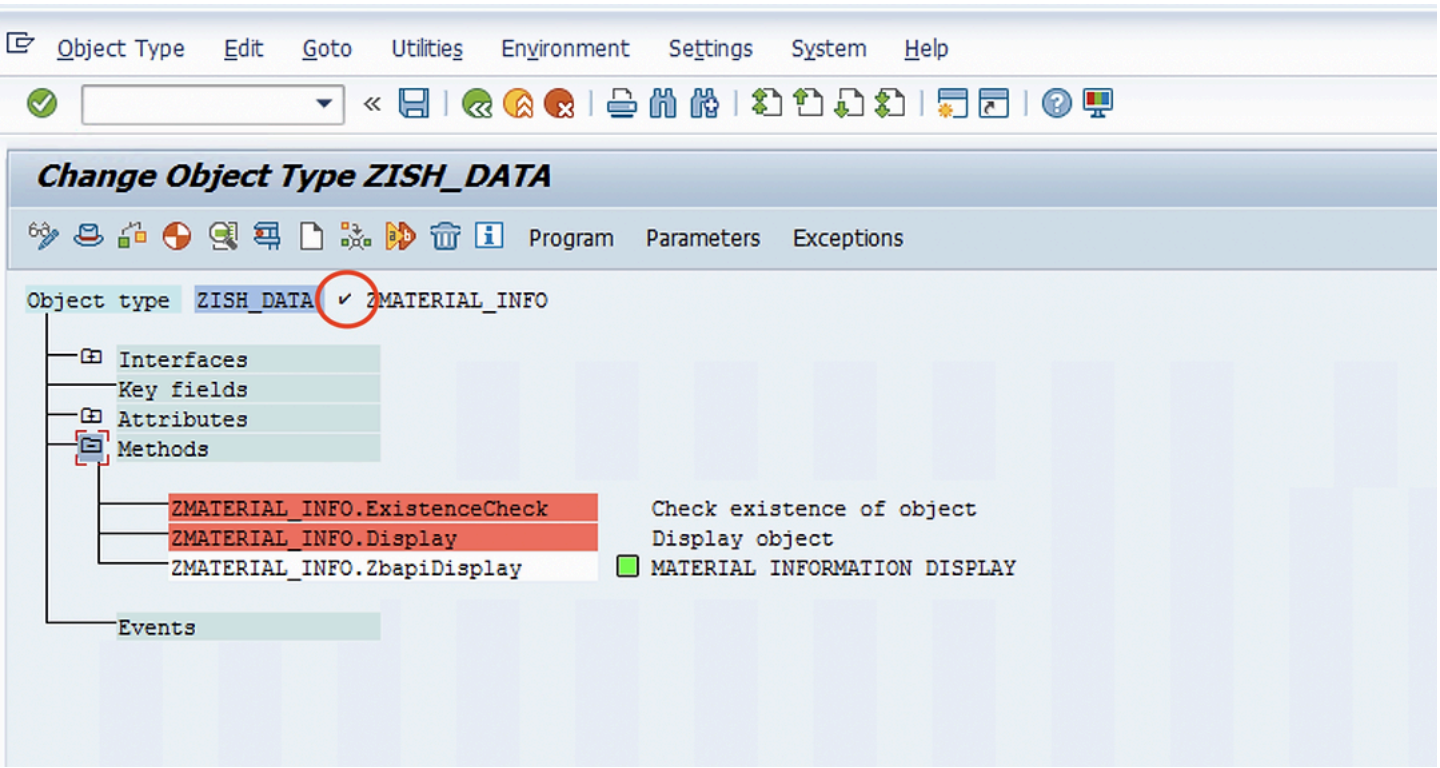
To release click on edit in the top menu then change release status. Now you will see two options. We have to release both object and object components.



Click on object type , 4 options will appear. In order to release first we have to click on 'To modeled' then 'To implemented' and last on 'To released'.

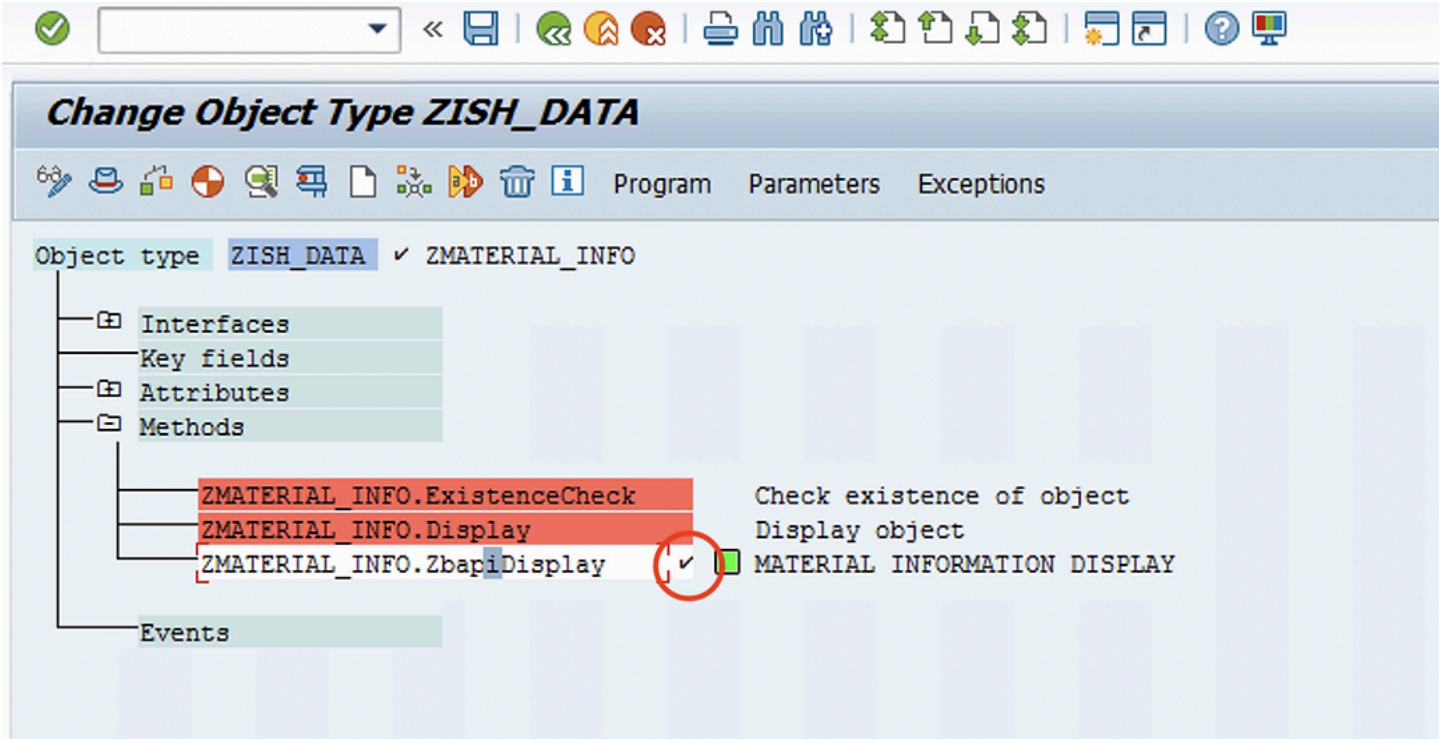


Tick symbol will ensure if the object type is successfully released or not.

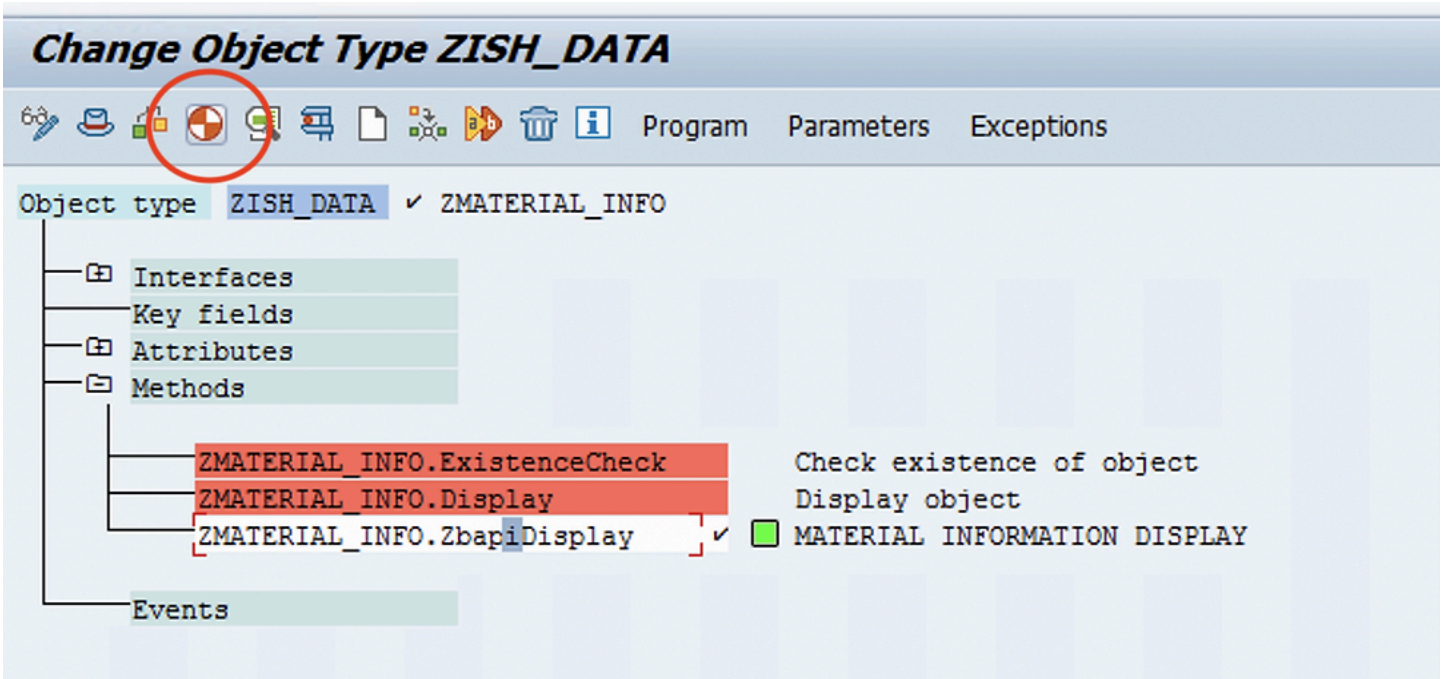


Likewise do the process for Object type component

Tick ensures if object type component is released or not



Last step is to click on generate (circled icon) and our custom BAPI is successfully created.



GET TRAINED BY EXPERTS-CHECK OUR TRAINING PROGRAM