

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 —

```

REPORT ZSAMPLE_ISHNOOR_BAPI_STAN.

TYPES : BEGIN OF TY_BNKA,
         BANKS TYPE BNKA-BANKS,
         BANKL TYPE BNKA-BANKL,
         BANKA TYPE BNKA-BANKA,
         PROVZ TYPE BNKA-PROVZ,
         STRAS TYPE BNKA-STRAS,
         ORTO1 TYPE BNKA-ORTO1,
         BRNCH TYPE BNKA-BRNCH,
      END OF TY_BNKA.

DATA : IT_BNKA TYPE STANDARD TABLE OF TY_BNKA,
      WA_BNKA TYPE TY_BNKA.

DATA : BANK_INFO TYPE BAPI1011_ADDRESS,
      RET1 TYPE BAPIRET2.

START-OF-SELECTION.
PERFORM GET_DATA.
PERFORM BAPI_PROCESS.

*<-----*
*& Form GET_DATA
*&-----*
*& text
*&-----*
*& --> p1      text
*& <-- p2      text
*&-----*

FORM get_data .
CALL FUNCTION 'GUI_UPLOAD'
  EXPORTING
    filename           = 'C:\Users\TRNE151\Desktop\BANK_INFO.txt'
    filetype          = 'ASC'
    has_field_separator = 'X'
    header_length     = 0
    read_by_line      = 'X'
    dat_mode          = ''
    codepage          = ''
    ignore_cerr       = ABAP_TRUE
    replacement        = '#'
    check_bom         = ''
    virus_scan_profile = ''
    no_auth_check     = ''
  IMPORTING
    filelength        =
    header            =
  tables
    data_tab          = IT_BNKA
  CHANGING
    isscanperformed   = ''
  EXCEPTIONS
    file_open_error    = 1
    file_read_error    = 2
    no_batch           = 3
    gui_refuse_filetransfer = 4
    invalid_type       = 5
    no_authority        = 6
    unknown_error       = 7
    bad_data_format     = 8
    header_not_allowed = 9
    separator_not_allowed = 10
    header_too_long     = 11
  
```

The screenshot shows the SAP ABAP Editor interface with the title bar "ABAP Editor: Change Report ZSAMPLE_ISHNOOR_BAPI_STAN". The code area contains ABAP code for a report named ZSAMPLE_ISHNOOR_BAPI_STAN. The code defines data types (TY_BNKA), declares data (IT_BNKA, WA_BNKA), and performs function modules (GET_DATA, BAPI_PROCESS). It also includes a FORM get_data with various parameters and exceptions related to file upload and BAPI processing.

```

67  * HEADER_TOO_LONG           = 11
68  * UNKNOWN_DP_ERROR         = 12
69  * ACCESS_DENIED            = 13
70  * DP_OUT_OF_MEMORY         = 14
71  * DISK_FULL                = 15
72  * DP_TIMEOUT               = 16
73  * OTHERS                   = 17
74
75  IF sy-subrc <> 0.
76  * Implement suitable error handling here
77  ENDIF.
78  ENDFORM.
79
80  *->
81  *& Form BAPI_PROCESS
82  *->
83  *& text
84  *&--> p1      text
85  *&--> p2      text
86  *&-->
87  FORM bapi_process .
88  LOOP AT IT_BNKA INTO WA_BNKA.
89  BANK_INFO-BANK_NAME = WA_BNKA-BANKA.
90  BANK_INFO-REGION = WA_BNKA-PROVZ.
91  BANK_INFO-STREET = WA_BNKA-STRAS.
92  BANK_INFO-CITY = WA_BNKA-ORT01.
93  BANK_INFO-BANK_BRANCH = WA_BNKA-BRNCH.
94
95  CALL FUNCTION 'BAPI_BANK_CREATE'
96    EXPORTING
97      bank_ctry          = WA_BNKA-BANKS
98      BANK_KEY            = WA_BNKA-BANKL
99      bank_address        = BANK_INFO
100     * BANK_METHOD         =
101     * BANK_FORMATTING    =
102     * BANK_ADDRESSI       =
103     * I_XUPDATE           = 'X'
104     * I_CHECK_BEFORE_SAVE =
105     * BANK_IBAN_RULE      =
106     * BANK_B2B_SUPPORTED   =
107     * BANK_CORI_SUPPORTED  =
108     * BANK_R_TRANSACTION_SUPPORTED =
109     * BANK_INTERNAL_BANK   =
110    IMPORTING
111      RETURN              = RET1
112      * BANKCOUNTRY        =
113      * BANKKEY             =
114
115  IF SY-SUBRC = 0.
116    WRITE : / 'BANK KEY ',WA_BNKA-BANKS, 'WAS CREATED'.
117    COMMIT WORK.
118  ELSE.
119    RET1-TYPE = 'E'.
120    RET1-MESSAGE = 'NO DATA INSERTED'.
121  ENDIF.
122 ENDOLOOP.
123 ENDFORM.

```

ABAP

Ln 25 Col 1

CAP

NUM



D | ES1 (1) 400 | s4h2020 | INS |

Key points in BAPI programs —

```

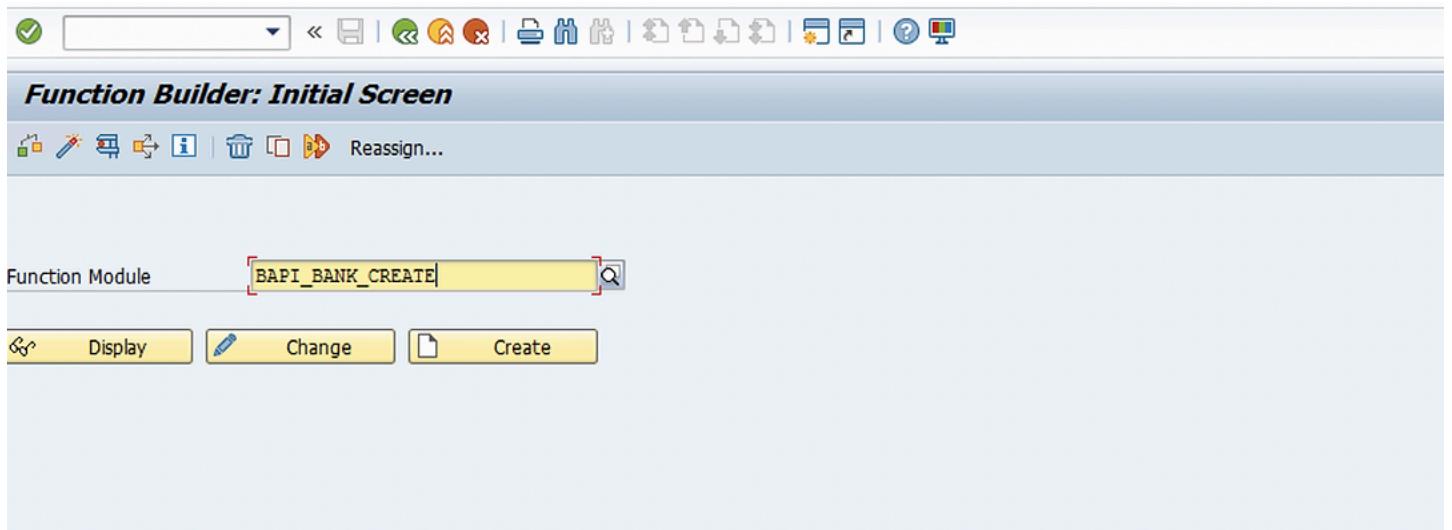
0
1  DATA : BANK_INFO TYPE BAPI1011_ADDRESS,
2    RET1 TYPE BAPIRET2.

```

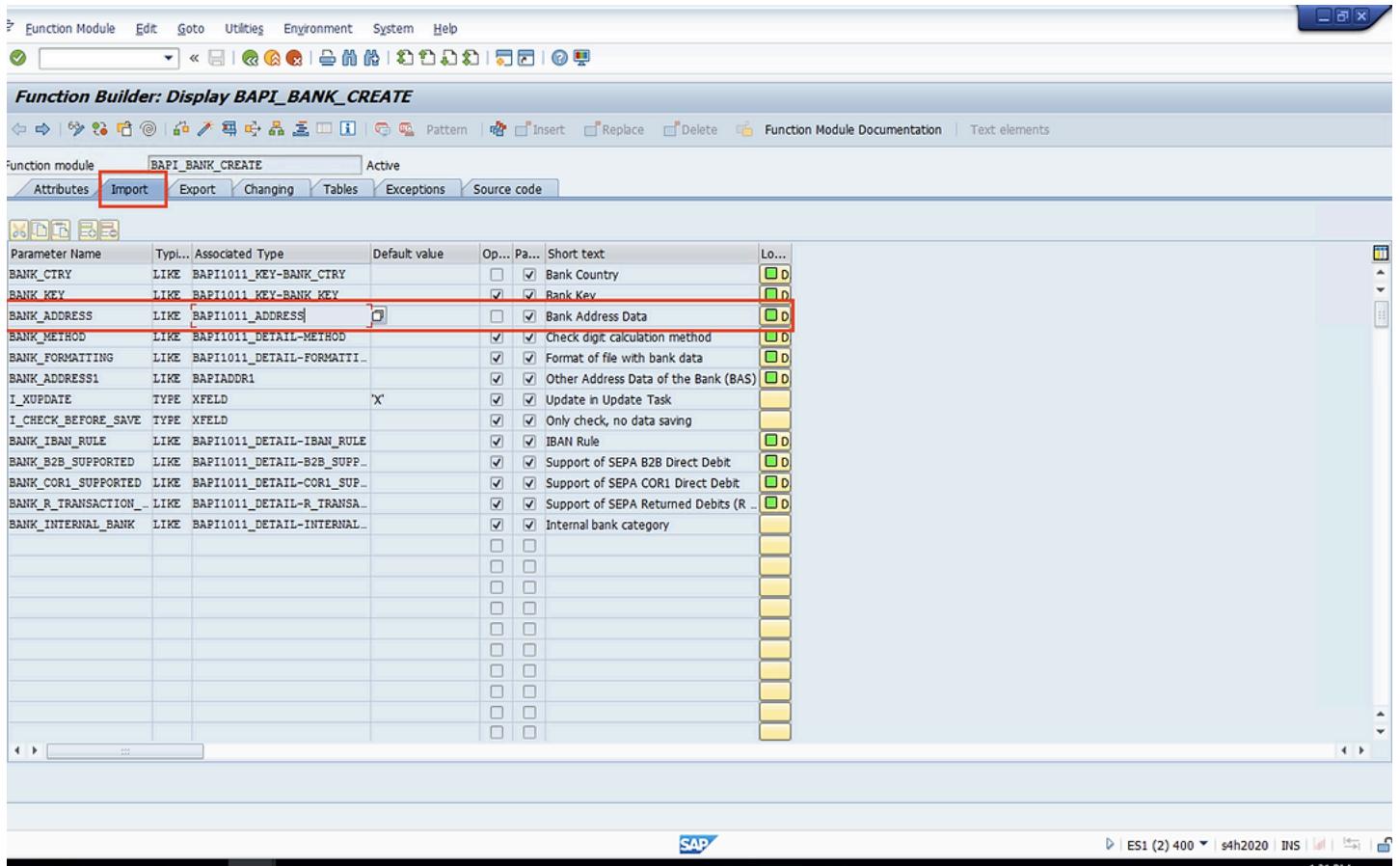
We got these two fields from the function that we called i.e “BAPI_BANK_CREATE”.

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.

The screenshot shows the SAP Dictionary: Display Structure interface. The title bar reads "Dictionary: Display Structure". The menu bar includes "Structure", "Edit", "Goto", "Utilities", "Extras", "Environment", "System", and "Help". Below the menu is a toolbar with various icons for navigating and managing structures.

The main area displays the structure details for "BAPI1011_ADDRESS" (Active). The short description is "Transfer structure object 1011: Bank address".

The structure is defined with the following components:

Component	Typing Method	Component Type	Data Type	Length	Deci...	Coordinate	Short Description	Group
BANK_NAME	Types	BANKA	CHAR	60	0		0 Name of bank	
REGION	Types	REGIO	CHAR	3	0		0 Region (State, Province, County)	
STREET	Types	STRAS_GP	CHAR	35	0		0 Street and House Number	
CITY	Types	ORT01_GP	CHAR	35	0		0 City	
SWIFT_CODE	Types	SWIFT	CHAR	11	0		0 SWIFT/BIC for International Payments	
BANK GROUP	Types	BGRUP	CHAR	2	0		0 Bank group (bank network)	
POBK_CURAC	Types	XFGRO	CHAR	1	0		0 Post Office Bank Current Account	
BANK NO	Types	BANKL	CHAR	15	0		0 Bank number	
POST_BANK	Types	PSKTO_CH	CHAR	16	0		0 Post office bank current account number	
BANK BRANCH	Types	BRNCH	CHAR	40	0		0 Bank Branch	
ADDR_NO	Types	AD_ADDRNUM	CHAR	10	0		0 Address Number	

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

Output —

STANDARD BAPI

STANDARD BAPI

```
BANK KEY IN WAS CREATED
BANK KEY IN WAS CREATED
BANK KEY IN WAS CREATED
```

Data Browser: Table BNKA Select Entries 3

Check Table...

Table: BNKA
Displayed Fields: 14 of 37 Fixed Columns: 3 List Width 0250

MANDT	BANKS	BANKL	ERDAT	ERNAM	BANKA	PROVZ	SIRAS	ORT01	SWI
400	IN	12273606	22.09.2021	TRNE151	IOB	01	JAMES STREET	BANGALORE	
400	IN	12472508	22.09.2021	TRNE151	ICICI	01	SWAMI STREET	CHENNAI	
400	IN	12913701	22.09.2021	TRNE151	SBI	01	RANGA STREET	HYDERABAD	

BANK_INFO - Notepad

File Edit Format View Help

IN	12913701	SBI	01	RANGA STREET	HYDERABAD	KPHB
IN	12472508	ICICI	01	SWAMI STREET	CHENNAI	KLPD
IN	12273606	IOB	01	JAMES STREET	BANGALORE	BTM2

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.

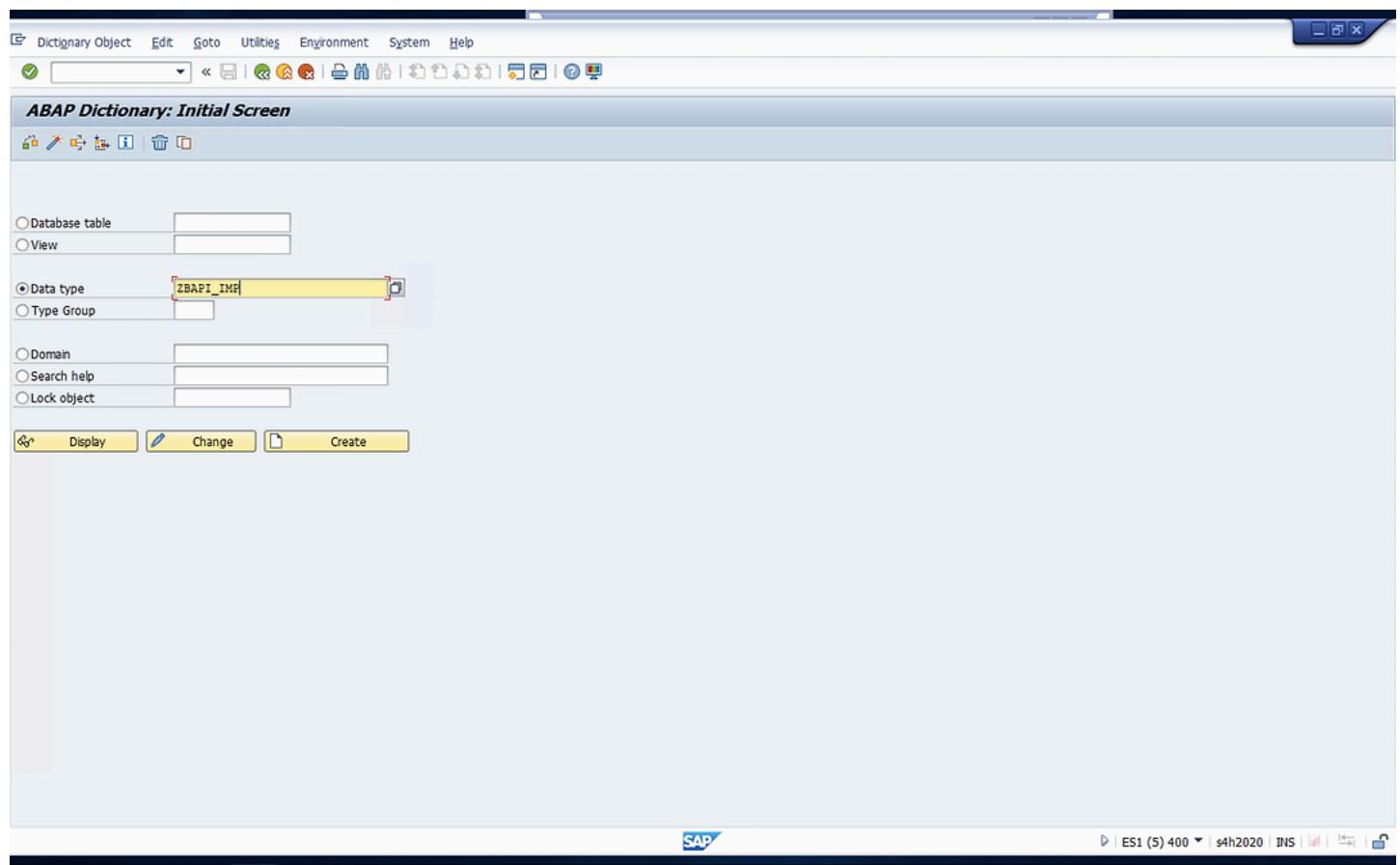
Step 4 - Create business object using tcode “SW01”.

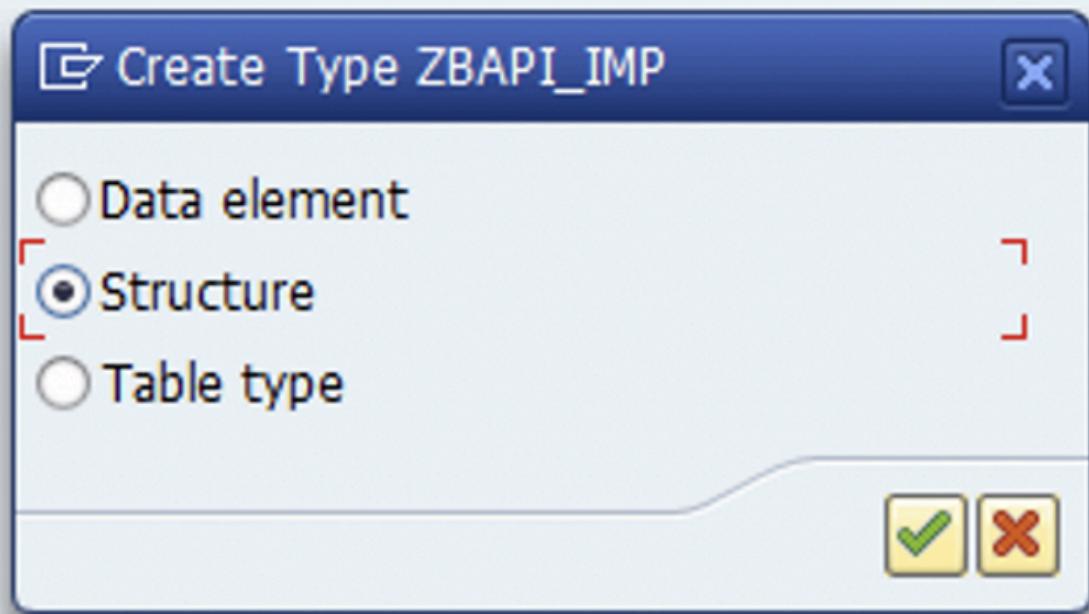
Step 5 - After BAPI creation we have to add RFC Function module into business object.

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 —



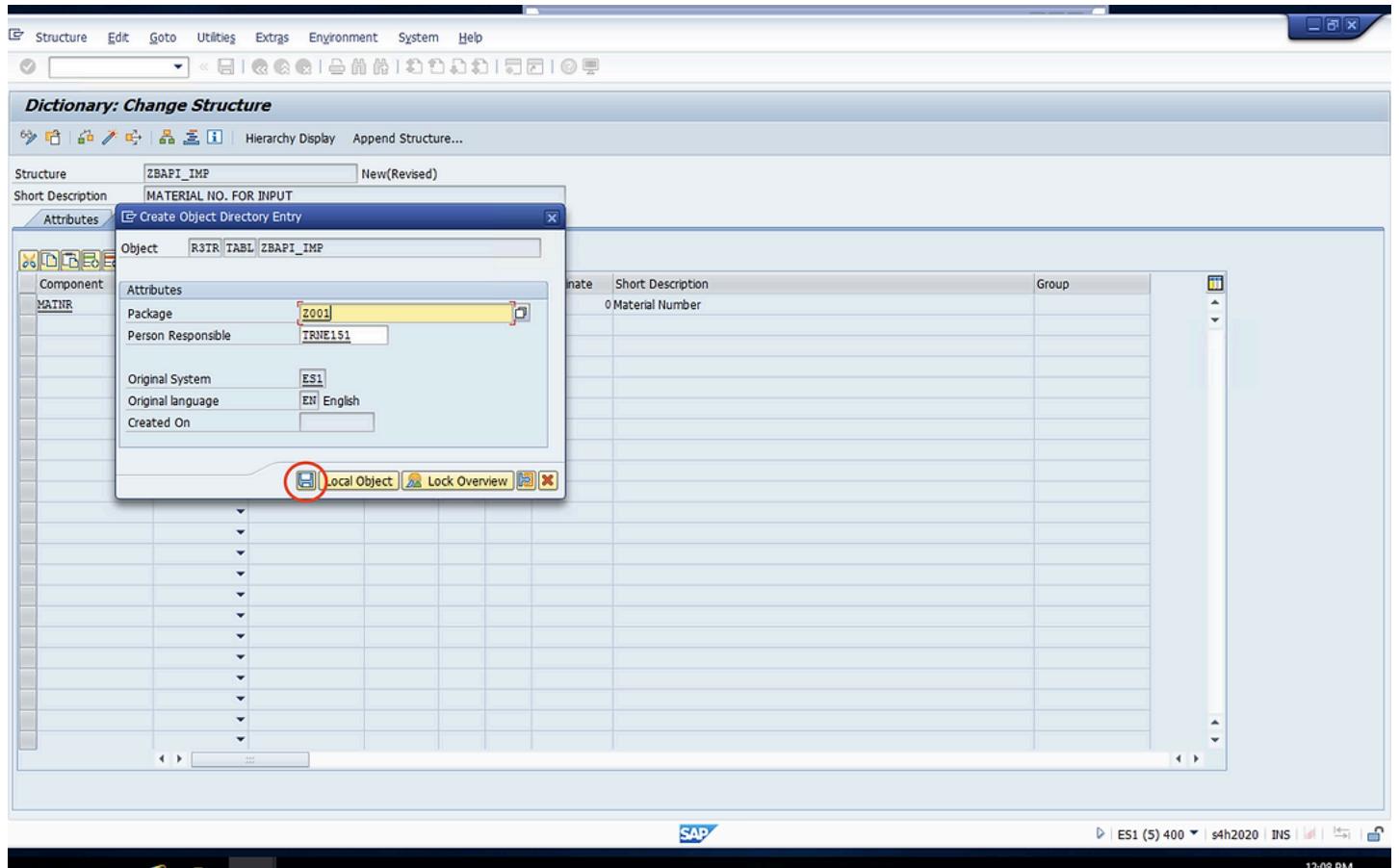


Enter required fields.

The screenshot shows the SAP Dictionary: Change Structure screen. The title bar says "Dictionary: Change Structure". The main area displays a table with the following data:

Component	Typing Method	Component Type	Data Type	Length	Deci...	Coordinate	Short Description	Group
MATNR		MATNR	CHAR	40	0		0 Material Number	

Save the structure in package.



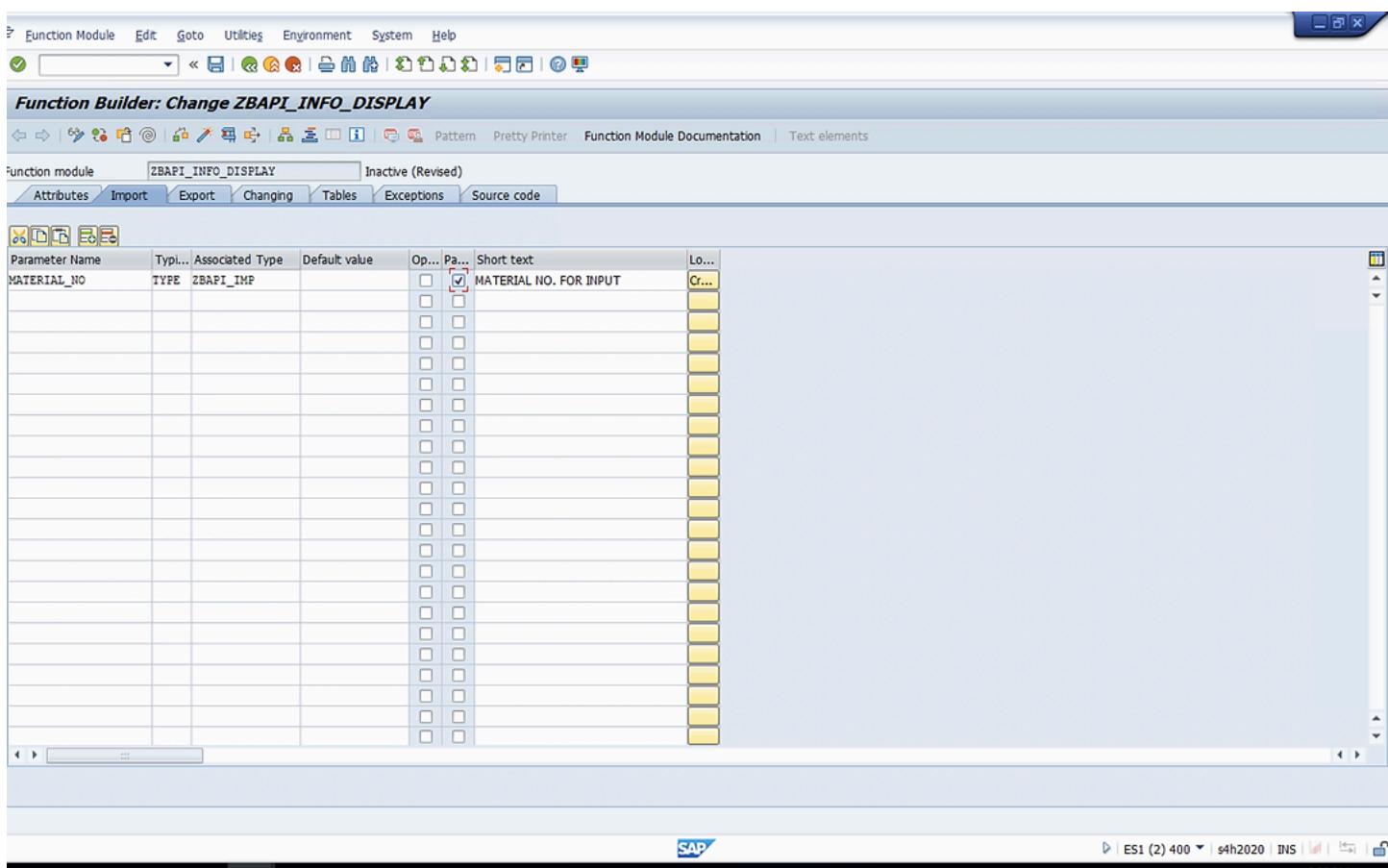
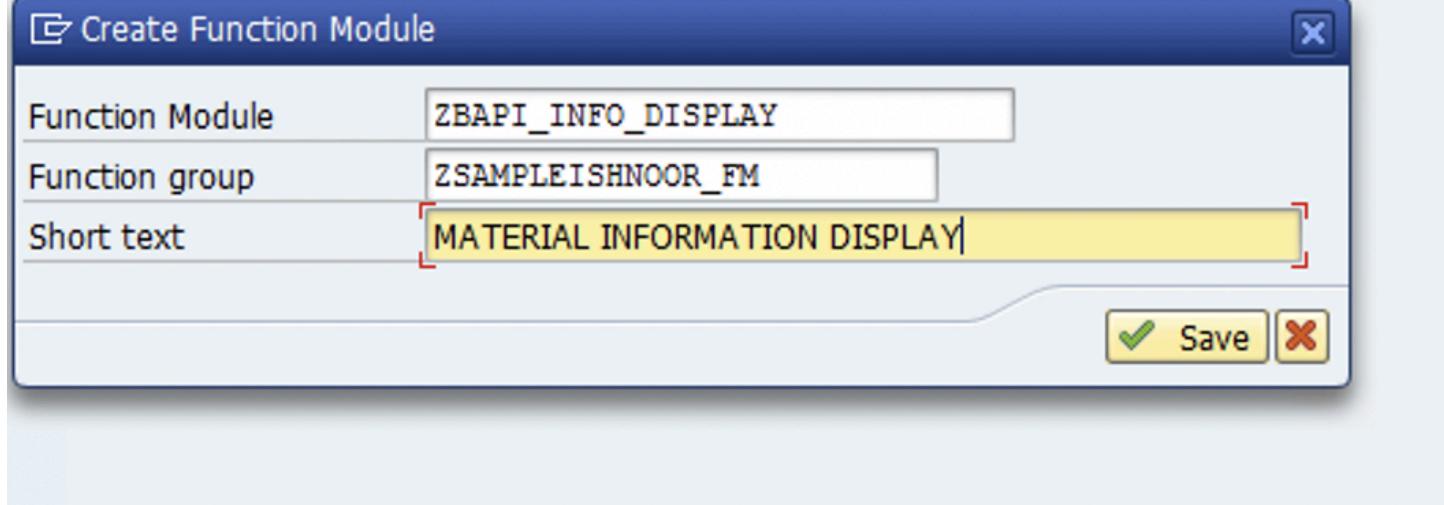
Do same for Exporting structure —

The screenshot shows the SAP Dictionary: Change Structure interface. The structure is named ZBAPI_EXP and is described as MATERIAL INFORMATION. It contains four components:

Component	Typing Method	Component Type	Data Type	Length	Decim...	Coordinate	Short Description	Group
MATNR	Types	MATNR	CHAR	40	0		Material Number	
MTART	Types	MTART	CHAR	4	0		Material type	
ERSDA	Types	ERSDA	DATS	8	0		Created On	
ERNAME	Types	ERNAME	CHAR	12	0		Name of Person who Created the Object	

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

The screenshot shows the Function Builder: Initial Screen. The function module name is ZBAPI_INFO_DISPLAY. The search bar indicates "No results found". Below the search bar are three buttons: Display, Change, and Create.



The screenshot shows the SAP Function Builder interface for the function module ZBAPI_INFO_DISPLAY. The 'Attributes' tab is selected. A table lists a single parameter: MATERIAL_INFO, which is of type TYPE and associated with ZBAPI_EXP. The 'Pass by ...' column has a checked checkbox next to 'Short text', and the 'Long T...' column contains the value 'Create'. The SAP logo is visible in the bottom right corner.

The screenshot shows the SAP Function Builder interface displaying the ABAP source code for the function module ZBAPI_INFO_DISPLAY. The code defines a function with a local interface, importing MATERIAL_NO and exporting MATERIAL_INFO. It then performs a SELECT statement to retrieve MATNR, MTART, and ERSDA from MARA, mapping them to MATERIAL_INFO where MATNR equals MATERIAL_NO. The code concludes with an ENDFUNCTION statement. The SAP logo is visible in the bottom right corner.

```

1  FUNCTION ZBAPI_INFO_DISPLAY.
2  *-- Local Interface:
3  *-- IMPORTING
4  *--   VALUE(MATERIAL_NO) TYPE ZBAPI_IMP
5  *-- EXPORTING
6  *--   VALUE(MATERIAL_INFO) TYPE ZBAPI_EXP
7  *--
8
9
10 SELECT SINGLE MATNR
11   MTART
12   ERSDA
13   | ERNAM FROM MARA INTO MATERIAL_INFO WHERE MATNR = MATERIAL_NO .
14
15
16
17
18 ENDFUNCTION.

```

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

Function Builder: Change ZBAPI_INFO_DISPLAY

Function module **ZBAPI_INFO_DISPLAY** Active

Attributes Import Export Changing Tables Exceptions Source code

Classification

Function Group **ZSAMPLEISHNOOR_FM** FUNCTION MODULE
Short Text MATERIAL INFORMATION DISPLAY

Processing Type

- Regular Function Module
- Remote-Enabled Module
- Update Module
- Start immed.
- Immediate start (not updateable)
- Start Delayed
- Coll.run

Scope Not classified (unrestricted scope)
Interface Contract Any BasXML supported

General Data

Person Responsible	TRNE151
Last Changed By	TRNE151
Changed on	22.09.2021
Package	\$TMP
Program Name	SAPLZSAMPLEISHNOOR_FM
Include Name	LZSAMPLEISHNOOR_FMU03
Original Language	EN
Not released	
<input type="checkbox"/> Edit Lock	
<input type="checkbox"/> Global	

Enter a return type variable as BAPI has a return type

Function Builder: Change ZBAPI_INFO_DISPLAY

Function module **ZBAPI_INFO_DISPLAY** Active (Revised)

Attributes Import Export Changing Tables Exceptions Source code

Parameters

Parameter Name	Typing	Associated Type	Pass by ...	Short text	Long T...
MATERIAL_INFO	TYPE	ZBAPI_EXP	<input checked="" type="checkbox"/>	MATERIAL INFORMATION	Create
RETURN	TYPE	BAPIRET2	<input checked="" type="checkbox"/>	Return Parameter	Create
			<input type="checkbox"/>		
			<input type="checkbox"/>		
			<input type="checkbox"/>		

Source Code —

The screenshot shows the SAP ABAP Function Builder interface. The title bar reads "Function Builder: Change ZBAPI_INFO_DISPLAY". The function module name is "ZBAPI_INFO_DISPLAY" and it is marked as "Active". The tabs at the top include "Attributes", "Import", "Export", "Changing", "Tables", "Exceptions", and "Source code". The "Source code" tab is selected, displaying the following ABAP code:

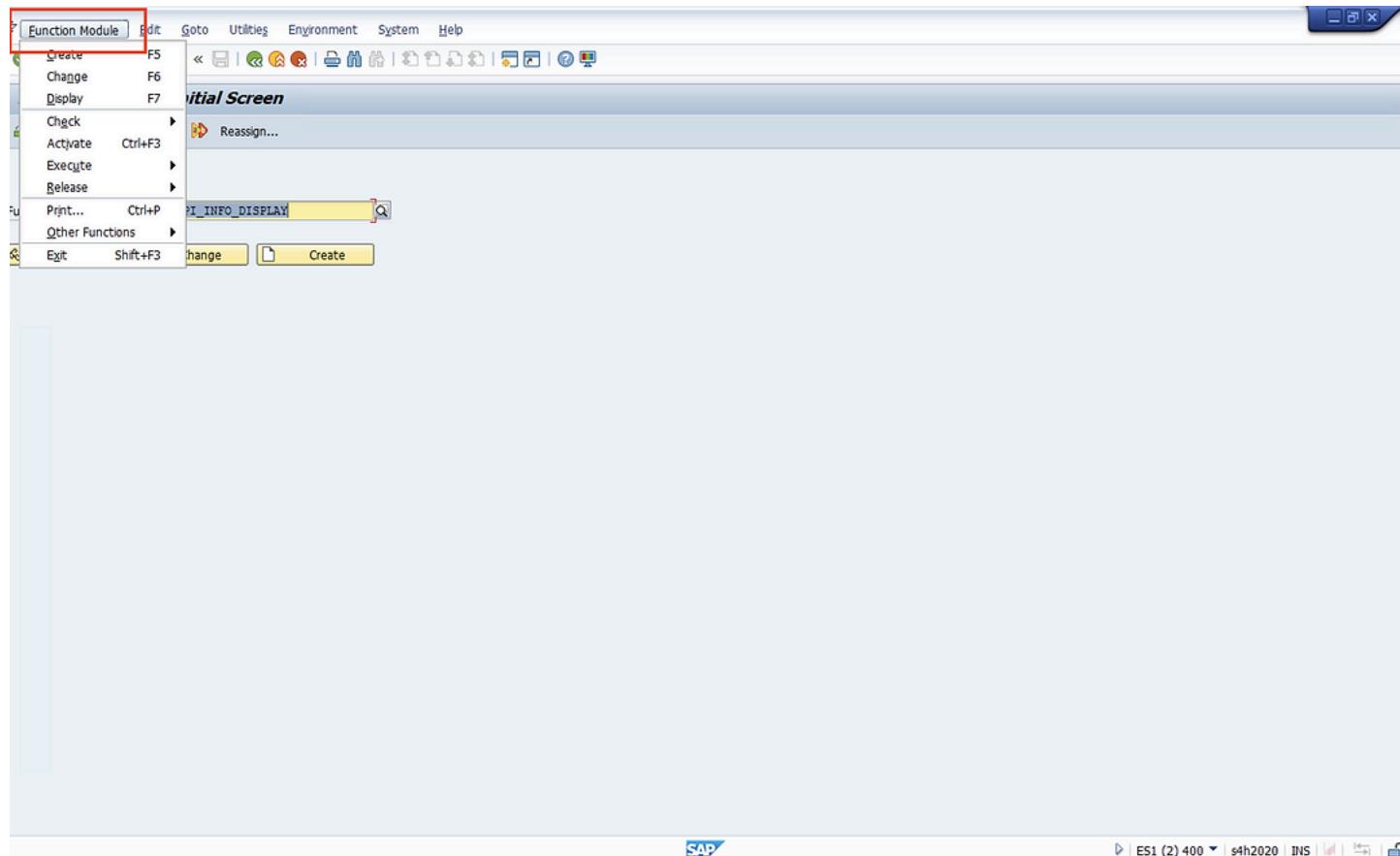
```

1 FUNCTION ZBAPI_INFO_DISPLAY.
2   *--> Local Interface:
3   *--> IMPORTING
4     MATERIAL_NO TYPE ZBAPI_IMP
5   *--> EXPORTING
6     MATERIAL_INFO TYPE ZBAPI_EXP
7   *--> RETURN TYPE BAPIRET2
8   *-->
9
10  SELECT SINGLE MATNR
11    MTART
12    ERSDA
13    ERNAM FROM MARA INTO MATERIAL_INFO WHERE MATNR = MATERIAL_NO .
14
15  IF SY-SUBRC <> 0.
16    RETURN-TYPE = 'E'.
17    RETURN-MESSAGE = 'NO DATA INSERTED'.
18  ENDIF.
19
20
21
22
23 ENDFUNCTION.

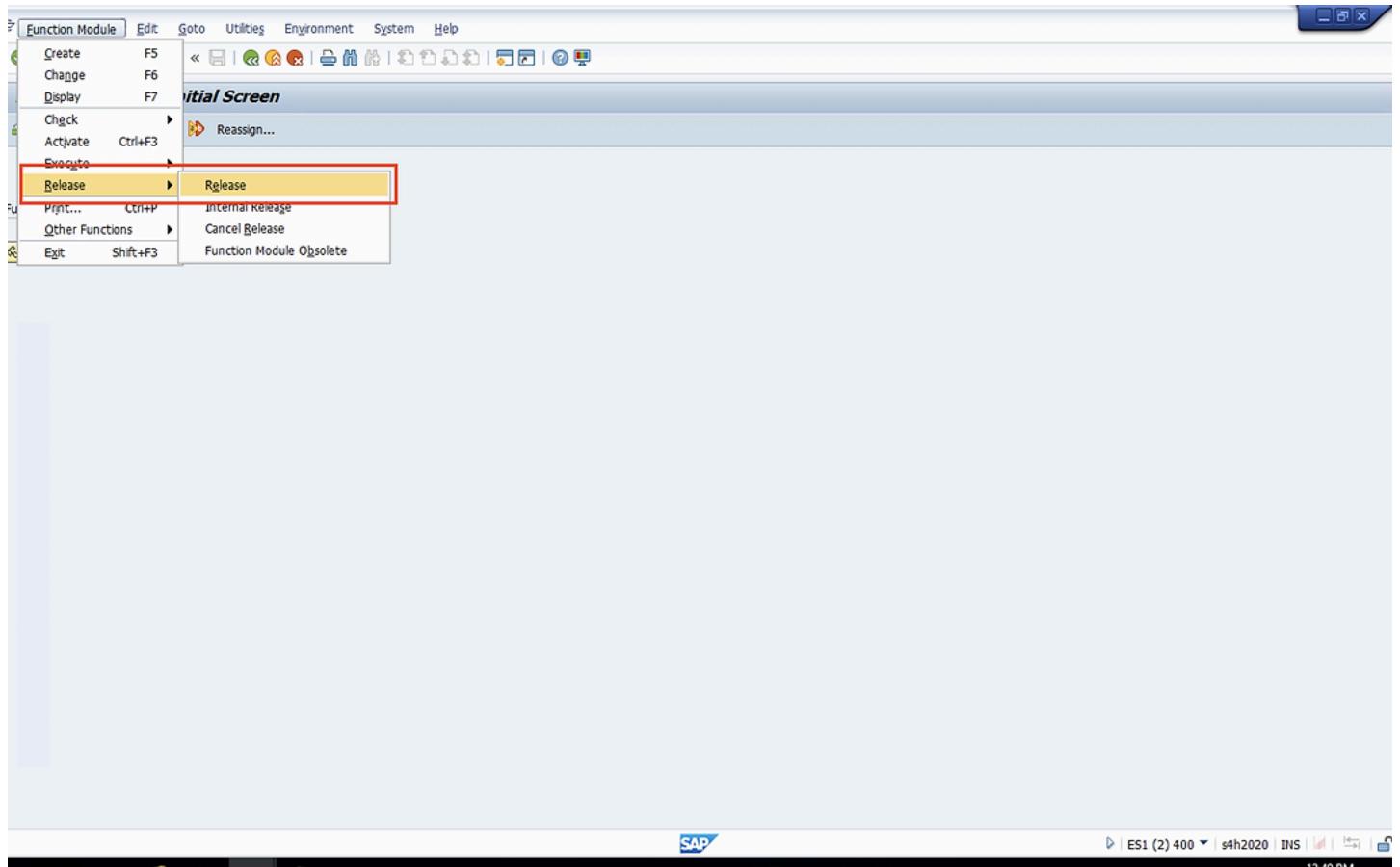
```

The status bar at the bottom shows "Scope: \FUNCTION ZBAPI_INFO_DISPLAY", "ABAP", "Ln 1 Col 1", "CAP", and "NUM". The SAP logo is in the bottom right corner.

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.

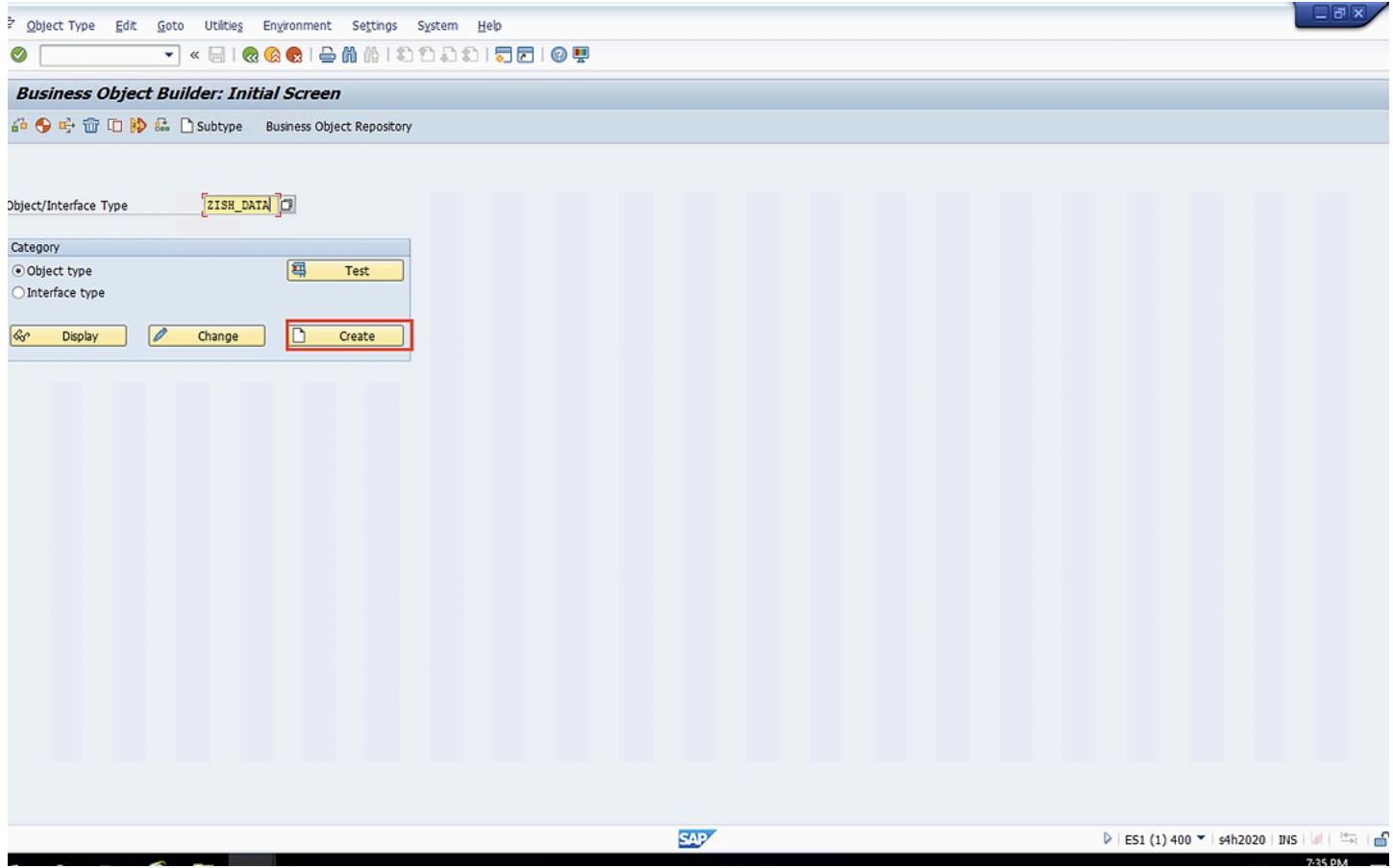


After that click on release and release.

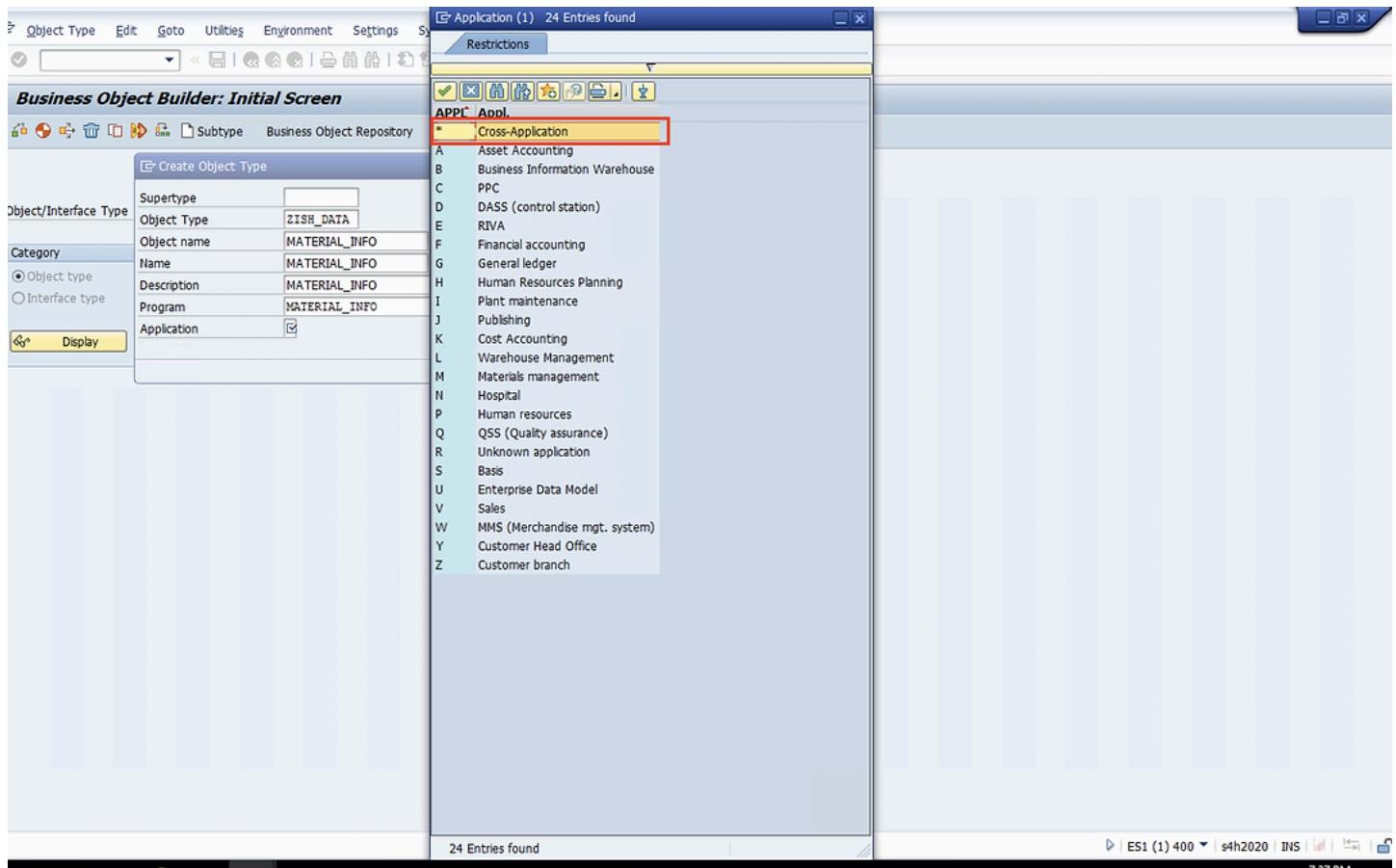


Step 4 - Create business object using tcode "SW01".

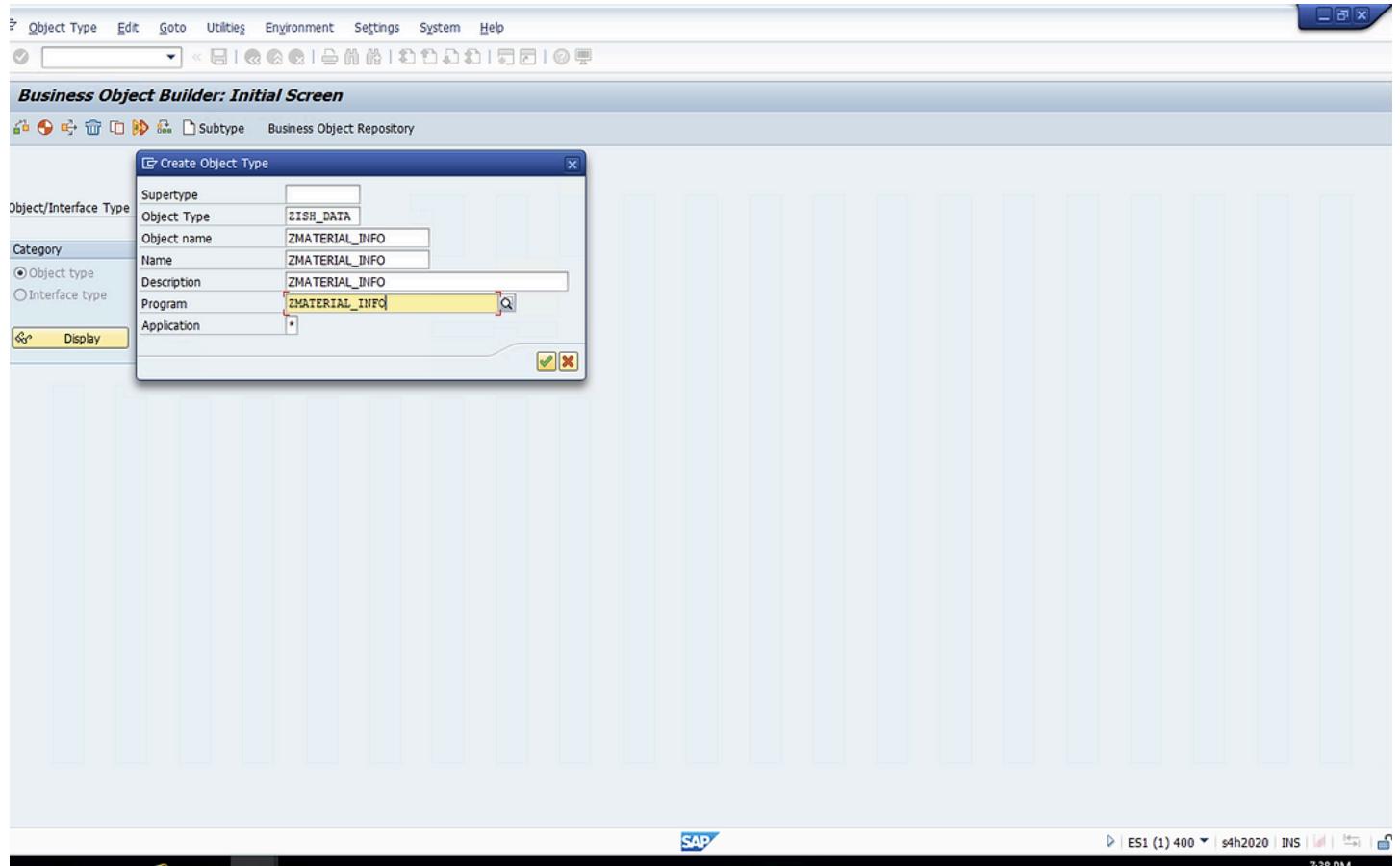
Write name of the business object and click on create.



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

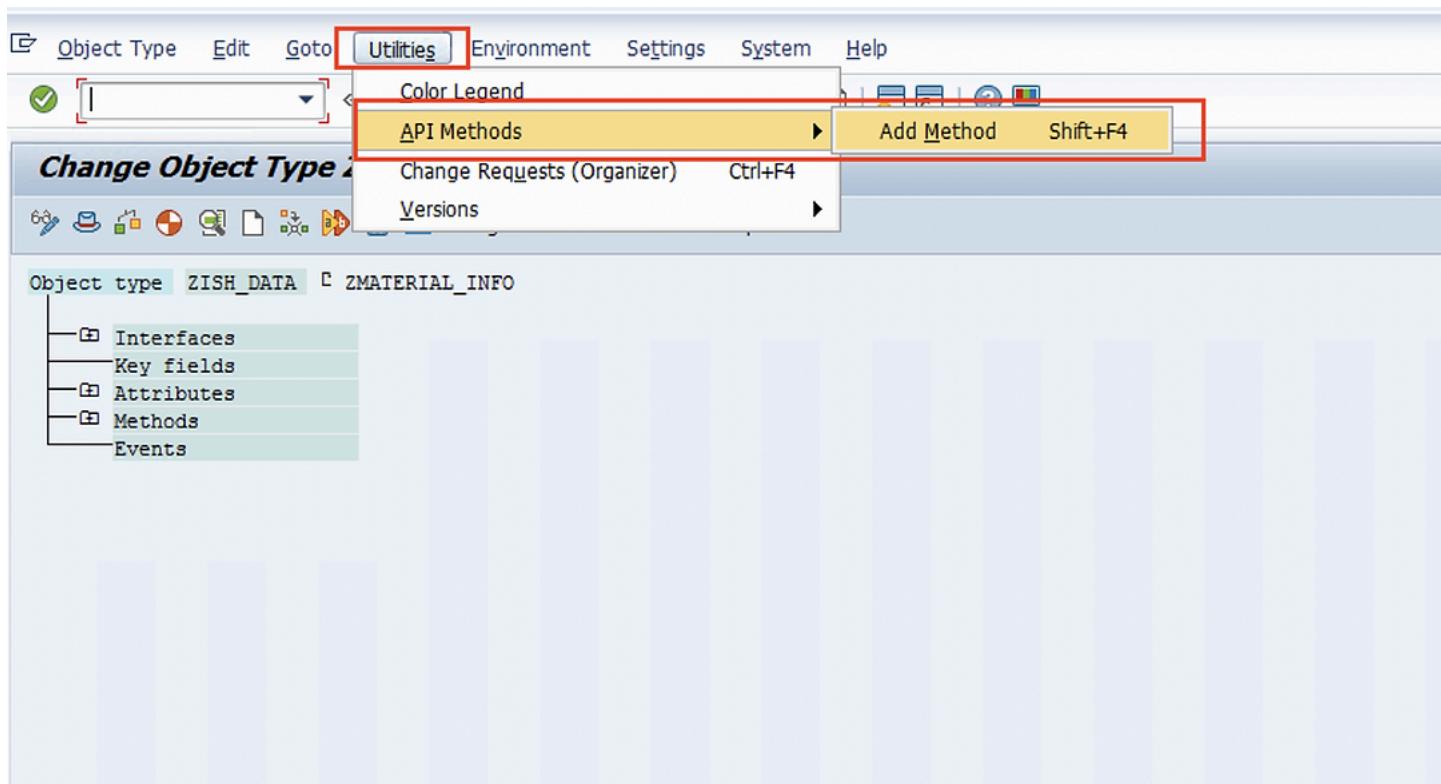
Change Object Type ZISH_DATA

Object type ZISH_DATA < ZMATERIAL_INFO

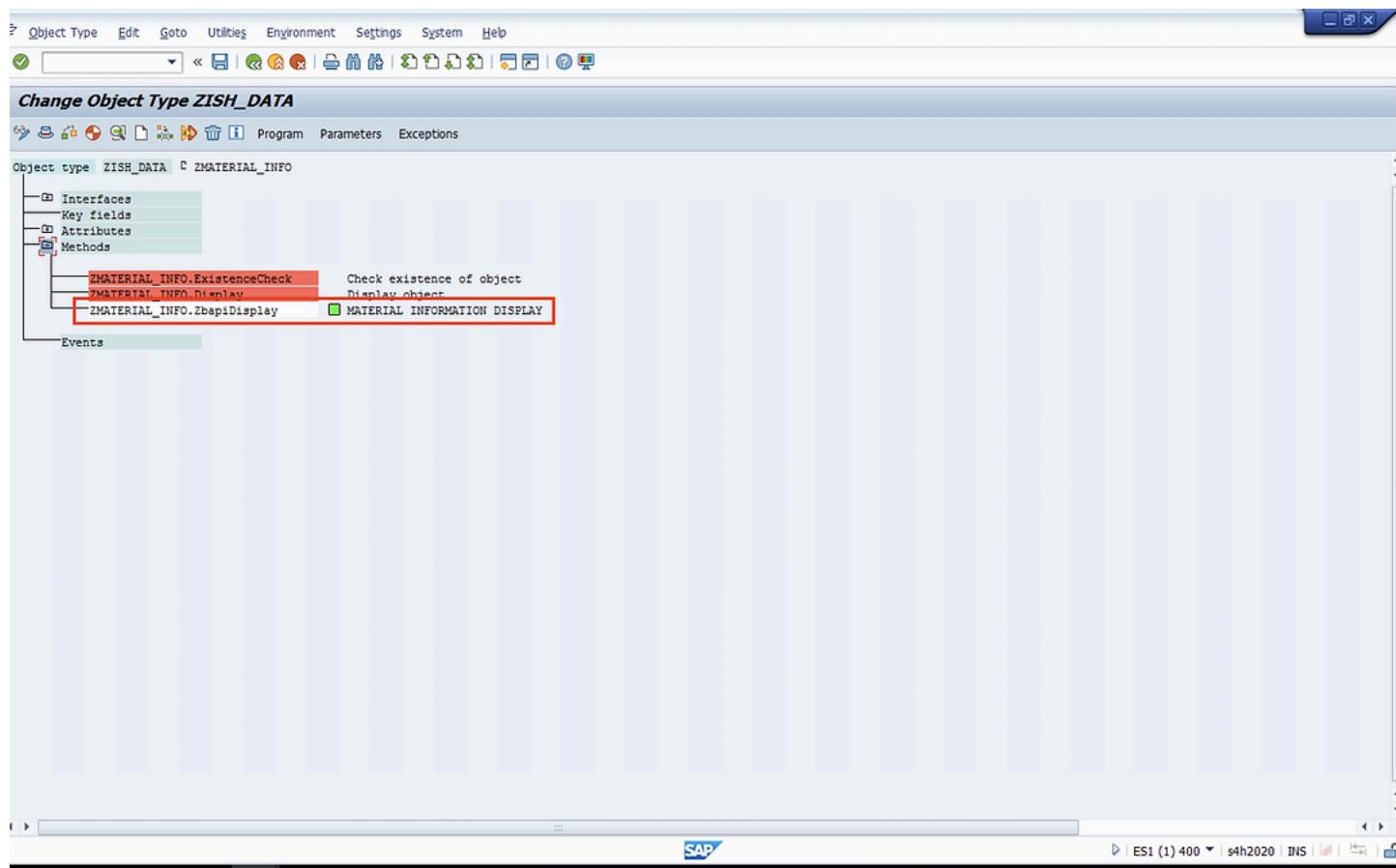
- Interfaces
- Key fields
- Attributes
- Methods
- Events

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.

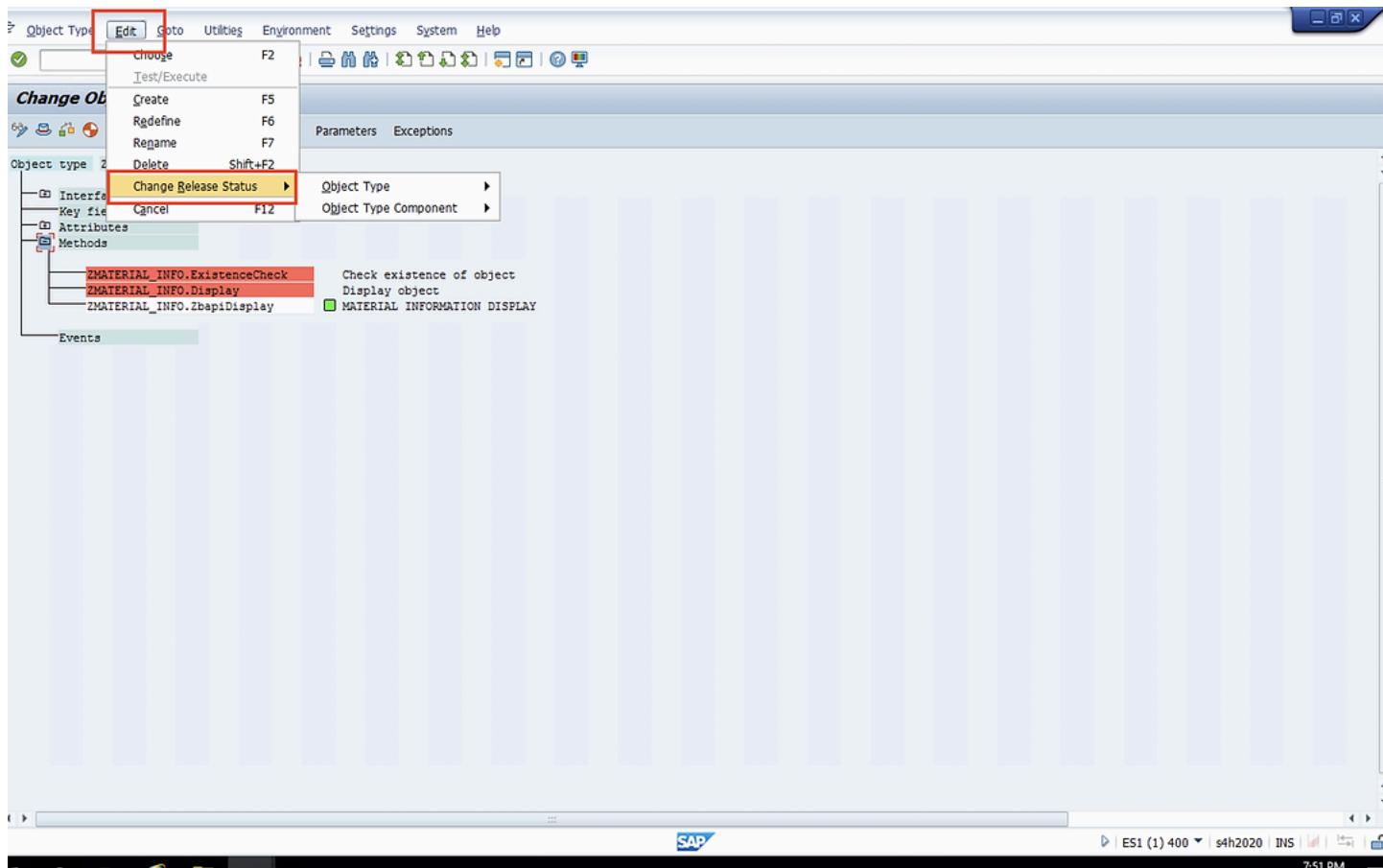


After this type the function module created for this program and click on enter. Function module will successfully be entered in the business object.

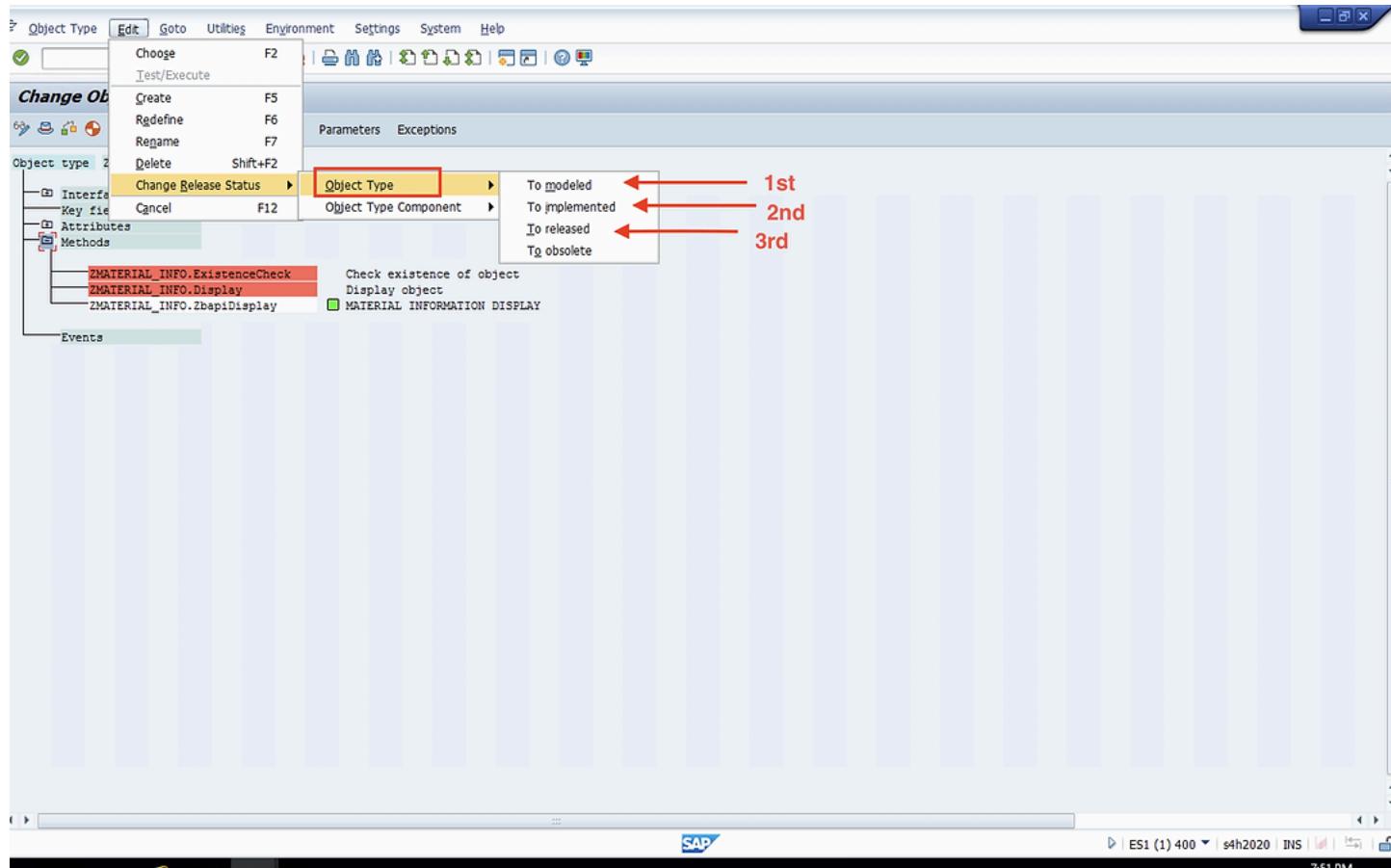


Step 6 - At last release business object components and component types and then finally release Business Object.

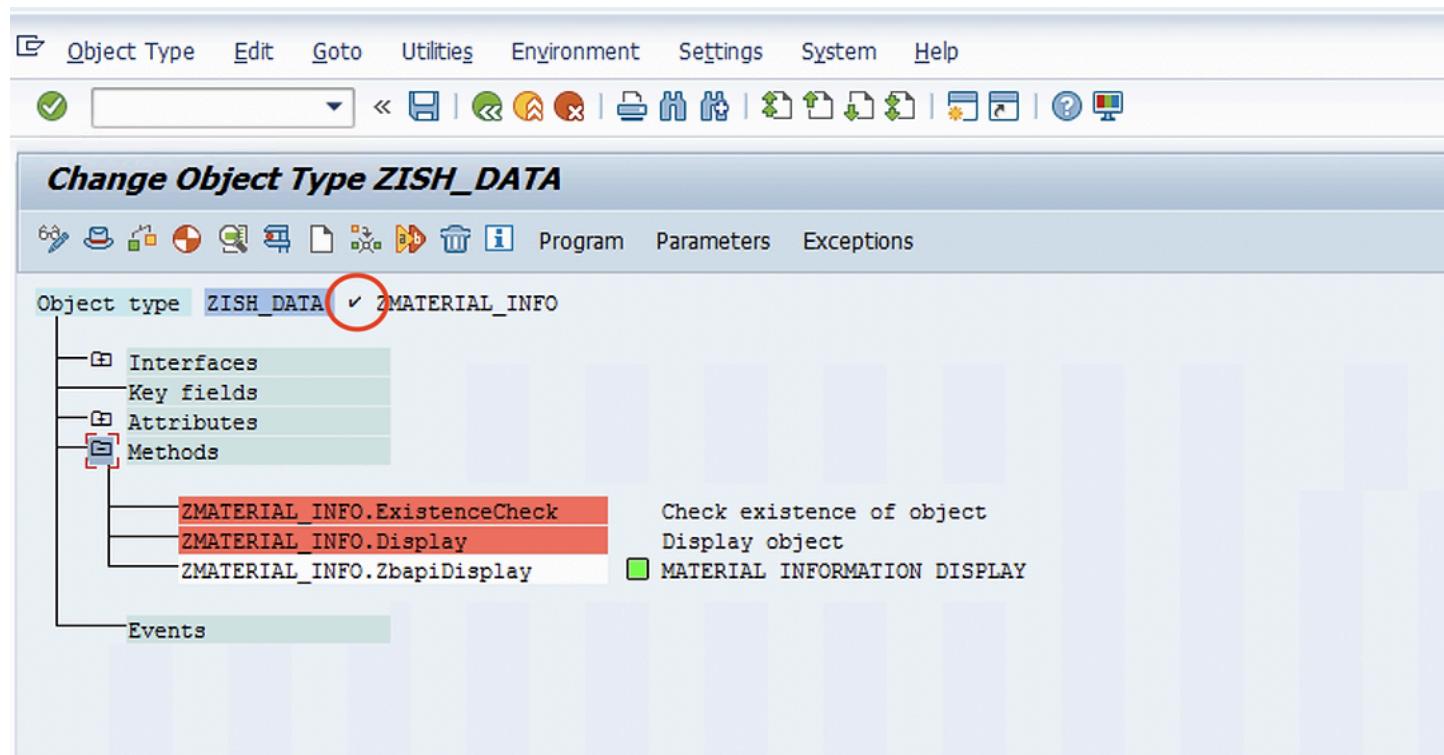
To release click on edit in the top menu then change release status. Now you will see two options. We have to release both object and object components.



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

Object type **ZISH_DATA** ✓ ZMATERIAL_INFO

- Interfaces
- Key fields
- Attributes
- Methods
 - ZMATERIAL_INFO.ExistenceCheck** Check existence of object
 - ZMATERIAL_INFO.Display** Display object
 - ZMATERIAL_INFO.ZbapiDisplay** ✓ MATERIAL INFORMATION DISPLAY
- Events

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

Object type **ZISH_DATA** ✓ ZMATERIAL_INFO

- Interfaces
- Key fields
- Attributes
- Methods
 - ZMATERIAL_INFO.ExistenceCheck** Check existence of object
 - ZMATERIAL_INFO.Display** Display object
 - ZMATERIAL_INFO.ZbapiDisplay** ✓ MATERIAL INFORMATION DISPLAY
- Events

GET TRAINED BY EXPERTS-CHECK OUR TRAINING PROGRAM