



Enhancement Framework - Function group and Function module enhancement - Concept and Simple Scenario



Anonymous
Feb 19, 2009

Watch

Enhancement Framework → Function group (Function module) enhancements

→ The enhancement framework for function modules allows us to enhance the standard application function module parameters with the new business functionality.

→ New parameters can be added based on the functionality required to be extended for a standard SAP object.

⚠ Note: The parameters which are enhanced must be '**optional**' in nature as being mandatory will require the changes to be registered for all the calls made by. This can be a real time consuming and tedious process.

⚠ Caution! - This functional ability can be enabled or achieved using the '**COPY**' or popularly called as '**CLONING**' technique which will serve the purpose, but will be a great tough time working during an '**UPGRADE**'.

→ As a SAP developer the best utilization would be the combined '**Benefit**' of '**Function-group (function module)**' enhancement with 'Source code enhancements (**Source code plug-ins**)'.

💡 Here, to demonstrate the powerful technique of 'Function-group' enhancement we consider a basic level scenario which will make things simple.

⚠ Note: This scenario deals with a '**Z**' function module but practically what would be the approach to enhance the standards can be well seen.

📘 A few conventions followed in this process and terminologies,

'DV_KUNNR' - Customer Number
'DV_CCODE' - Country Code
'DV_NAME' - Customer Name
'DV_STRAS' - Street or House Number
'DV_LIFNR' - Vendor

Consider a customer has a function module (say) '**ZDAVE_ENHC_INTERFACE**'.

There are two standard (say) parameters as '**Importing**' (DV_KUNNR and DV_CCODE)

Open in app

Function Module Edit Goto Utilities(M) Environment System Help

Function Builder: Display ZDAVE_ENHC_INTERFACE

Function module ZDAVE_ENHC_INTERFACE Active

Attributes **Import** Export Changing Tables Exceptions Source code

Standard Interface Parameters

Parameter Name	Type	Associated Type	Default value	Opti	Pas	Short text
DV_KUNNR	TYPE	KUNNR		<input type="checkbox"/>	<input checked="" type="checkbox"/>	Customer Number 1
DV_CCODE	TYPE	LAND1		<input type="checkbox"/>	<input checked="" type="checkbox"/>	Country Key

'DV_NAME' as 'Exporting'

Function Module Edit Goto Utilities(M) Environment System Help

Function Builder: Display ZDAVE_ENHC_INTERFACE

Function module ZDAVE_ENHC_INTERFACE Active

Attributes Import **Export** Changing Tables Exceptions Source code

Standard Interface Parameter

Parameter Name	Type spec.	Associated Type	Pass Val	Short text
DV_NAME	TYPE	NAME1	<input checked="" type="checkbox"/>	Name

★ Check out the 'Global Data' declarations for structures and internal tables in the function group.

Function Module Edit **Goto** Utilities(M) Environment System Help

Function Builder

Function module ZDAVE_ENHC_INTERFACE Active

Attributes Import

Global Data

- Main Program
- Text elements
- Messages
- Further Options
- Object Directory Entry
- Documentation
- Translation
- Application Help
- Back

Open in app

Parameter Name DV_NAME

Parameter Name	Type	Associated Type	Pass Val	Short text
DV_NAME	TYPE	NAME1	<input checked="" type="checkbox"/>	Name



Originally the structure 'FS_KNA1' and internal table 'T_KNA1' are present.

```
1  function-pool zdave_enhc_interface_fgp.      "MESSAGE-ID ..
2
3  data:
4  ▢ begin of fs_kna1,
5    |   kunnr type kunnr,
6    |   land1 type land1,
7    |   name1 type name1,
8    |   end of fs_kna1.
9
10 data:
11   t_kna1 like
12   standard table
13   of fs_kna1.
```

Original Declarations in structure - Function group

Referring back to the function module '**Source code**' which was originally written

Function Builder: Display ZDAVE_ENHC_INTERFACE

Function module: ZDAVE_ENHC_INTERFACE Active

Attributes Import Export Changing Tables Exceptions Source code

```

1 function zdave_enhc_interface.
2
3
4
5
6
7
8
9
10
11 select kunnr
12        land1
13        name1
14 from kna1
15 into table
16        t_kna1
17 where kunnr = dv_kunnr
18        and land1 = dv_ccode.
19
20 loop at t_kna1 into fs_kna1.
21   dv_name = fs_kna1-name1.
22 endloop.
23
24
25 endfunction.

```

Original Source Code - Function module

★ Test the function module to check its original functionality

Test Function Module: Initial Screen

Debugging Test data directory

Test for function group ZDAVE_ENHC_INTERFACE_FGP
Function module ZDAVE_ENHC_INTERFACE
Uppercase/Lowercase ☐

Import parameters	Value
DV_KUNNR	1000
DV_CCODE	IN

✔ Observe the output, which gives the 'Name' [Open in app](#) on input of 'Customer' number and 'Country code'.

Function modules Edit Goto Utilities(M) System Help

Test Function Module: Result Screen

Test for function group ZDAVE_ENHC_INTERFACE_FGP
 Function module ZDAVE_ENHC_INTERFACE
 Uppercase/Lowercase ☐

Runtime: 599 Microseconds

Import parameters	Value
DV_KUNNR	1000
DV_CCODE	IN

Export parameters	Value
DV_NAME	Harsh Dave

- Now, the time has come to enhance the original business functionality
- Follow the path '**Function module → Enhance Interface**'.

Function Module Edit Goto Utilities(M) Environment System Help

ZDAVE_ENHC_INTERFACE

Enhance Interface

Field Type	Default value	Opti	Pas	Short text
		<input type="checkbox"/>	<input checked="" type="checkbox"/>	Customer Number 1
		<input type="checkbox"/>	<input checked="" type="checkbox"/>	Country Key

Fill the 'Create' implementation details

Create Enhancement Implementation

Enhancement Implementation ZDAVE_ENHC_INTERFACE_CUSTOMER

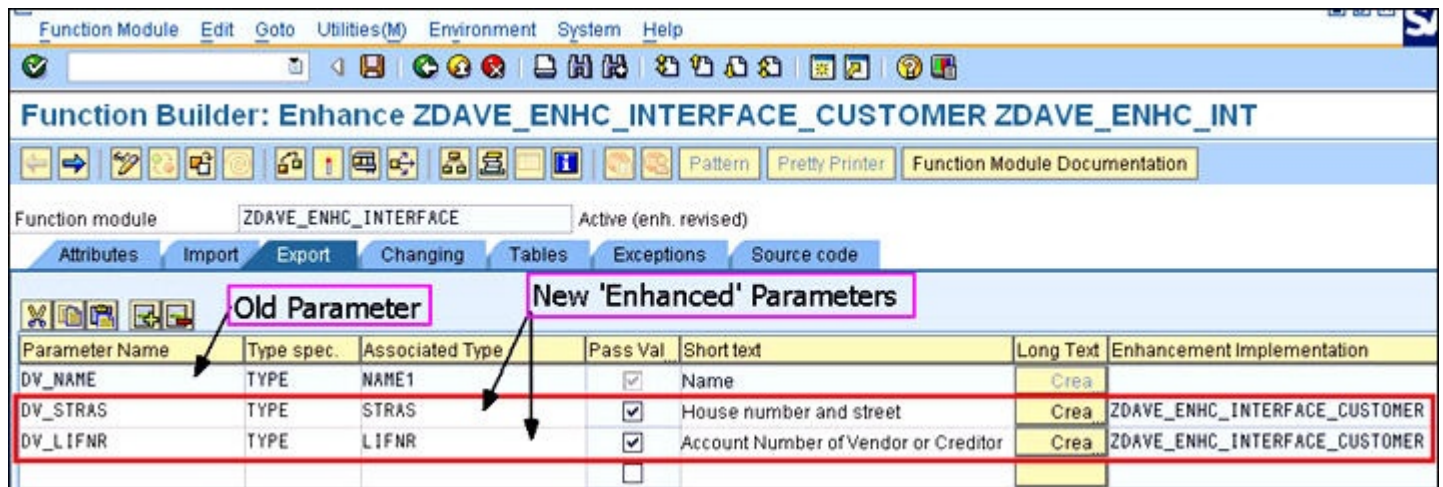
Short Text Enhance the customer interface

Composite Enhancement Implementation

Open in app

⊕ The customer (say) wants to add two new parameters to the original **'export'** parameter 'DV_NAME' with 'DV_STRAS' and 'DV_LIFNR'.

★ Observe the difference,



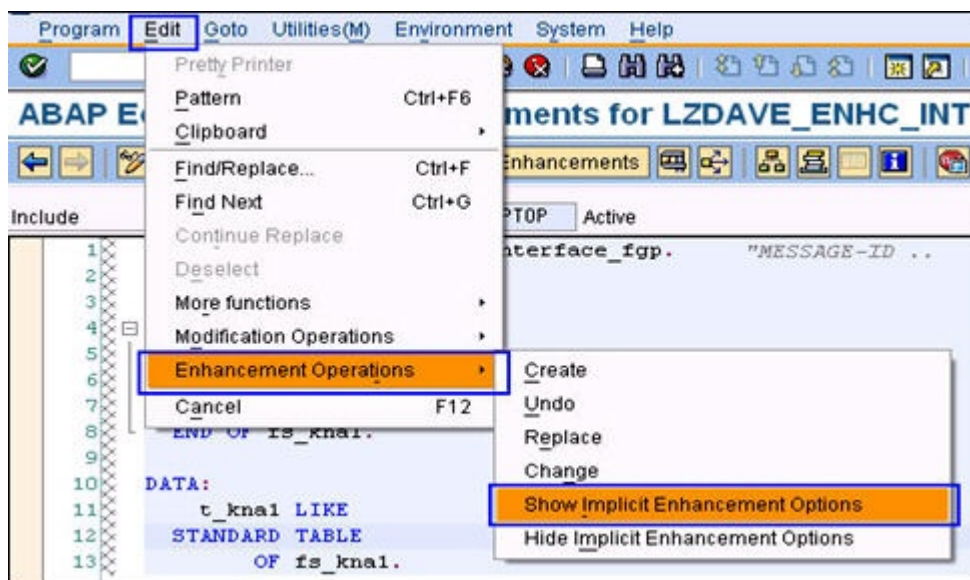
The screenshot shows the SAP Function Builder interface for the function module ZDAVE_ENHC_INTERFACE_CUSTOMER. The 'Export' tab is active, displaying a table of parameters. The table has columns: Parameter Name, Type spec., Associated Type, Pass Val, Short text, Long Text, and Enhancement Implementation. The original parameter DV_NAME is highlighted in yellow. Two new parameters, DV_STRAS and DV_LIFNR, are added and highlighted in red. Arrows point from labels 'Old Parameter' and 'New 'Enhanced' Parameters' to their respective rows in the table.

Parameter Name	Type spec.	Associated Type	Pass Val	Short text	Long Text	Enhancement Implementation
DV_NAME	TYPE	NAME1	<input checked="" type="checkbox"/>	Name	Crea	
DV_STRAS	TYPE	STRAS	<input checked="" type="checkbox"/>	House number and street	Crea	ZDAVE_ENHC_INTERFACE_CUSTOMER
DV_LIFNR	TYPE	LIFNR	<input checked="" type="checkbox"/>	Account Number of Vendor or Creditor	Crea	ZDAVE_ENHC_INTERFACE_CUSTOMER

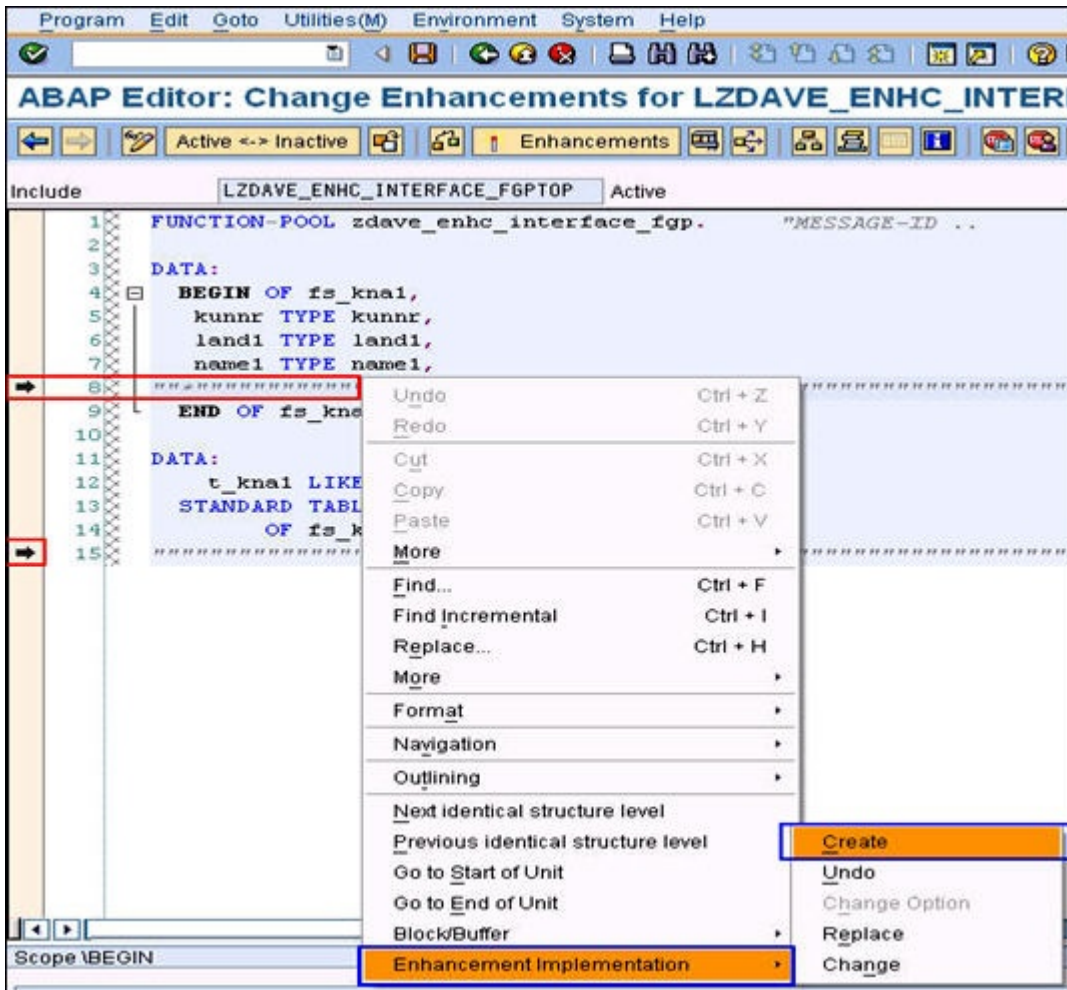
Next step would be to enhance the **'Source code'** in order to link the new parameters and perform the required operations.

So create an **'Implicit enhancement'**.

⚠ **Remember!** : We can add our code implicitly which makes it little flexible as **'Explicit'** needs **'ENHANCEMENT-POINT'** or an **'ENHANCEMENT-SECTION'** which are not present everywhere but only at some locations.



ℹ As it is understood that 'Implicit' can be done at certain predefined places in the source code, here we enhance the original structure 'FS_KNA1' in the **function group** → **global data**.



Fill in the details to enhance the original structure 'T_KNA1' with fields 'STRAS' and 'LIFNR' respectively.

Create Enhancement Implementation

Enhancement Implementation	ZDAVE_ENHC_FGP_IMPLICIT
Short Text	Implicit enhancement to enhance the KNA1 structures
Composite Enhancement Implementation	

☒ ☐

★ Activate the enhancement and observe.

Program Edit Goto Utilities(M) Environment System Help

ABAP Editor: Change Enhancement ZDAVE_ENHC_FGP_IMPLICIT

Include LZDAVE_ENHC_INTERFACE_FGPTOP Active

```

1 FUNCTION-POOL zdave_enhc_interface_fgp. "MESSAGE-ID ..
2
3 DATA:
4 BEGIN OF fs_knal,
5   kunnr TYPE kunnr,
6   land1 TYPE land1,
7   name1 TYPE name1,
8
9   *$*$-Start: (1 )-----
10  ENHANCEMENT 1 ZDAVE ENHC FGP IMPLICIT. "active version
11  DATA:
12    stras TYPE stras,
13    lifnr TYPE lifnr.
14  ENDENHANCEMENT.
15  *$*$-End: (1 )-----
16  END OF fs_knal.
17
18 DATA:
19   t_knal LIKE
20   STANDARD TABLE
21   OF fs_knal.
22

```

Create another implicit enhancement for after the complete original code ends.

Program Edit Goto Utilities(M) Environment System Help

ABAP Editor: Change Enhancement ZDAVE_ENHC_FGP_IMPLICIT

Include LZDAVE_ENHC_INTERFACE_FGPTOP Active

```

4 BEGIN OF fs_knal,
5   kunnr TYPE kunnr,
6   land1 TYPE land1,
7   name1 TYPE name1,
8
9   *$*$-Start: (1 )-----
10  ENHANCEMENT 1 ZDAVE ENHC FGP IMPLICIT. "active version
11  DATA:
12    stras TYPE stras,
13    lifnr TYPE lifnr.
14  ENDENHANCEMENT.
15  *$*$-End: (1 )-----
16  END OF fs_knal.
17
18 DATA:
19   t_knal LIKE
20   STANDARD TABLE
21   OF fs_knal.
22

```

Context Menu:

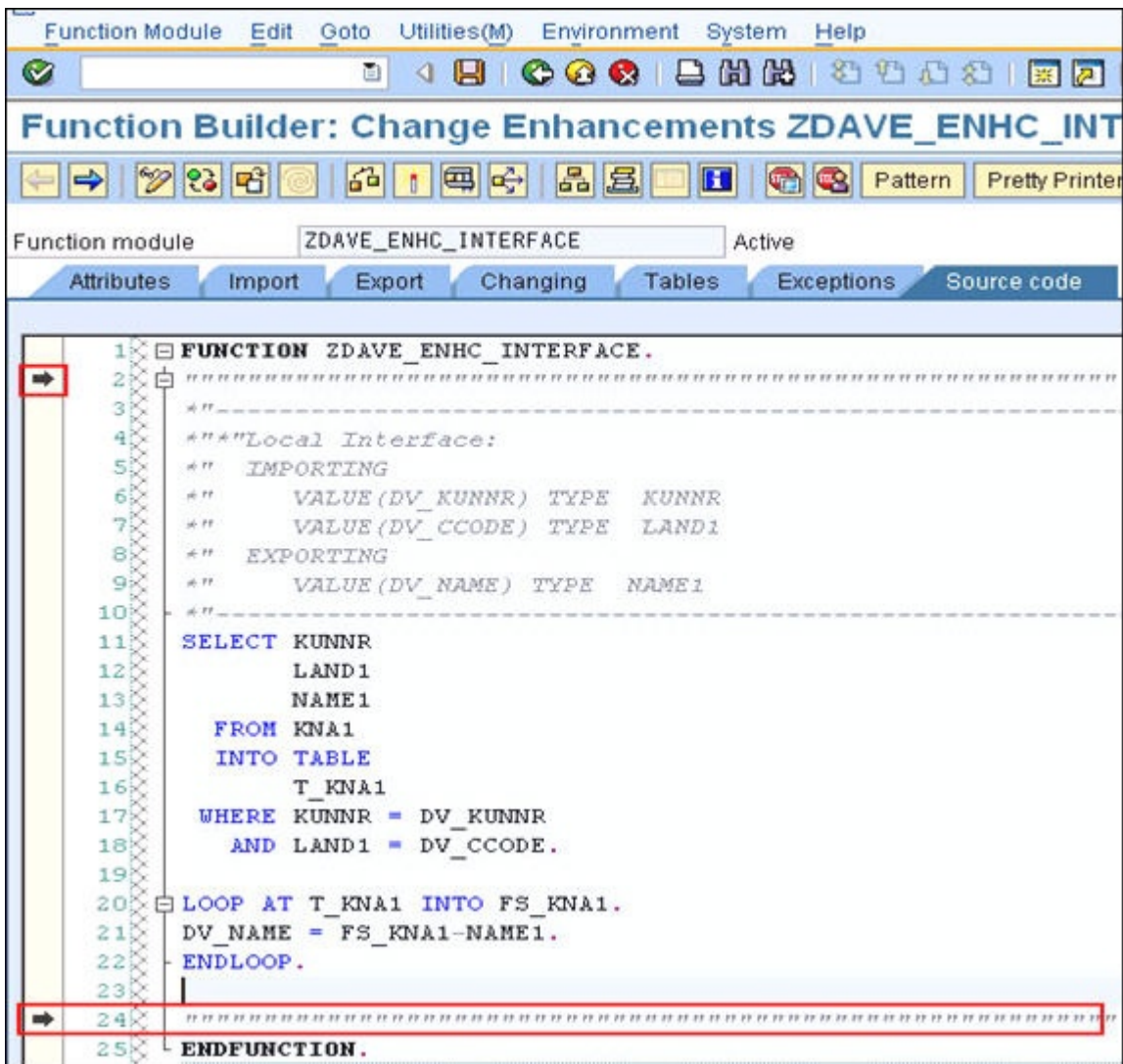
- Undo (Ctrl + Z)
- Redo (Ctrl + Y)
- Cut (Ctrl + X)
- Copy (Ctrl + C)
- Paste (Ctrl + V)
- More
- Find... (Ctrl + F)
- Find Incremental (Ctrl + I)
- Replace... (Ctrl + H)
- More
- Format
- Navigation
- Outlining
- Next identical structure level
- Previous identical structure level
- Go to Start of Unit
- Go to End of Unit
- Block/Buffer
- Enhancement Implementation
- Create
- Undo
- Change Option
- Replace
- Change

Open in app

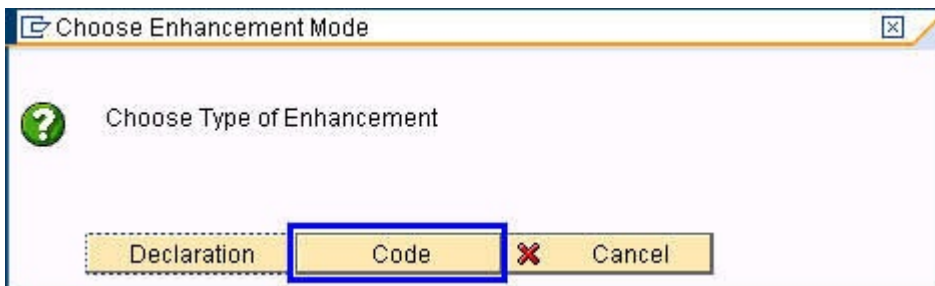
Note: Fill in the 'Create' implementation dialog for the implicit enhancement which will allow creating a new internal table which will hold the original with the additional data required and finally the data will be transferred to the original table in order to make the functionality work.

```
10 ENHANCEMENT 1 ZDAVE_ENHC_FGP_IMPLICIT. "active version
11 DATA:
12     stras TYPE stras,
13     lifnr TYPE lifnr.
14 ENDENHANCEMENT.
15 *$$$-End: (1)-----
16 END OF fs_kna1.
17
18 DATA:
19     t_kna1 LIKE
20     STANDARD TABLE
21     OF fs_kna1.
22
23 *$$$-Start: (2)-----
24 ENHANCEMENT 2 ZDAVE_ENHC_FGP_IMPLICIT.
25 DATA:
26     BEGIN OF FS_KNA2,
27         KUNNR TYPE KUNNR,
28         LAND1 TYPE LAND1,
29         NAME1 TYPE NAME1,
30         STRAS TYPE STRAS,
31         LIFNR TYPE LIFNR,
32     END OF FS_KNA2.
33
34 DATA:
35     T_KNA2 LIKE
36     STANDARD TABLE
37     OF FS_KNA2.
38
39 ENDENHANCEMENT.
```

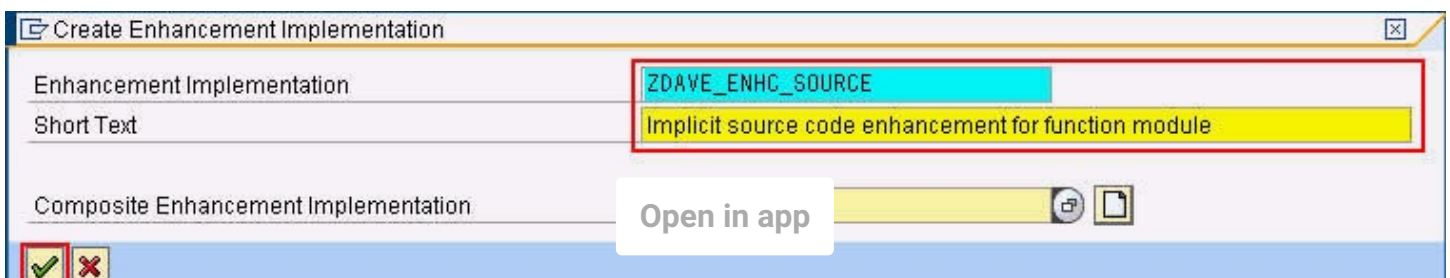
After enhancing the function group, make sure all the enhancements are active and return to the function module source code to enhance the functionality.



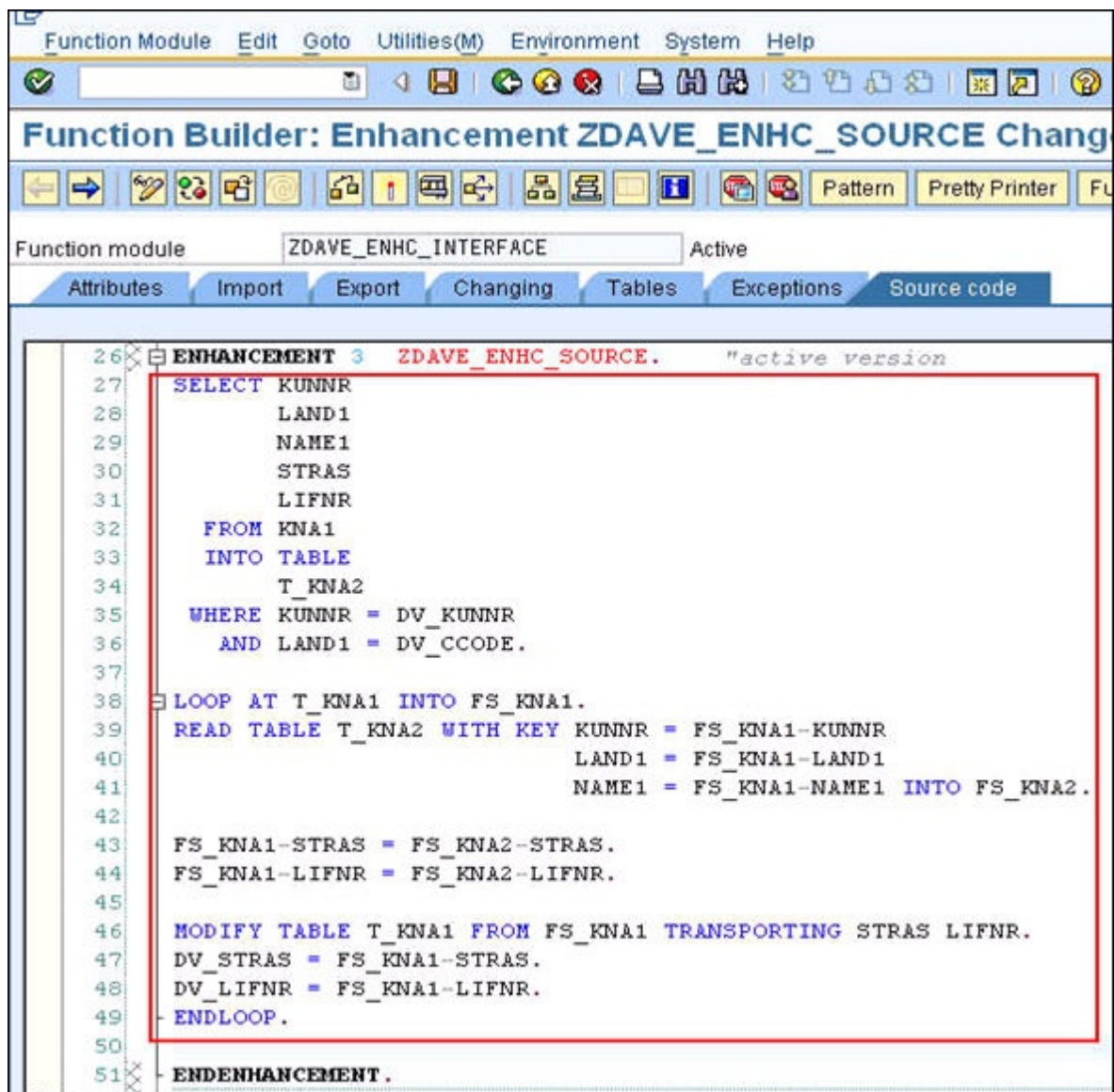
★ Choose 'CODE' for 'dynamic enhancement implementation'.



Fill in the details and possibly choose a different name in order to easily manage the implementations



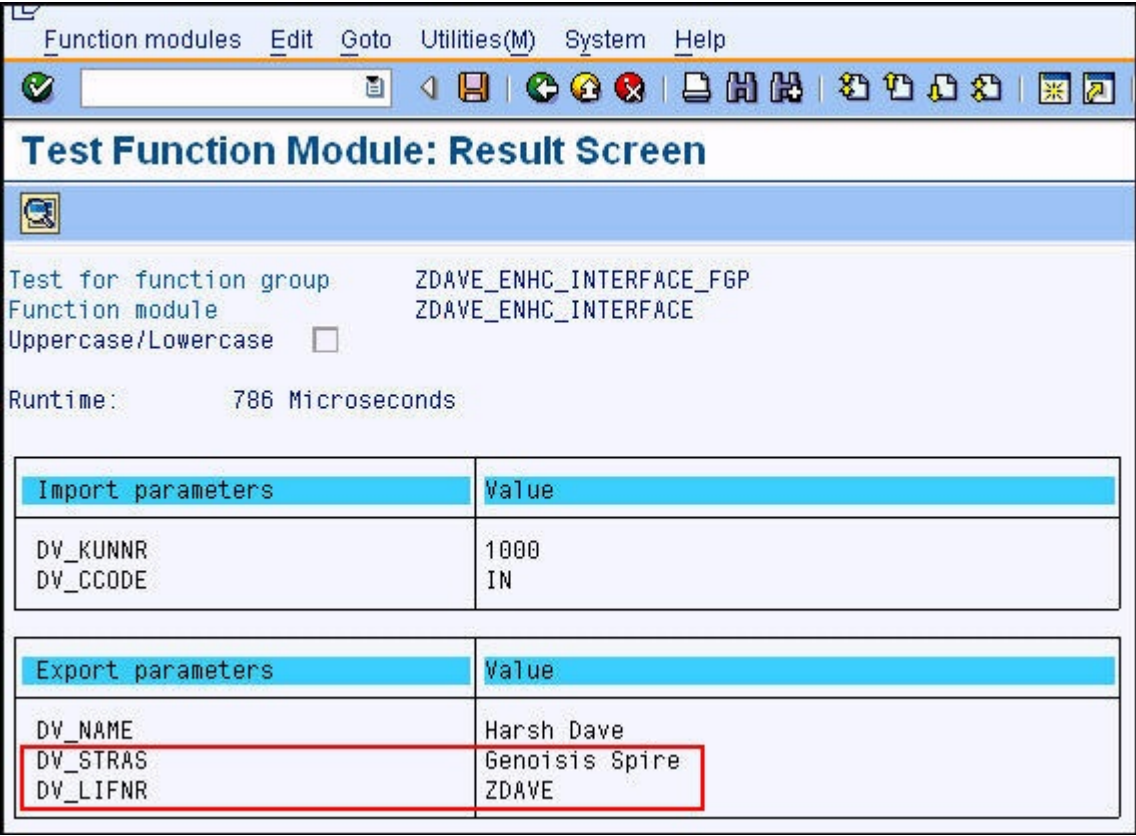
★ Observe the code very carefully and then activate the enhancement.



```
26  ENHANCEMENT 3  ZDAVE_ENHC_SOURCE.      "active version
27  SELECT KUNNR
28         LAND1
29         NAME1
30         STRAS
31         LIFNR
32  FROM KNA1
33  INTO TABLE
34         T_KNA2
35  WHERE KUNNR = DV_KUNNR
36         AND LAND1 = DV_CCODE.
37
38  LOOP AT T_KNA1 INTO FS_KNA1.
39    READ TABLE T_KNA2 WITH KEY KUNNR = FS_KNA1-KUNNR
40                                LAND1 = FS_KNA1-LAND1
41                                NAME1 = FS_KNA1-NAME1 INTO FS_KNA2.
42
43    FS_KNA1-STRAS = FS_KNA2-STRAS.
44    FS_KNA1-LIFNR = FS_KNA2-LIFNR.
45
46    MODIFY TABLE T_KNA1 FROM FS_KNA1 TRANSPORTING STRAS LIFNR.
47    DV_STRAS = FS_KNA1-STRAS.
48    DV_LIFNR = FS_KNA1-LIFNR.
49  ENDLOOP.
50
51  ENDENHANCEMENT.
```

💡 Finally, the time has come to confirm the functionality of the enhanced object.

✅ Test and observe the difference. 😊



Like

Open in app