

How to Extend an Outbound IDoc



Applies to:

Developing and configuring SAP Intermediate Documents (IDocs) for data transfer.

Related till version ECC 6.0.

For more information, visit the [Idoc homepage](#) and the [ABAP homepage](#).

Summary

This document provides details on why we need an extended Idoc and how to create it. We will also see a step by step procedure of creating an extended Idoc. The Steps involve creating a segment of required structure, Extend basic Idoc and assign new segment to extended Idoc. Also involves Maintaining message type, partner profile and implementing the User exit. Further the Idoc is transferred to the destination. A whole overview of Extended Outbound Idoc can be learned with this article. Though some related information already available in SCN, to make it more comprehensive, I plan to explain it using screen shots along with a real world example making it easier.

Author: Lakshmi Narayana Neeli

Company: Intelligroup Asia Pvt Limited.

Created on: 14 Aug 2009

Author Bio

Lakshmi Narayana Neeli works with Intelligroup Asia Pvt Limited since 2008. He has good knowledge in SAP ABAP and CRM.

Table of Contents

Introduction	3
Why We Need an Extended Idoc?	3
Business Scenario – Invoice.....	4
Basic Steps in Invoice Extension Outbound Idoc	4
WE31- Create Segment Type	4
Create IDOC Type	9
Create Logical Message Type	15
Create Output Types and Assignment to IDOC Type.....	16
Creating Partner Profile.....	17
Finding and Updating Customer Exit Using SMOD	21
Outbound Process Code.....	25
Create a Transaction and Send IDoc.....	31
Monitoring IDoc	34
Related Content.....	36
Disclaimer and Liability Notice.....	37

Introduction

SAP's presence in the IT world is propelled by its unique integration of all its sub systems to a central system. More than a linking together of applications, implementing SAP redirects the flow of information through a company and its partners to enhances the potential of its business functions. This flow of information is enabled by a core element - Intermediate Document, or IDoc. Technically, the IDoc is an example of Electronic Data Interchange (EDI). The IDoc concept borrows the best features of EDI and combines them with the best features of conventional transaction file formats. SAP R/3 systems send out data through Idoc (Intermediate Document), which internally has segments and fields containing the data.

Why We Need an Extended Idoc?

SAP send out data through Idoc with many predefined segments, Message types and fields. But sometimes, these fields are not sufficient for a specific end-to-end business scenario as far as data transfer is concerned. So in such scenario, we can add new segments with completely new structure to the standard Idoc as an extension called as extended Idoc. Here we create a new structure and insert it into existing Idoc structure creating a whole new Idoc satisfying the requirement. Following blog gives out step-by-step approach for creation of the same.

Business Scenario – Invoice

For instance let us take a scenario in billing, where we already have a predefined IDoc type 'INVOIC02'. But the requirement is to transfer additional structure containing VBRK-KTGRD (Account assignment group for this customer) and VBRK-MANSP (Dunning block).

To get the requirement fulfilled we wish to create a segment structure add segment with two additional fields as an extension to the existing IDoc Type 'INVOIC02'. The basic steps are explained here and going further a more detailed step by step illustration is done.

Basic Steps in Invoice Extension Outbound IDoc

- Create a segment with two VBRK-KTGRD (Account assignment group for this customer) and VBRK-MANSP (Dunning block)
- Extend basic IDoc with new segment
- Assign Extended IDoc to message type
- Maintain partner profile i.e., Update Extended IDoc in partner profile
- Find relevant user exit
- Implement User exit
- Update values into this your segment of extended IDoc
- Trigger output from VF02 with medium 6
- Check your IDoc in WE02

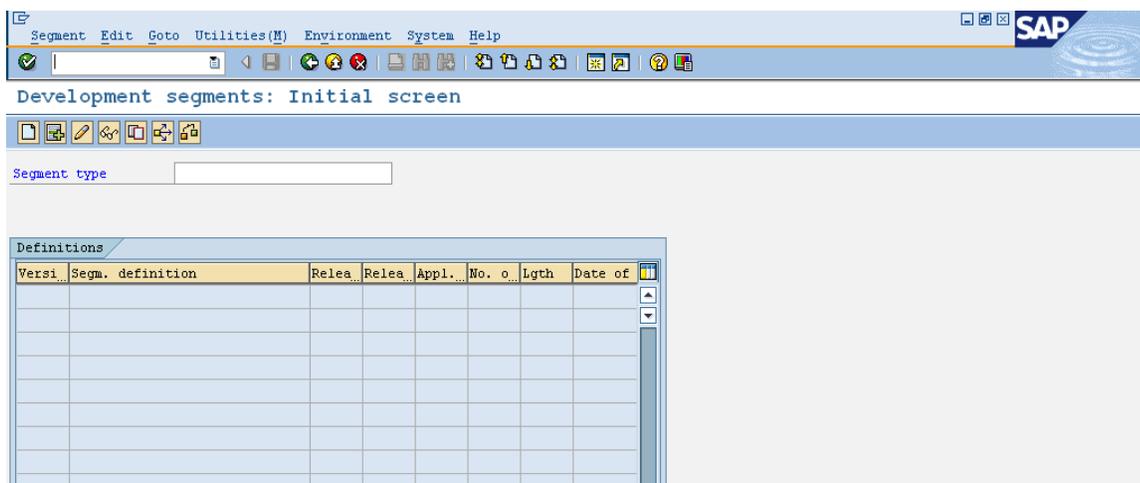
Now we will see a step by step creation of the extended IDoc.

Create a Segment WE31

To create a segment with two VBRK-KTGRD (Account assignment group for this customer) and VBRK-MANSP (Dunning block)

WE31- Create Segment Type

In this transaction we create a segment type. This segment type has two fields KTGRD and MANSP as specified from VBRK table. This segment will be used in extended IDoc as follows. In transaction WE31 we get the following screen.



Segment definition Edit Goto System Help

Development segments: Create segment definition

Segment type attributes
 Segment type: ZTRGEDC Qualified segment
 Short Description: BILLING SEGMENT

Segm. definition: Released
 Last Changed By:

Segment type ZTRGEDC: Change persons involved

Person responsible: NTUDEV
 Processing person: NTUDEV

Pos.	Field Name	Data element	ISO c.	Exp.
1	KTGRD	KTGRD	<input type="checkbox"/>	
2	MANSP	MANSP	<input type="checkbox"/>	
3			<input type="checkbox"/>	
4			<input type="checkbox"/>	
5			<input type="checkbox"/>	
6			<input type="checkbox"/>	
7			<input type="checkbox"/>	
8			<input type="checkbox"/>	
9			<input type="checkbox"/>	
10			<input type="checkbox"/>	
11			<input type="checkbox"/>	
12			<input type="checkbox"/>	
13			<input type="checkbox"/>	
14			<input type="checkbox"/>	
15			<input type="checkbox"/>	

Segment definition Edit Goto System Help

Development segments: Create segment definition

Segment type attributes
 Segment type: ZTRGEDC Qualified segment
 Short Description: BILLING SEGMENT

Segm. definition: Released
 Last Changed By:

Create Object Directory Entry

Object: R3TR TABL ZTRGEDC

Attributes

Package: Z15988_PACKAGE
 Person Responsible: NTUDEV
 Original System: EC6
 Original language: EN/English

Local Object Lock Overview

Click here

Pos.	Field Name	Data element
1	KTGRD	KTGRD
2	MANSP	MANSP
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Segment definition Edit Goto System Help

Development segments: Create segment definition

Segment type attributes
 Segment type: ZTRGEDC Qualified segment
 Short Description: BILLING SEGMENT

Segm. definition:
 Last Changed By:

Pos.	Field Name	Data element
1	KTGRD	KTGRD
2	MANSP	MANSP
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Prompt for transportable Workbench request

Table: ZTRGEDC

Request: EC6K904902 Workbench request

Short Description: purple's request

Own Requests

Segment definition Edit Goto System Help

Development segments: Change segment definition ZTRGEDC000

Segment type attributes
 Segment type: ZTRGEDC Qualified segment
 Short Description: BILLING SEGMENT

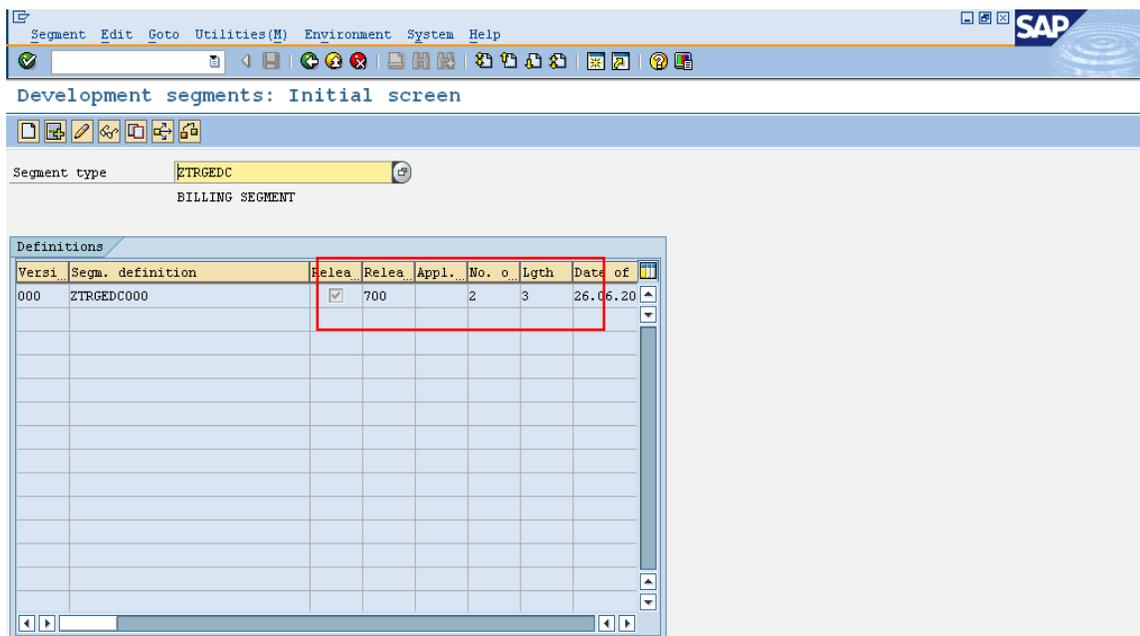
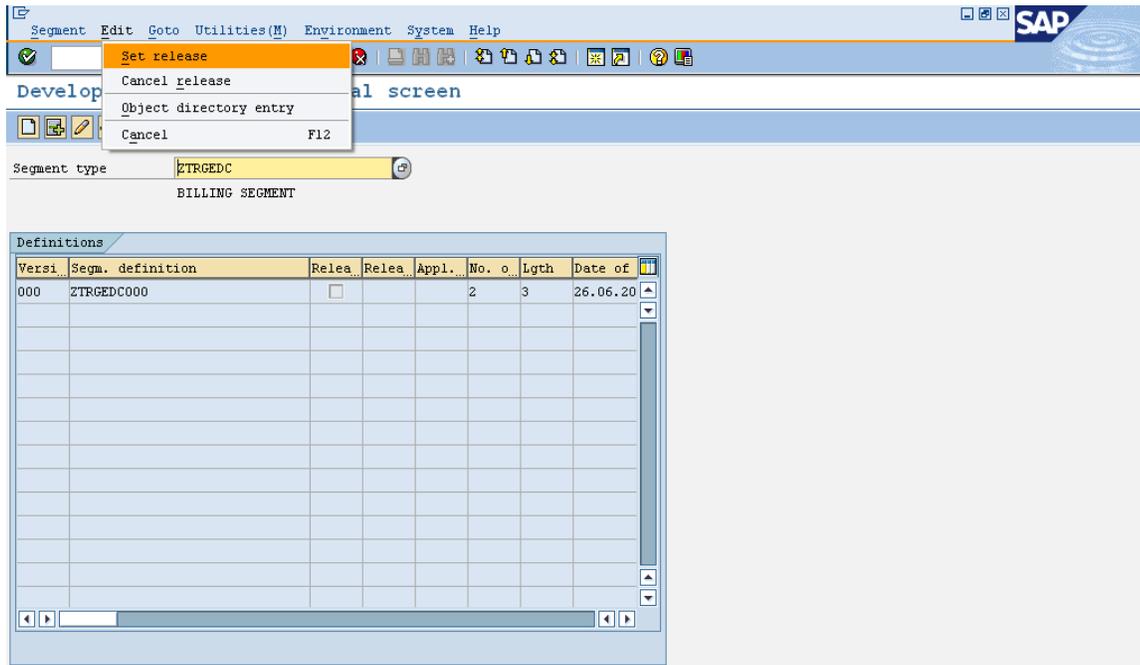
Segm. definition: ZTRGEDC000 Released
 Last Changed By: NTWDEV

Pos.	Field Name	Data element	I30 c.	Exp.
1	KTGRD	KTGRD	<input type="checkbox"/>	2
2	MANSP	MANSP	<input type="checkbox"/>	1
3			<input type="checkbox"/>	
4			<input type="checkbox"/>	
5			<input type="checkbox"/>	
6			<input type="checkbox"/>	
7			<input type="checkbox"/>	
8			<input type="checkbox"/>	
9			<input type="checkbox"/>	
10			<input type="checkbox"/>	
11			<input type="checkbox"/>	
12			<input type="checkbox"/>	
13			<input type="checkbox"/>	
14			<input type="checkbox"/>	
15			<input type="checkbox"/>	
16			<input type="checkbox"/>	

Segment is saved and assigned request number

Segment definition ZTRGEDC000 was saved with transport request EC6K904902

EC6 (2) 800 IGINECC6 INS

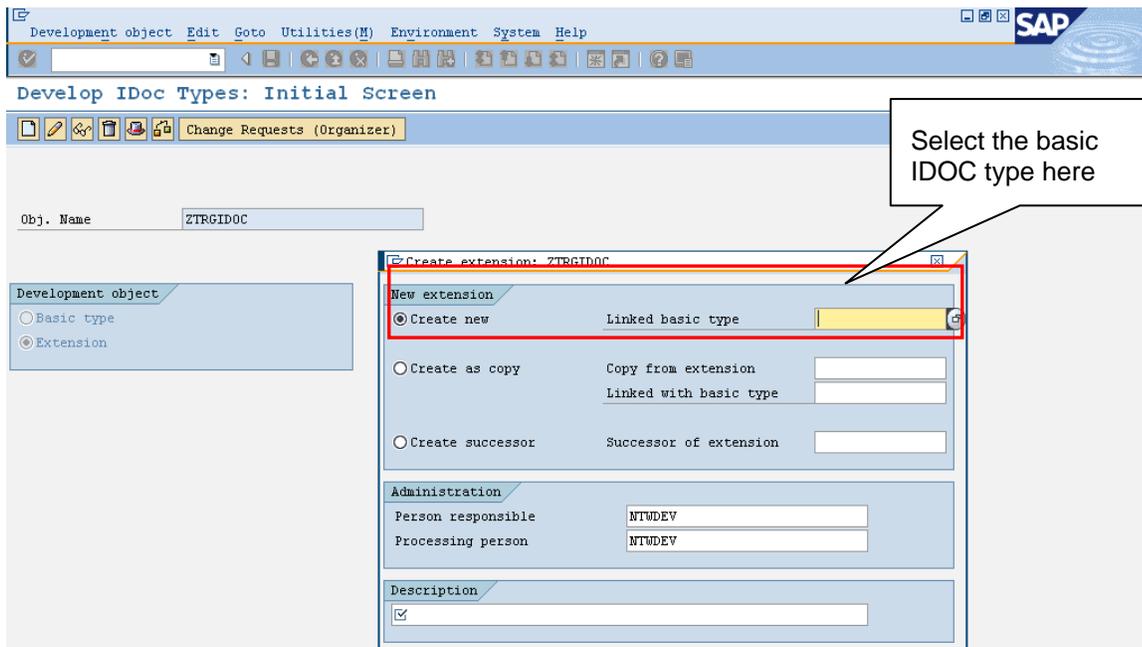
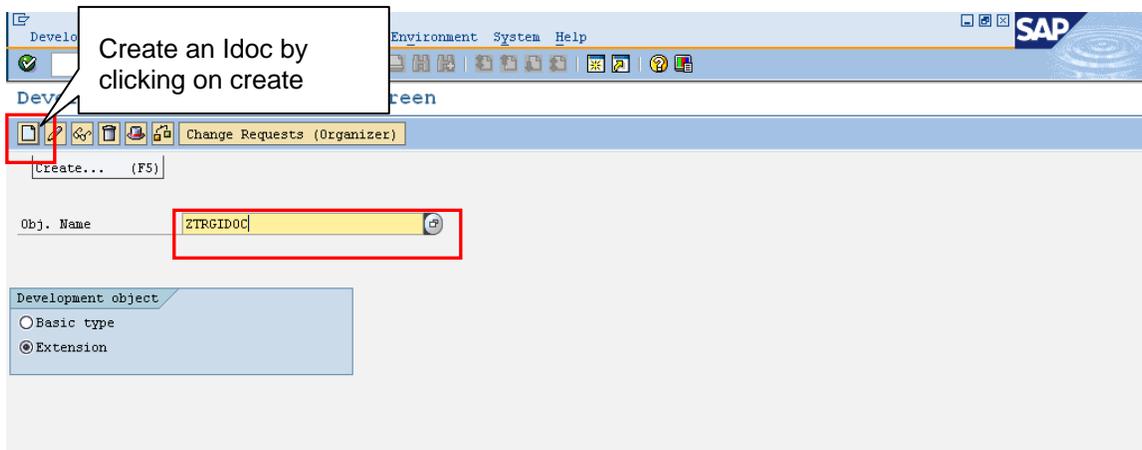
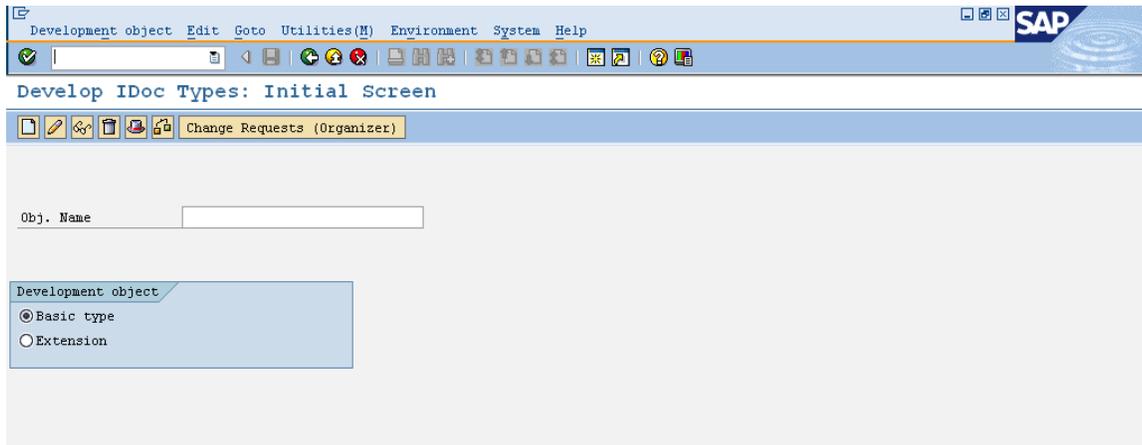


So segment is created and set to release. After you release the system automatically updates the **version release** (Here it is 700 in above diagram). Only one unique version segment is released per version i.e. here ZTRGEDC with release 700 exists once only and no changes are allowed until it is in released status (the Tick mark in Released tab).

So in order to make changes unrelease the segment and make changes and once again "set release" it. So now ZTRGEDC segment is created and set to release. Now we need to create the IDOC type.

Create IDOC Type

Go to WE30 to Create IDOC Type screen is as shown



Since we are creating the existing IDoc, we assign invoice02 here.

Development object Edit Goto Utilities(M) Environment System Help

Obj. Name: ZTRGIDOC

Development object: Basic type Extension

Create extension: ZTRGIDOC

New extension: Create new Linked basic type: INV*

Linked basic type (2) 6 Entries found

Basic type	Description	Released
INV_ID01	Invoice	<input checked="" type="checkbox"/>
INVCON01	Stock Change Data for Inventory Controlling	<input checked="" type="checkbox"/>
INVCON02	Stock Change Data for Inventory Controlling	<input type="checkbox"/>
INVOIC01	Invoice/Billing document	<input checked="" type="checkbox"/>
INVOIC02	Invoice/Billing document	<input checked="" type="checkbox"/>
INVRPT01	inventory report	<input checked="" type="checkbox"/>

Select this after pressing F4

Development object Edit Goto Utilities(M) Environment System Help

Obj. Name: ZTRGIDOC

Development object: Basic type Extension

Create extension: ZTRGIDOC

New extension: Create new Linked basic type: INVOIC02

Administration: Person responsible: NTWDEV, Processing person: NTWDEV

Description: EXTENDED IDOC

We get the following screen showing the set of segments. Since the extensions VBRK-KTGRD and VBRK-MANSP belong to "HEADER" tab in transaction VF02 the extension is done for the relevant segment type E1EDK01 related to "Header General Data". We extend it as shown.

The image consists of two screenshots from the SAP GUI showing the 'Create extension: ZTRGIDOC' dialog. The top screenshot shows the 'Create segment...' dialog with a list of segments. A red box highlights the 'E1EDK01' segment, and another red box highlights the 'EXTENDED IDOC' column. Callouts point to the 'Create' button and the 'E1EDK01' segment. A larger callout explains that extensions VBRK-KTGRD and VBRK-MANSP belong to the 'HEADER' tab, so E1EDK01 was selected. The bottom screenshot shows the same dialog with a callout pointing to the 'Create' button.

Segment	Description
E1EDK01	IDoc: Document header general data
E1EDK02	IDoc: Document header reference data
E1EDK03	IDoc: Document header date segment
E1EDK04	IDoc: Document header conditions
E1EDK05	IDoc: Document header taxes
E1EDK06	IDoc: Document header terms of delivery
E1EDK07	IDoc: Document header terms of payment
E1EDK08	IDoc: Document header currency segment
E1EDK09	IDoc: Document header bank data
E1EDK10	IDoc: Document header general foreign trade data
E1EDK11	IDoc: Document header text identification
E1EDK12	IDoc: Document header organizational data
E1EDK13	IDoc: Document item general data
E1EDK14	IDoc: Summary segment general

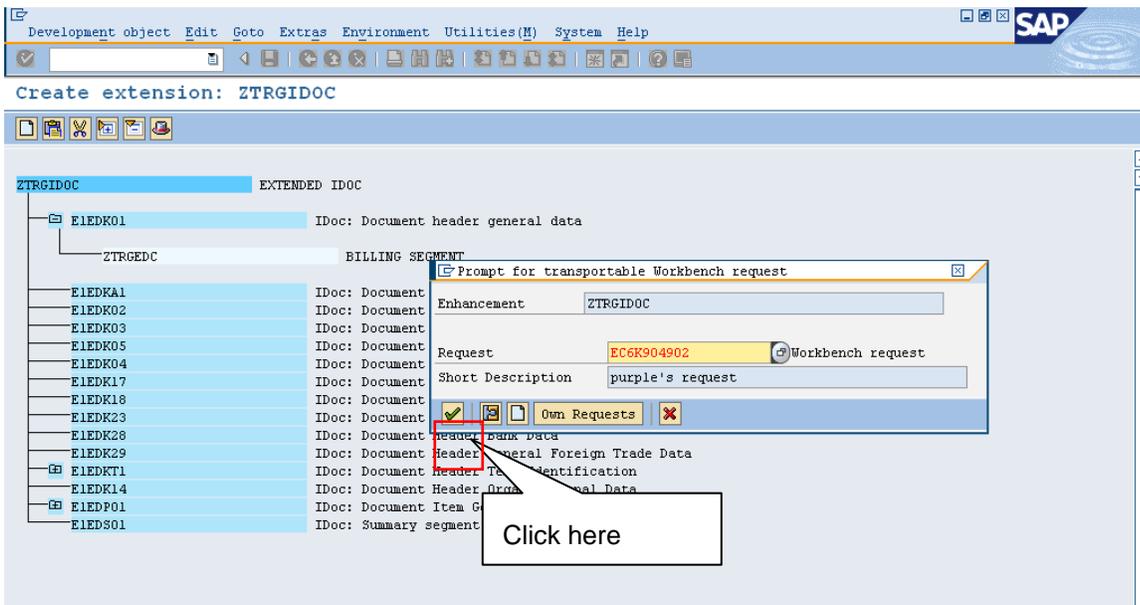
We get a popup to maintain attributes as shown

The screenshot shows the SAP 'Maintain Attributes' dialog box. The 'Segment type' is set to 'ZTRG+'. The 'Hier.level' is set to '0'. A callout box points to the 'Release-independent name of segment in EDI IDoc' field with the text 'Select this'. Below the dialog, a table lists segment types and descriptions:

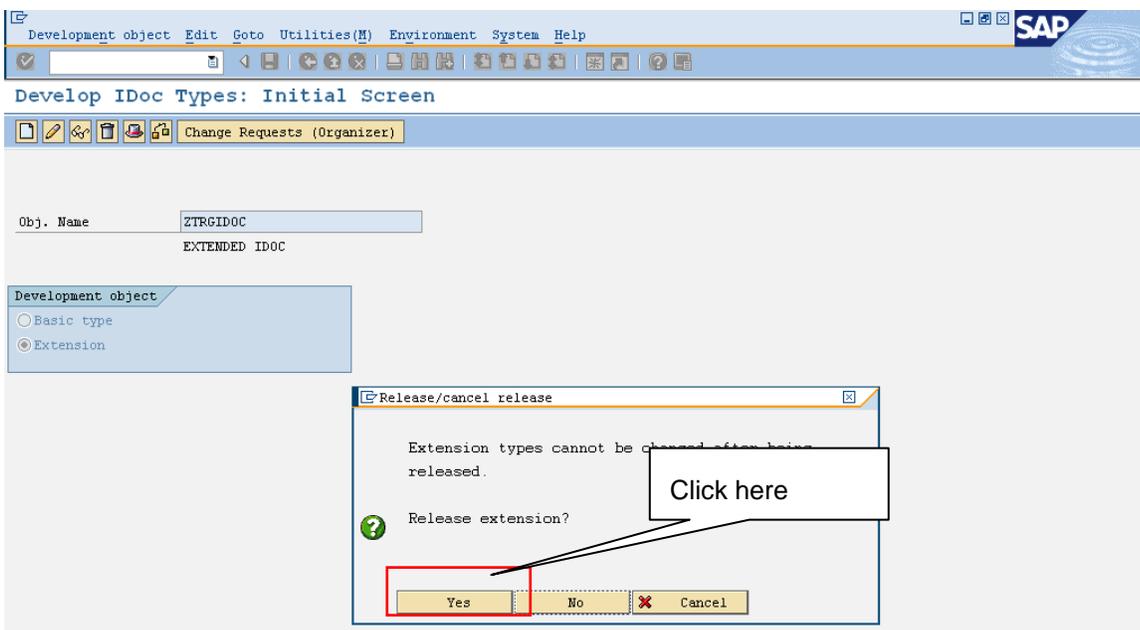
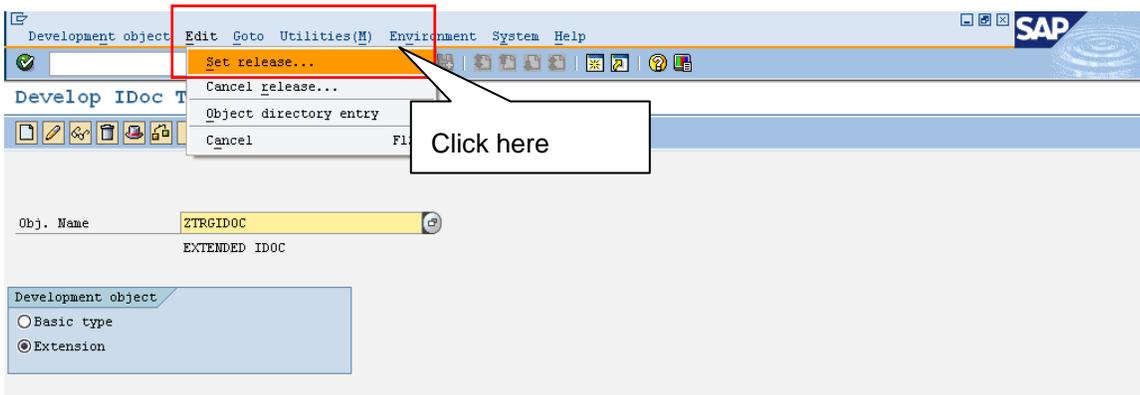
Segment type	Description
ZTRGEDC	BILLING SEGMENT

The screenshot shows the SAP 'Maintain Attributes' dialog box for segment 'ZTRGEDC'. The 'Minimum number' and 'Maximum number' are both set to '1'. The 'Hier.level' is set to '0'. Annotations include:

- A callout box pointing to the 'Mandatory seg.' checkbox with the text: 'This flag suggests whether this segment is **mandatory** in sequence'.
- A callout box pointing to the 'Minimum number' and 'Maximum number' fields with the text: 'Minimum and Maximum numbers define the minimum and Maximum number of segments in sequence'.
- A callout box pointing to the 'Hier.level' field with the text: 'Hierarchy level is maintained one more than parent else as 0'.
- A callout box pointing to the 'Segment editor' button with the text: 'Give attributes as shown and click here'.



Press on "back" and Release the IDoc as shown.



So IDoc type is created and released.

Create Logical Message Type

The message type determines the technical structure of the message. Message type determines data contained and also determines process involved in flow of message across Distributed environment. The Message Type controls Process Code, in turn drives Function Module to determine the content of the message. It also controls IDocs processing (batch, immediately etc).

There exist standard message types predefined by SAP. Some of them are

CUSTOMER – **DEBMAS** SALES ORDER – **ORDRSP**
 VENDOR – **CREMAS** PURCHASE ORDER – **ORDERS**
 MATERIAL – **MATMAS** INVOIC - **INVOICE**

We can also create the customized logical message type according to the requirements. Since we are using the invoice here we can use the existing message type INVOIC.

In transaction WE81- Logical message type we can see for message type INVOIC

The screenshot shows the SAP WE81 transaction interface. The title bar reads 'Change View "EDI: Logical Message Types": Overview'. Below the title bar is a toolbar with various icons. The main area displays a table with the following data:

Message Type	Short text
INVOIC	Invoice/Billing Document
INVRPT	
IOAUPD	work order assignments
IORDER	SM / PM order
IORUPD	Order Status Update
ISMPEX_SAVE	IS-M: BAPI Export Business Partner
ISMPEX_CHANGE	IS-M: BAPI Change Business Partner
ISMPEX_CREATE	IS-M: BAPI Create Business Partner
ISM_MATMAS	IS-M: Media product master
IST_EBS_ARC_MSG	Place document from external system in archive
IST_EBS_BUPART_MSG	Transfer Business Partner Data to External Bil
IST_EBS_COACC_MSG	Transfer Contract Account Data to External Bil
IST_EBS_POI_IN	Triggers passing on of lower items
IST_EBS_POI_OUT	Output Open Items from FI-CA
IST_EBS_REVERSE_DOC	Reversal carried out by external billing syste
IST_EBS_SDBILL	IS-T: Send SD Billing Documents to External B

A callout box points to the 'INVOIC' row with the text: 'INVOIC is the standard message type for the Billing document'. At the bottom of the screen, there is a 'Position...' field and the text 'Entry 927 of 1,943'.

Since we have a standard message type we did not create it else we can create using NEW ENTRIES tab.

Create Output Types and Assignment to IDOC Type

Go to transaction WE82- Output types and Assignment to IDOC types. Check for the entry with

Message Type	INVOIC
Basic type	INVOIC02
Extension	ZTRGIDOC

If it exists already we will use it else create it using new entries.

The first screenshot shows the 'Change View "Output Types and Assignment to IDoc Types": Overview' screen. A callout box points to the 'New Entries' button with the text 'Click on new entries'.

The second screenshot shows the 'New Entries: Overview of Added Entries' screen. A callout box points to a new entry in the table with the text 'Create a new entry as shown and save it'.

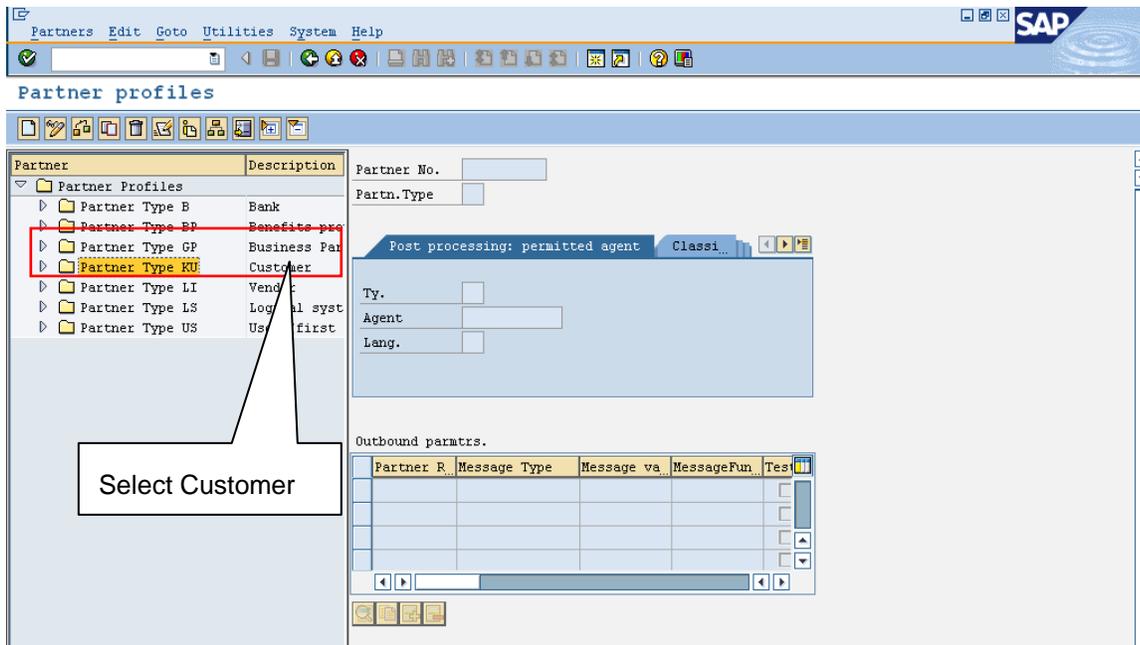
The third screenshot shows the same screen after saving. A callout box points to the entry with the text 'Saved successfully'.

Message Type	Basic type	Extension	Release
INVOIC	INVOIC02	ZTRGIDOC	700

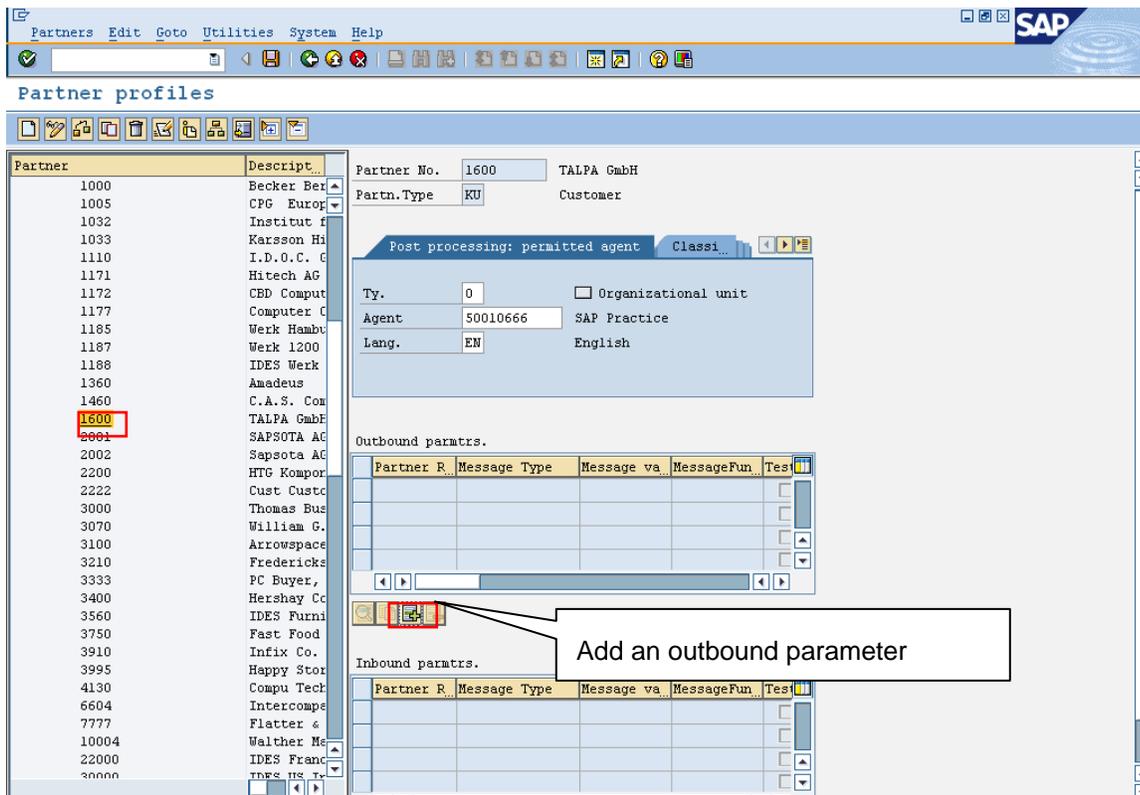
So we created Output types and Assignment to IDOC type as an entry. This is used in the IDoc processing.

Creating Partner Profile

Then go to WE20 to Creating partner profile, as it is an invoice, Select **customer**



We chose 1600 as an example, for general you can choose any customer



Outbound parameters Edit Goto System Help

Partner profiles: Outbound parameters

Partner No. 1600 TALPA GmbH
 Parth.Type KU Customer
 Partner Role BP

Message Type INVOIC
 Message code
 Message function Test

Outbound Options Message Control Post Processing: Permitted Age...

Receiver port PORT_BT

Output Mode
 Transfer IDoc Immed. Start subsystem Output Mode
 Collect IDocs Do not start subsystem

IDoc Type
 Basic type INVOIC02
 Extension ZTRCIDOC
 View
 Cancel Processing After Syntax Error
 Seg. release in IDoc type Segment Appl. Rel.

In real time, we do not use test flag in message control to send an IDoc. We used this test flag enabled only to show that we are creating the IDoc for 'testing' purpose on the outbound side.

The Message type, Partner Role, Receiver port, Basic and extended IDoc type with output mode "Transfer IDoc immediately" is given as a partner profile as shown in above document.

Outbound parameters Edit Goto System Help

Partner profiles: Outbound parameters

Partner No. 1600 TALPA GmbH
 Parth.Type KU Customer
 Partner Role BP Bill-to party

Message Type INVOIC Invoice/Billing Document
 Message code
 Message function Test

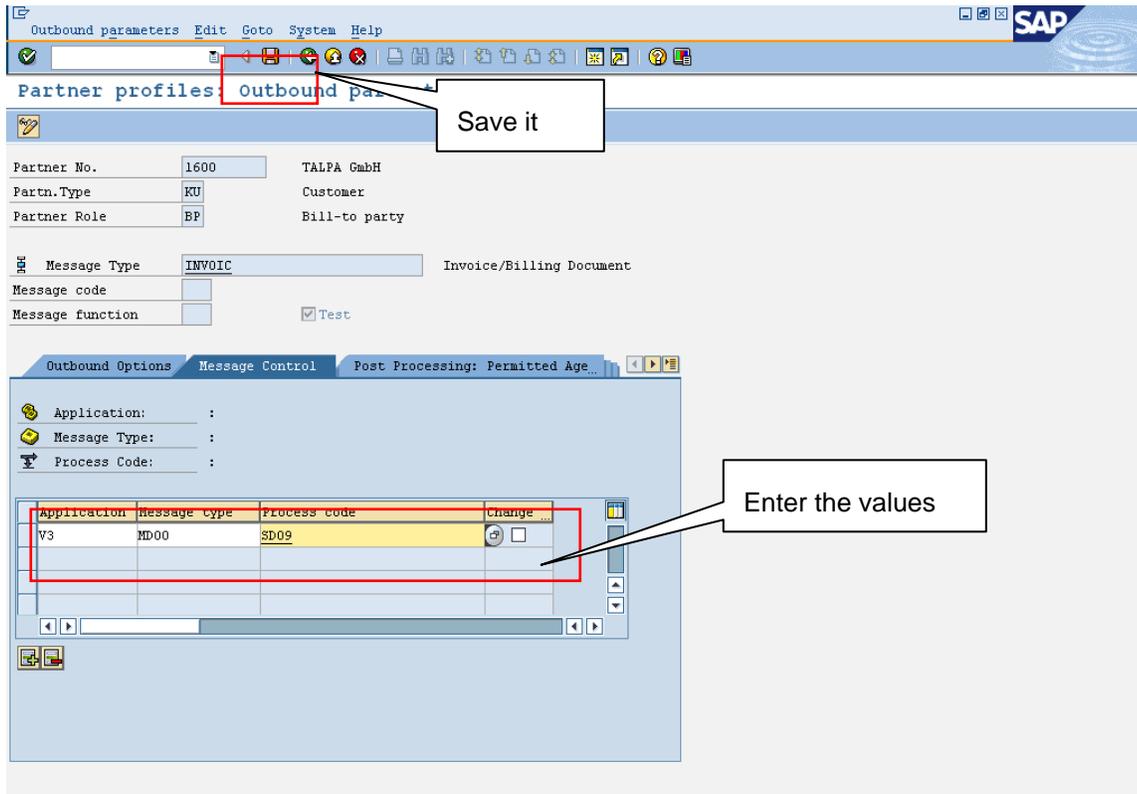
Outbound Options Message Control

Application:
 Message Type:
 Process Code:

Application	Message type	Process code	Change ...
			<input type="checkbox"/>

Click on Message Control

Click on the message control to give the values of application, Message type and process code.



These key fields application, Message type and process code assigned in the message control together uniquely identify a message type which uniquely identifies an IDoc type.

The key fields application, Message type and process code are used as

Application: The Application specified in message control determines the output type and uniquely identifies a message type which can be assigned uniquely to an IDoc type.

For Example: 'EA' is used for 'Purchasing RFQ' in Materials Management (MM).

V3' is used for 'Billing' in Sales and Distribution (SD).

Message type: Message type along with the application uniquely identifies a message type which can be assigned uniquely to an IDoc type.

For Example: 'LAVA' = 'Shipping notification' in dispatch (application 'V2').

Process code: The Process code is used by an IDoc Interface to determine the application selection module which converts the SAP document into an IDoc.

For Example: ME10: Purchase order (MM)

Here the values are chosen depending on the description having billing or invoice.

The application is **V3: Billing**

The Message type **MD00: MKS -Invoice O/P type**

The process code **SD09: INVOIC: Invoice**

We get the function module embedded in the process code where the functionality is coded. This is a **Top – Down** approach of finding the relevant function module to write User Exit. We can also find the process code and relevant function module in the **Bottom - Up** approach.

The function module which is embedded in each process code follows a naming convention **“IDOC_<OUTPUT / INPUT >_NAME OF BASIC TYPE”**. This helps in finding the process code and then the other attributes. For our example here we do an outbound with basic Idoc **INVOIC**. So the function module will be **“IDOC_OUTPUT_INVOIC”**.

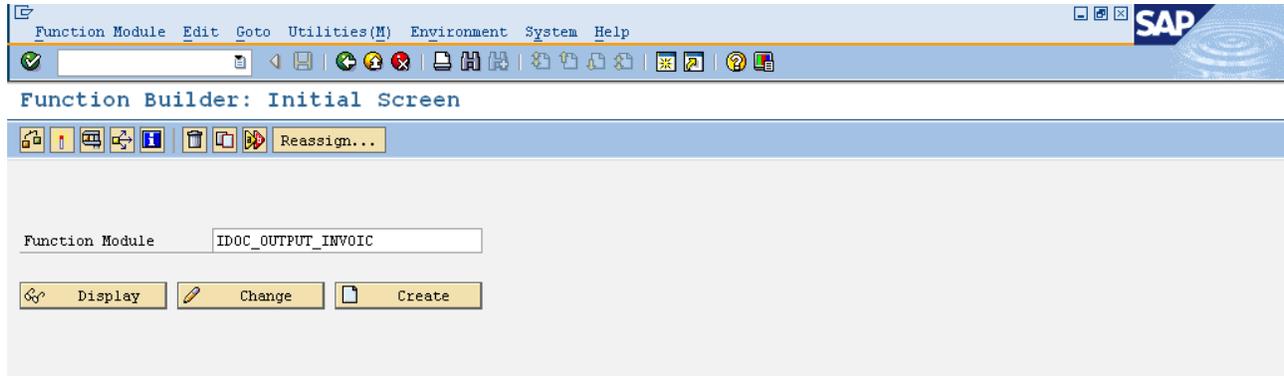
Search this in Function builder **SE37** and find the relevant process code in transaction codes **WE40** (inbound) or **WE41** (outbound). Here we followed the top approach. Many times depending on the description of the application, message type we find the relevant key fields and link them up.

But if we could not find the relevant function module then there is another method of finding it. The process is explained below using the Transactions **SMOD** and **CMOD**. We can use this method as an alternative. We choose among the **Bottom - Up**, **Top - Down** and **SMOD** depending on the scenario.

Finding and Updating Customer Exit Using SMOD

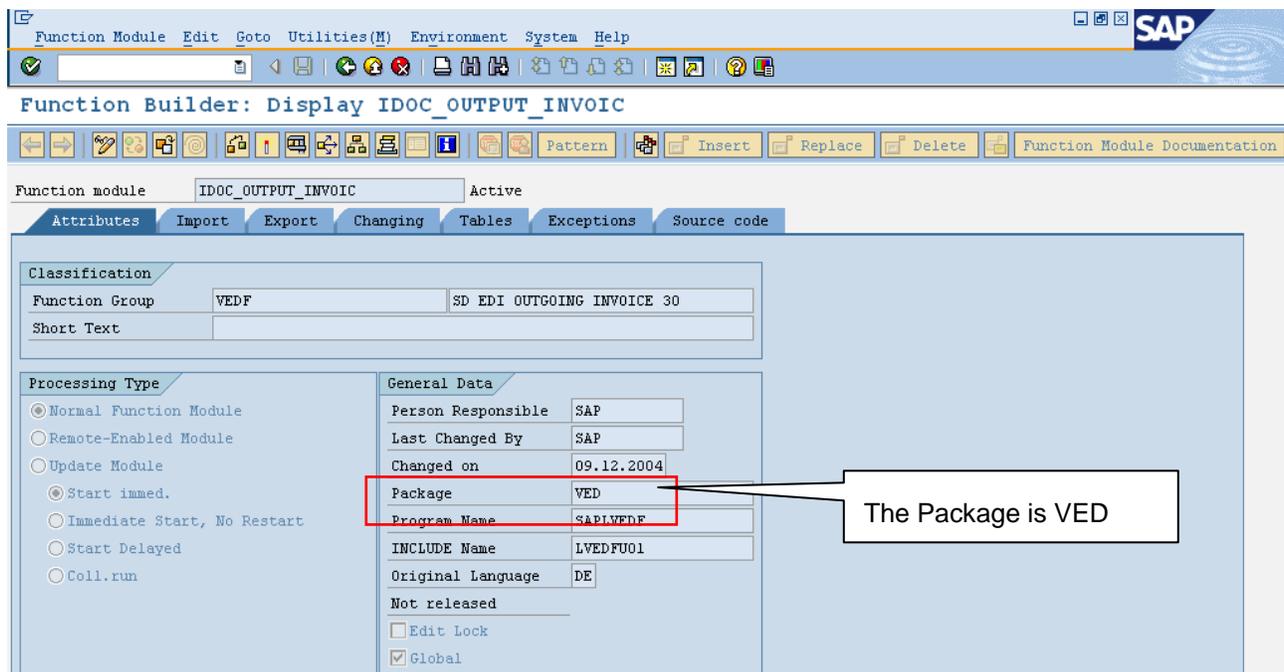
The Enhancement for the transaction can also be found using the transaction **SMOD**. Here we need to give the package to find the exact enhancement and the respective function module which serve our purpose.

So go to SE37 to find the package of "IDOC_OUTPUT_INVOIC".



After clicking on display. Go to the "Attributes" tab and look for the package name.

Here we get the Package as "VED", Use this Package name in SMOD to find the respective function module.



Go to Transaction SMOD and give the value of Package we got (here "VED") in F4 help or utilities → Find.

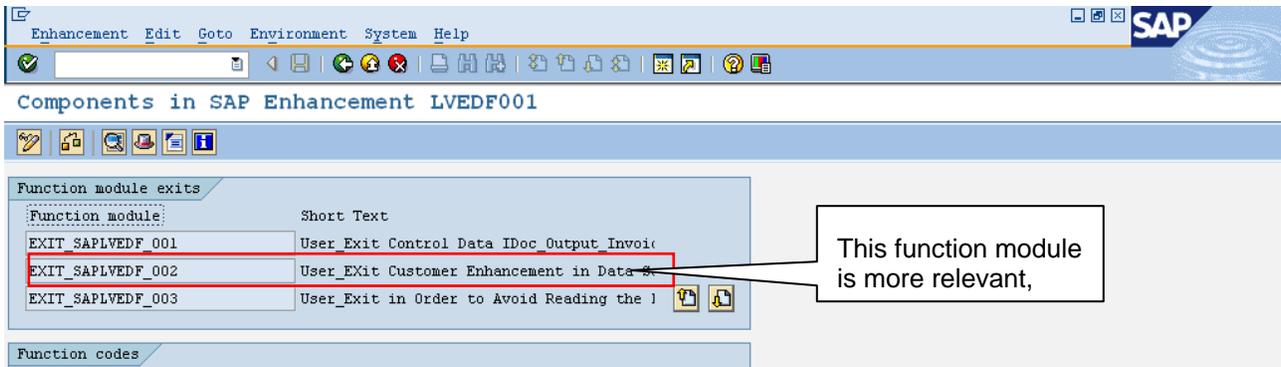
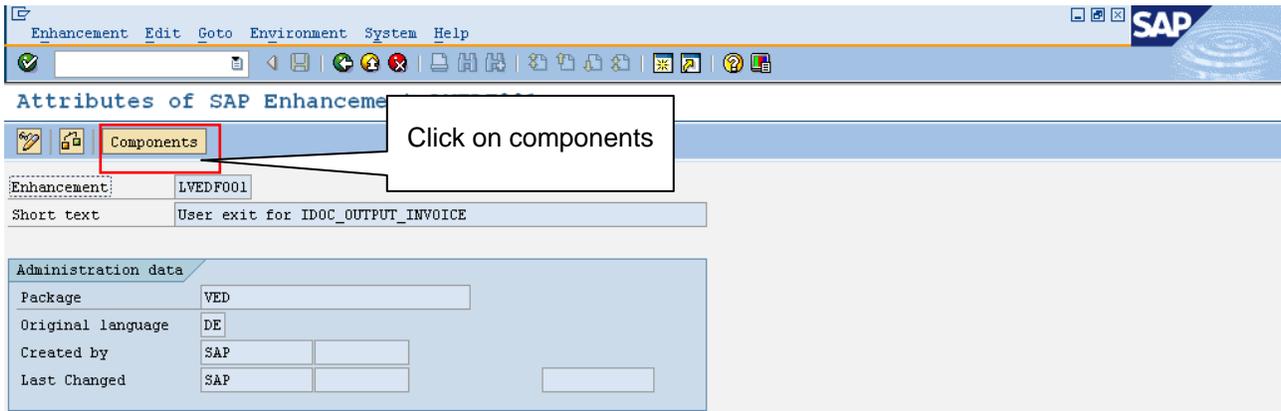
The screenshot shows the SAP SMOD transaction interface. The 'Repository Info System: Find Exits' dialog is open, displaying search criteria. The 'Package' field is set to 'VED' and is highlighted with a red box. A callout box points to this field with the text 'Give the Package VED and press enter'. Other fields include 'Exit name', 'Short text', and 'Application Component'. The 'Settings' section shows 'Maximum No. of Hits' set to 100.

We get the set of Exit names; Find the most suitable by the short text description. The description here "User Exit for Idoc infotype" is more apt. So select the exit "LVEDF001" to get the list of function modules From which we select the option which is more apt.

The screenshot shows the 'Repository Info System: Exits Find (9 Hits)' dialog. A table lists the search results:

Exit name	Short text
LVEDF001	User exit for IDOC_OUTPUT_INVOICE
SDEDI001	User exits for EDI
SDEDI002	Customer Enhancements for Object Type IDOCORDERS
VED40001	EDI supply
VED50001	EDI Component Supplier Processing:Self-Billing Procedur
VEDA0001	SD EDI Incoming Orders (Customer Extensions)
VEDE0001	SD EDI incoming Change Orders (Customer Extensions)
VEDE0001	SD EDI Outbound quotation (customer enhancement)?
VEDQ0001	SD EDI Inbound Inquiries (customer enhancements)

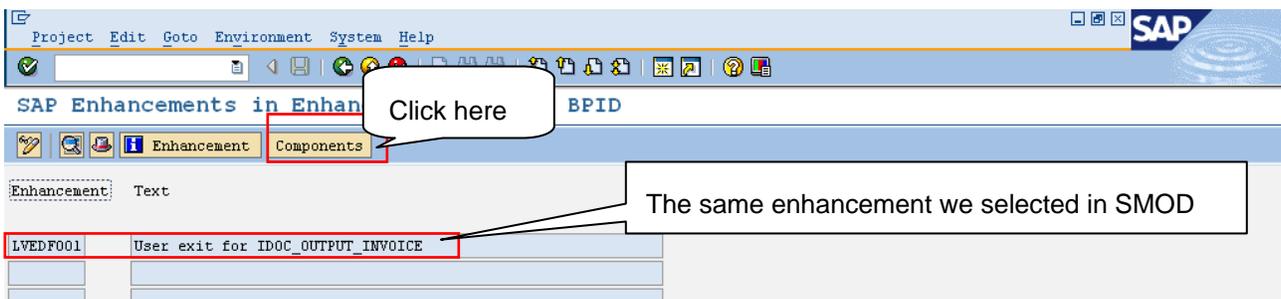
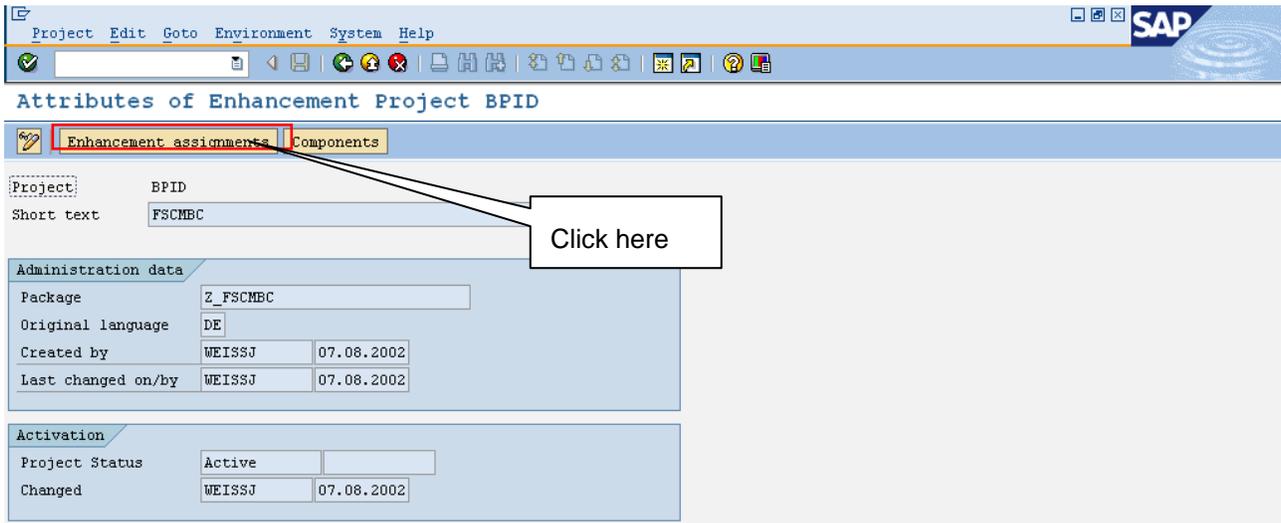
The first entry, 'LVEDF001 User exit for IDOC_OUTPUT_INVOICE', is highlighted with a red box. A callout box points to this entry with the text 'Select the exit which is more relevant'.



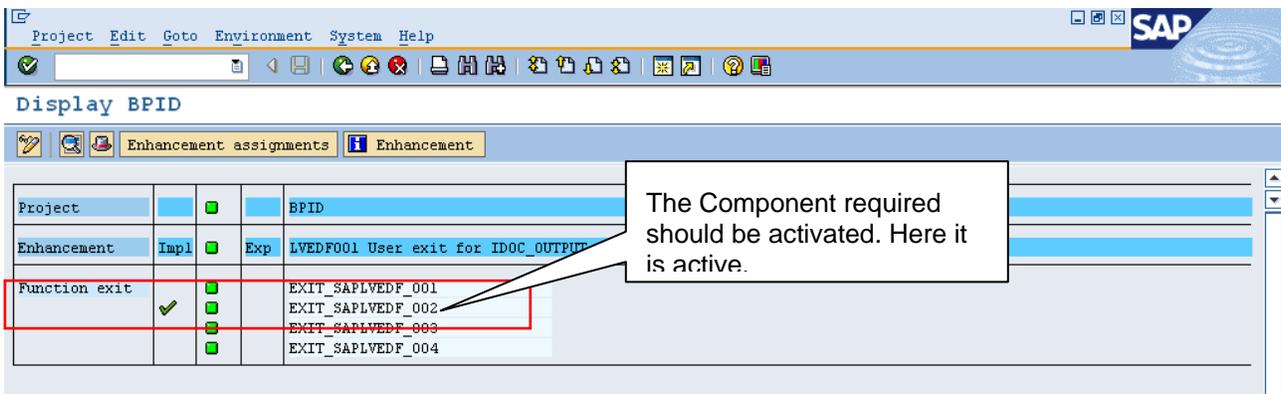
This is the function module where we are going to add the functional change.

Go to CMOD and give "BPID" standard enhancement or you can create a new project with 'LVEDF001'





The Enhancement is having our component active.



You can Click here to directly go to the function “exit_saplvdef_002” or follow the path as shown below.

You need to add the changes required in this customer exit. The process is explained in detail below.

So the process using the Transactions **SMOD** and **CMOD** is explained above. Now we need to make the changes in the code according to the requirement.

Outbound Process Code

We found the relevant user exit using SMOD and SMOD as "exit_saplvdf_002".

But in Top-Down or Bottom - Up we can find the relevant user exit do the following steps.

Go to transaction WE41 - Outbound process code. DOUBLE CLICK On SD09 as shown

Display View "Outbound process code": Overview

Outbound process cod	Description of process
MRKO	Consignment settlement for invoice verification
MRNE	New evaluation (Retro Active Pricing)
MRRL	ERS procedure for Logistics invoice verification
OPOD	STPPOD: POD confirms proof of delivery outbound
PJNO	Generation/export of IDOC orders03 for JIT call
QCERT_OUT	Fill IDoc Quality01
SD05	DESADV: Shipping notification (DESADV01)
SD06	DESADT: Transportation notification (DESADV01)
SD08	INVOIC: Internal allocation
SD09	INVOIC: Invoice
SD10	ORDRSP: Sales order confirmation
SD11	SHPMNT: Transport
SD12	QUOTES: Quotation
SHIPNOTIF_DLR	

Position... Entry 13 of 52

Display View "Outbound process code": Details

Process code: SD09

Description: INVOIC: Invoice

Function module: IDOC_OUTPUT_INVOIC

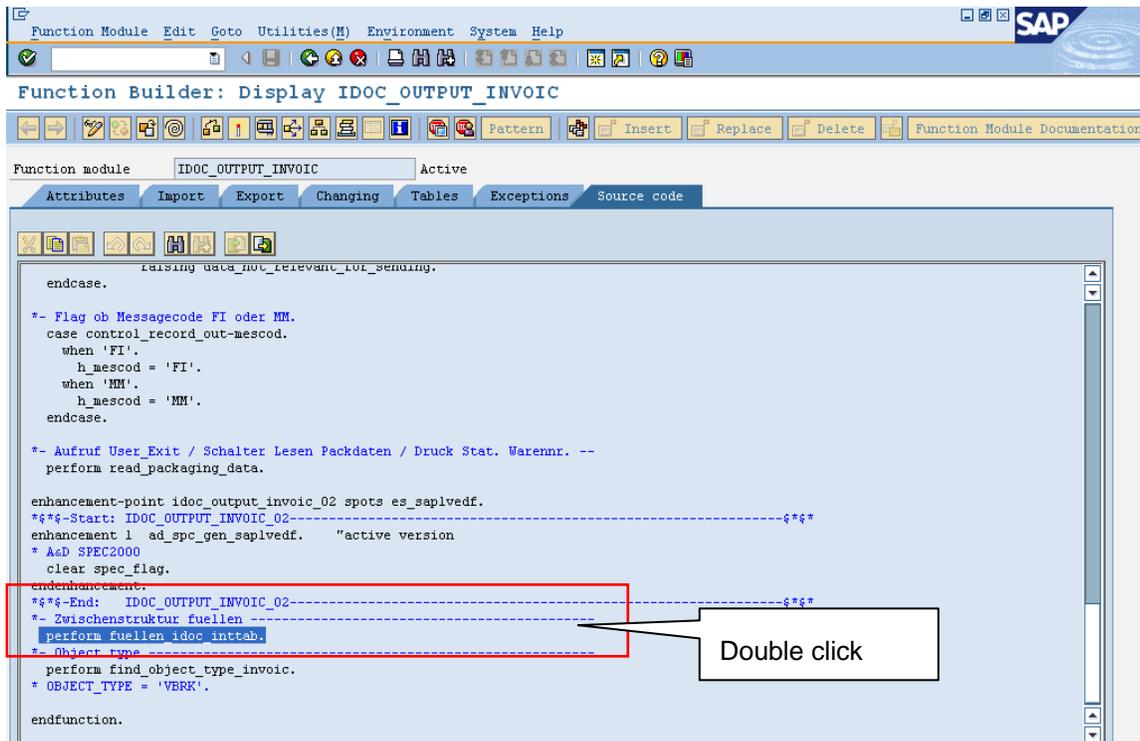
Option ALE-Service/Inb. procg

- Processing with ALE service
- Processing w/o ALE service
- Processing w. trigger (inbound)

Version of function module

- Processing with function module version 3.0
- Processing with function module version 2.2

We can get into the function module to find the customer exit. So after getting into the function module how you find the relevant customer exit is defined below. Double click on the function module to get to the next screen.



You will find a customer function for control record "customer-function '001'" which is not useful for us. Find the perform which normally come at the end of the Function module having a naming convention starting with "Fill" or "Fuellen". This is shown in the above screen shot.

ABAP Editor: Display Include LVEDFF0E

```

form fuellen_idoc_inttab.
data: lv_prog type localfile.
-----*
*           K O P F - D A T E N
-----*
*-Rechnungskopf - Standard-Daten
perform fill_e1edk01.

*-Rechnungskopf - Partnerinfo - Rechnungssteller
perform fill_e1edk01
using 'PS'.

*-Rechnungskopf - Partnerinfo - All
* (--> fill EIEDKAZ Rechnungskopf - Partnerinfo - Zusatzinfos (SCACode))
perform fill_e1edk01_all.

*-Rechnungskopf - Partnerinfo - Finanzdokumentpartner (AA, AW, OB)
perform fill_e1edk01_fd.

* EAN.UCC - Start 25.07.03 - because of IDOC sequence
*-Rechnungskopf - Partnerinfo - ILN for vendor (out of vendor view)
lv_prog = 'SAPLVEDF'.
perform fill_e1edk01_lfl in program (lv_prog) if found.
* EAN.UCC - END

*-Rechnungskopf - Referenzen
perform fill_e1edk02.

*-Rechnungskopf - Datumsfelder
perform fill_e1edk03.

*-Kopf-Konditionen werden noch benoetigt
perform get header prices.

```

Since we are adding our segment to **e1edk01** find the perform having this string. Here it is **perform fill_e1edk01**. Double click on it.

ABAP Editor: Display Include LVEDFF0F

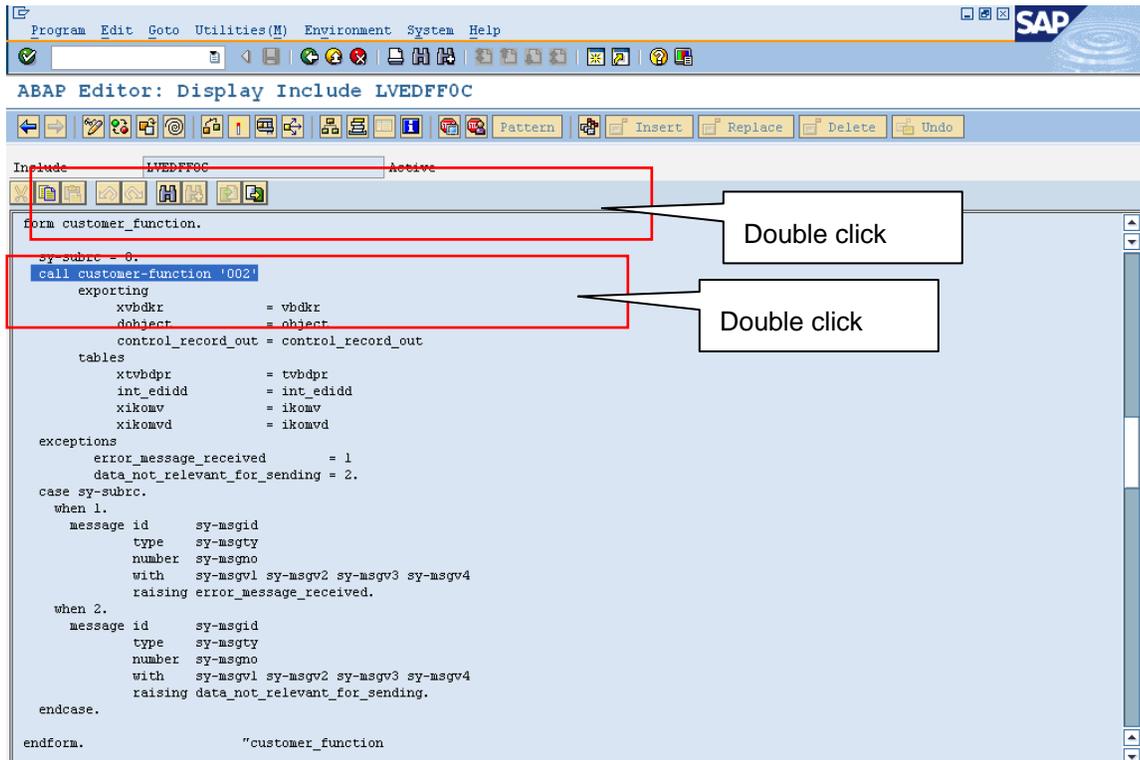
```

else.
perform move_field using 'E1EDK01-WKURS' vbdkr-kurrf '.
perform move_field using 'E1EDK01-WKURS_M' lv_wkurs_m '.
endif.
endif.
perform move_field using 'E1EDK01-HWAER' h_t001 waers '.
perform move_field using 'E1EDK01-WKURS' vbdkr-kurrf '.
perform move_field using 'E1EDK01-ZTERM' vbdkr-zterm '.
perform move_field using 'E1EDK01-NTGEW' vbdkr-ntgew '.
perform move_field using 'E1EDK01-BRGEW' vbdkr-brgew '.
if not vbdkr-gewei is initial.
*- clear für Fehlermeldung in form iso_code_unit
clear tvbdpr.
h_gewei = vbdkr-gewei.
perform iso_code_unit using h_gewei.
perform move_field using 'E1EDK01-GEWEI' h_gewei '.
endif.
perform move_field using 'E1EDK01-BSTZD' vbdkr-bstzd '.
perform move_field using 'E1EDK01-VSART' vbdkr-vsart '.
perform move_field using 'E1EDK01-VSART_BEZ' vbdkr-vsart_bez '.
perform move_field using 'E1EDK01-KUNDEUINR' vbdkr-stceg '.
perform move_field using 'E1EDK01-EIGENUINR' vbdkr-stceg_t001 '.
move e1edk01 to int_edidd-sdata.
append int_edidd.
perform customer_function.
endiform. "fill_e1edk01

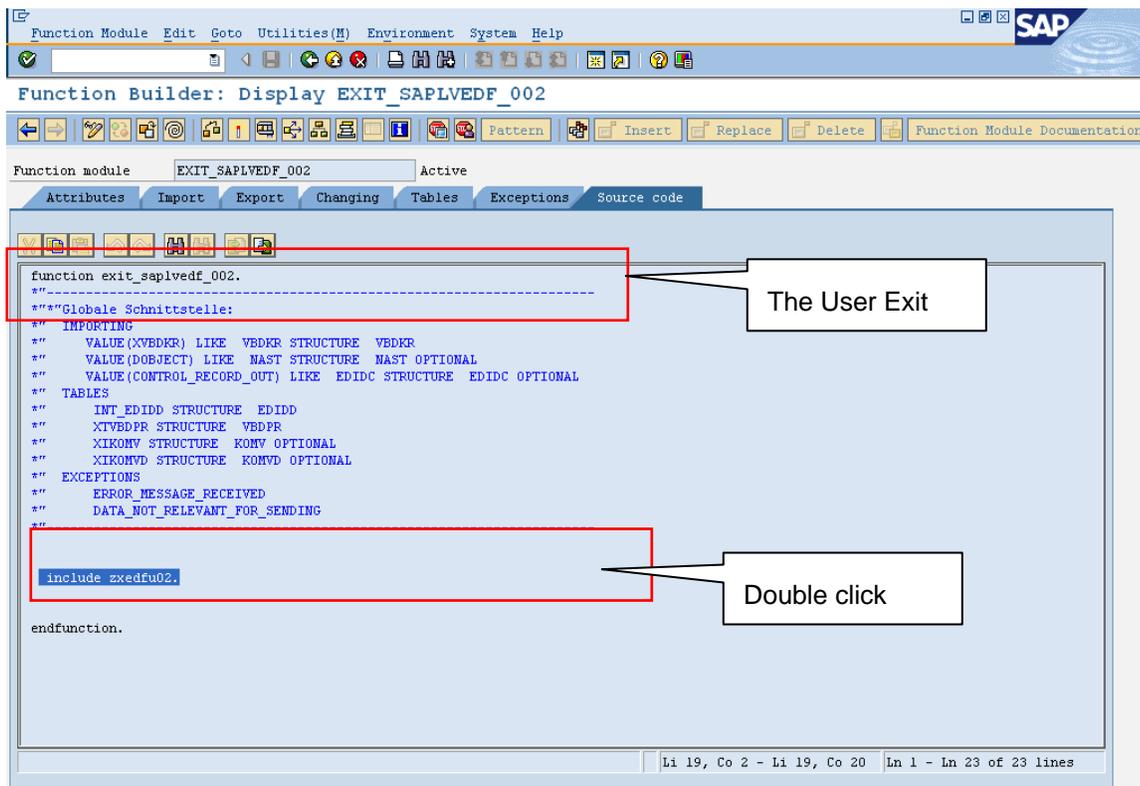
-----*
*           F O R M   F I L L   E I E D K 1 4
-----*
*           Zwischenstruktur für die Kopf-Organisationsdaten.
-----*

```

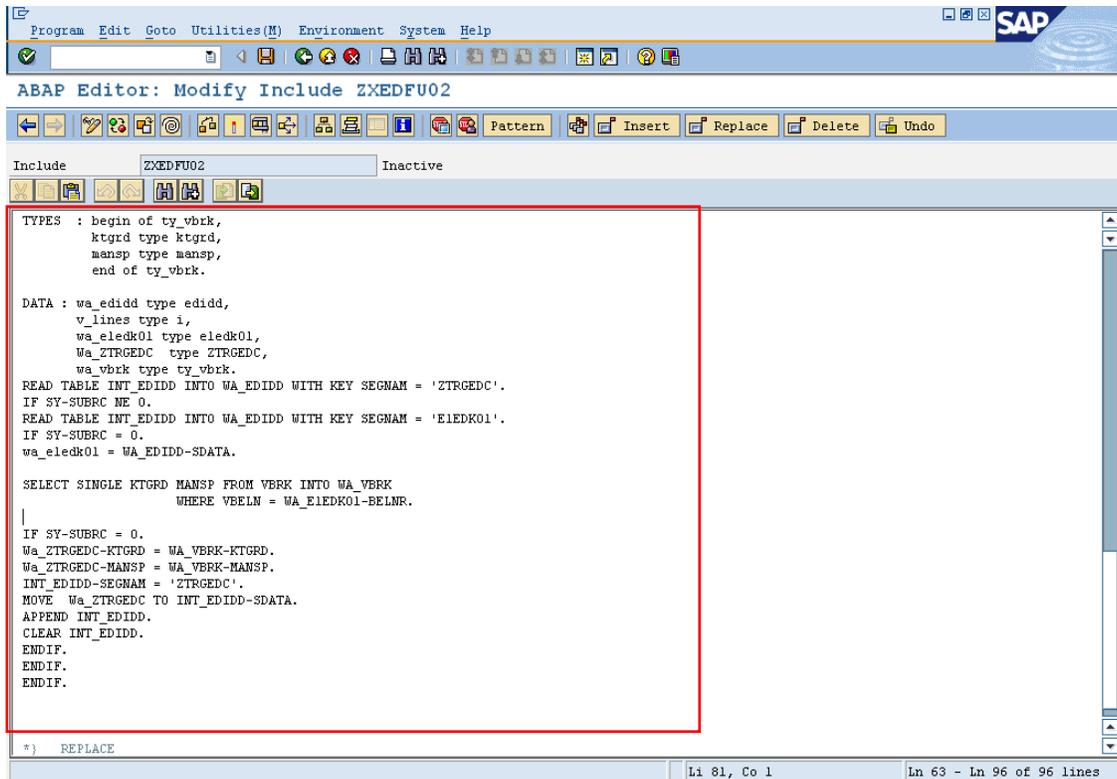
At the end of it you will have perform with name **"perform customer_function."** Double click it. You will get the **"customer-function '002'"** which is the required customer function. Double click it.



We will get the required customer function “**function exit_saplvdef_002**” with an include “**include zxedfu02**”. This will lead you to the customer exit where we write the required changes.



We will write the code as shown below.



The screenshot shows the SAP ABAP Editor interface. The title bar reads 'ABAP Editor: Modify Include ZXEDFU02'. The editor window contains the following code:

```

TYPES : begin of ty_vbrk,
        ktgrd type ktgrd,
        mansp type mansp,
        end of ty_vbrk.

DATA : wa_edidd type edidd,
        v_lines type i,
        wa_eledk01 type eledk01,
        wa_ztrgedc type ztrgedc,
        wa_vbrk type ty_vbrk.

READ TABLE INT_EDIDD INTO WA_EDIDD WITH KEY SEGNAM = 'ZTRGEDC'.
IF SY-SUBRC NE 0.
READ TABLE INT_EDIDD INTO WA_EDIDD WITH KEY SEGNAM = 'ELEDK01'.
IF SY-SUBRC = 0.
wa_eledk01 = WA_EDIDD-SDATA.

SELECT SINGLE KTGRD MANSP FROM VERK INTO WA_VBRK
           WHERE VBELN = WA_ELEDK01-BELNR.
|
IF SY-SUBRC = 0.
Wa_ztrgedc-KTGRD = WA_VBRK-KTGRD.
Wa_ztrgedc-MANSP = WA_VBRK-MANSP.
INT_EDIDD-SEGNAM = 'ZTRGEDC'.
MOVE Wa_ztrgedc TO INT_EDIDD-SDATA.
APPEND INT_EDIDD.
CLEAR INT_EDIDD.
ENDIF.
ENDIF.
ENDIF.

```

At the bottom of the editor, there is a status bar showing 'Li 81, Co 1' and 'Ln 63 - Ln 96 of 96 lines'. A small asterisk and the word 'REPLACE' are visible in the bottom left corner of the editor area.

Please find the code below.

```
TYPES : begin of ty_vbrk,
        ktgrd type ktgrd,
        mansp type mansp,
        end of ty_vbrk.

DATA : wa_edidd type edidd,
        v_lines type i,
        wa_e1edk01 type e1edk01,
        Wa_ZTRGEDC type ZTRGEDC,
        wa_vbrk type ty_vbrk.

READ TABLE INT_EDIDD INTO WA_EDIDD WITH KEY SEGNAM = 'ZTRGEDC'.
IF SY-SUBRC NE 0.
READ TABLE INT_EDIDD INTO WA_EDIDD WITH KEY SEGNAM = 'E1EDK01'.
IF SY-SUBRC = 0.
wa_e1edk01 = WA_EDIDD-SDATA.

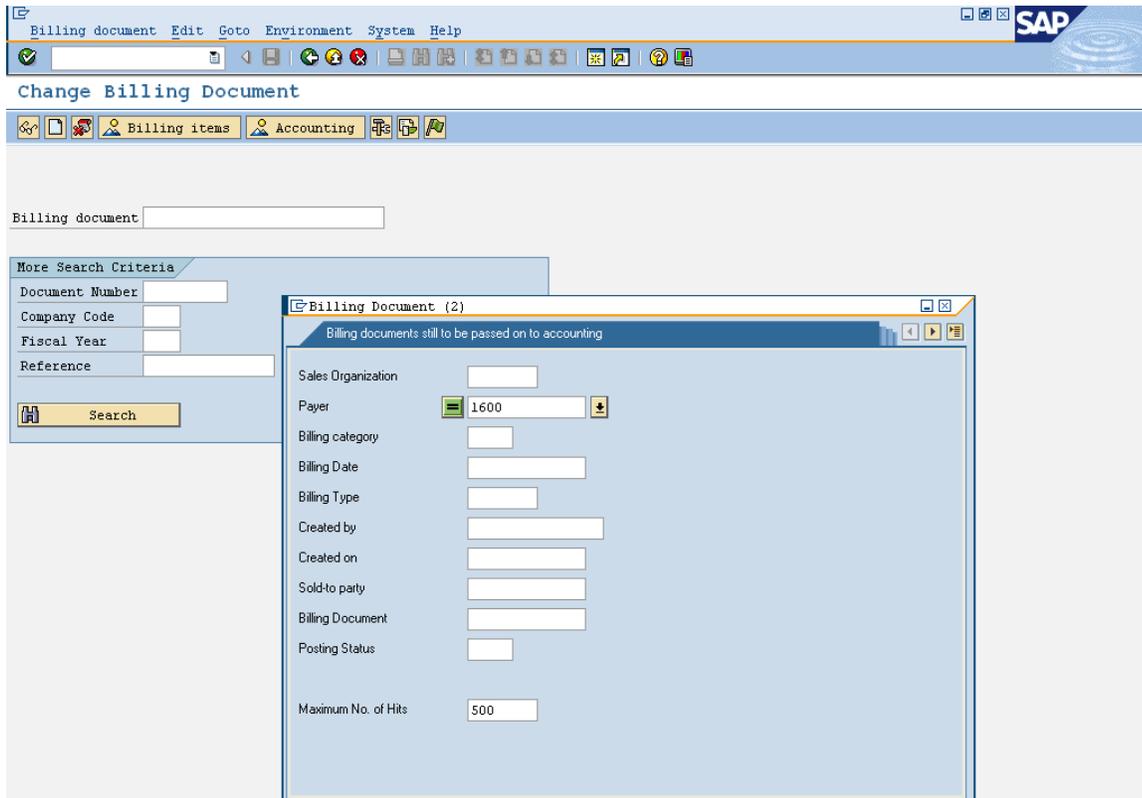
SELECT SINGLE KTGRD MANSP FROM VBRK INTO WA_VBRK
           WHERE VBELN = WA_E1EDK01-BELNR.

IF SY-SUBRC = 0.
Wa_ZTRGEDC-KTGRD = WA_VBRK-KTGRD.
Wa_ZTRGEDC-MANSP = WA_VBRK-MANSP.
INT_EDIDD-SEGNAM = 'ZTRGEDC'.
MOVE Wa_ZTRGEDC TO INT_EDIDD-SDATA.
APPEND INT_EDIDD.
CLEAR INT_EDIDD.
ENDIF.
ENDIF.
ENDIF.
```

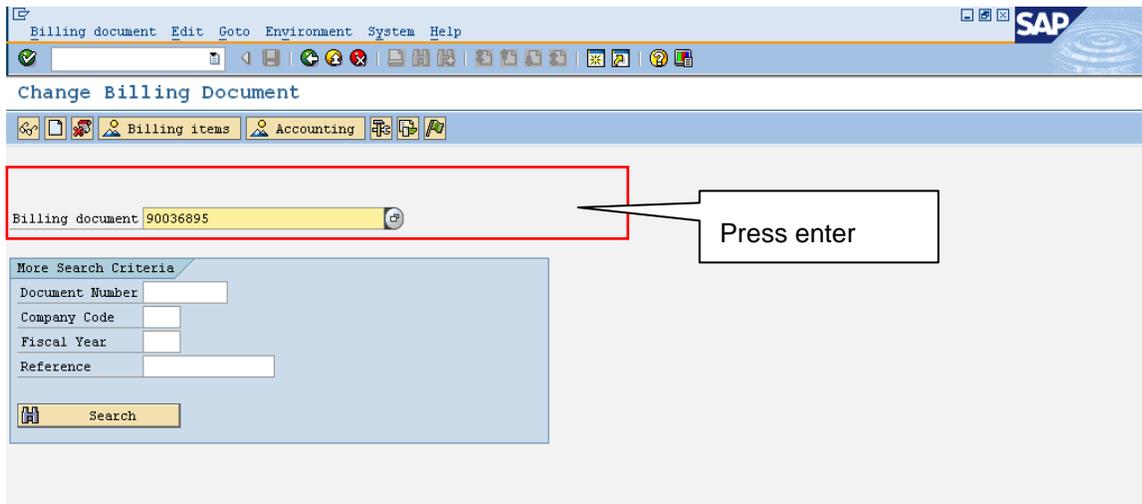
Save and activate it

Create a Transaction and Send IDoc

Go to transaction VF02 and search the bill for customer "1600" (we chose it)



Select a bill as shown and press enter.



Invoice (F2) 90036895

Accounting overview Billing items overview

Invoice (F2) 90036895

Payer: 1600 TALPA GmbH / An der Breiten Wiese 42

Billing Date: 15.05.2008

Item	Description	Billed Quantity	SU	Net value	Material	Tax amount
10	Steuereinheit / 220V 50Hz	1	PC	200.00	E-1517	0.00
20	Steuereinheit / 220V 50Hz	1	PC	400.00	E-1517	0.00
30	Steuereinheit / 220V 50Hz	1	PC	300.00	E-1517	0.00

Output Edit Goto System Help

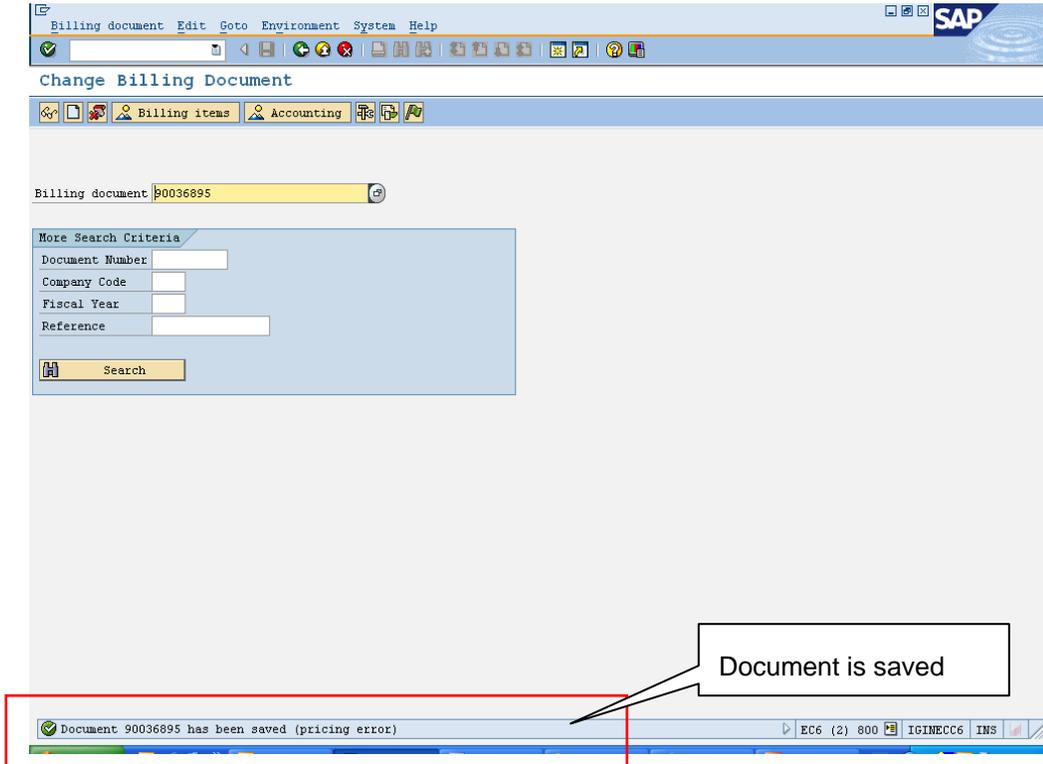
Invoice (F2) 90036895 Save (Ctrl+S) nge: output

Communication method Processing log Further data Repeat output Change output

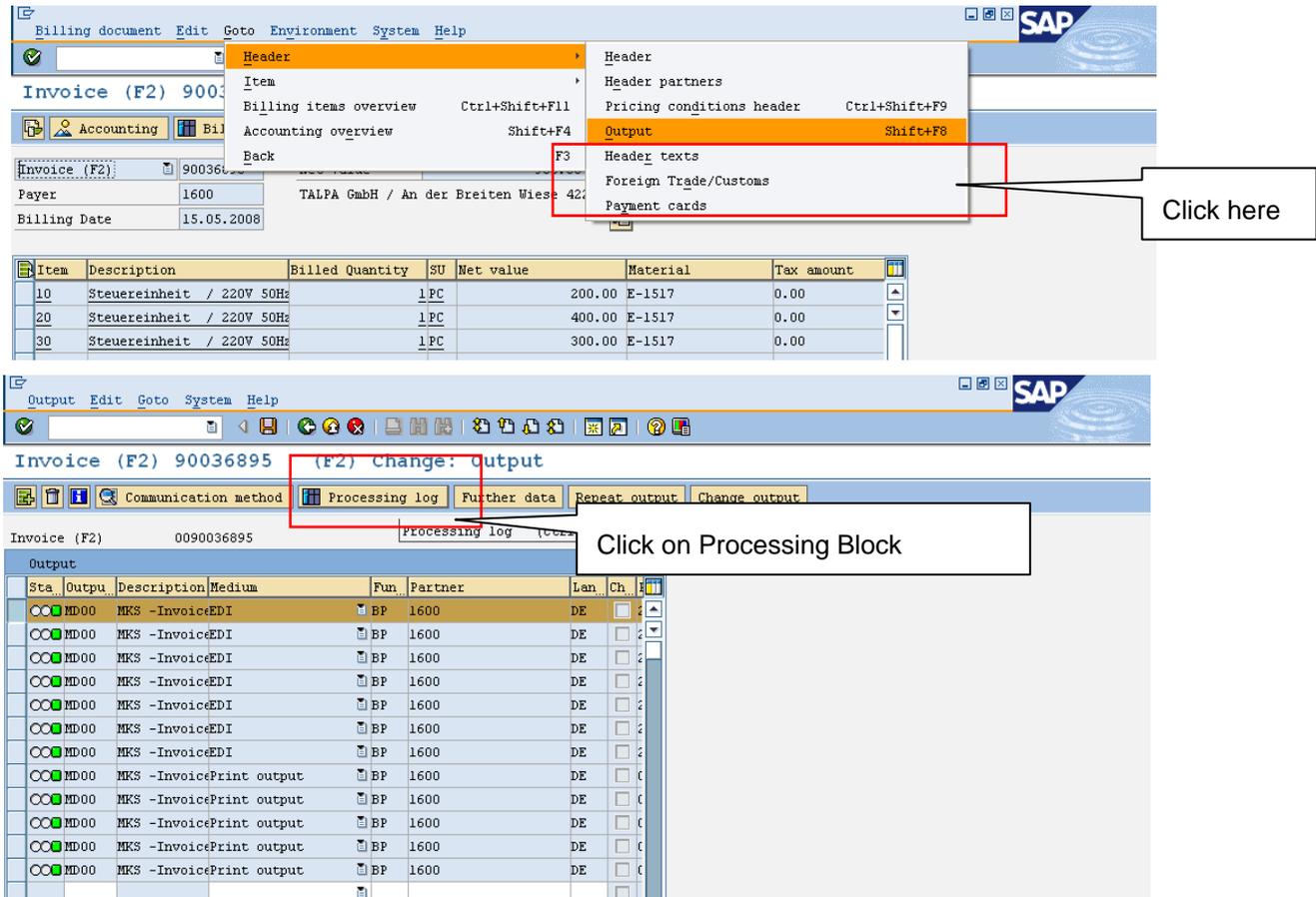
Invoice (F2) 0090036895

Sta	Output	Description	Medium	Fun	Partner	Lan	Ch
<input type="checkbox"/>	MD00	MKS - InvoiceEDI		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>
<input type="checkbox"/>	MD00	MKS - InvoiceEDI		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>
<input type="checkbox"/>	MD00	MKS - InvoiceEDI		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>
<input type="checkbox"/>	MD00	MKS - InvoiceEDI		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>
<input type="checkbox"/>	MD00	MKS - InvoiceEDI		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>
<input type="checkbox"/>	MD00	MKS - InvoiceEDI		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>
<input type="checkbox"/>	MD00	MKS - InvoiceEDI		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>
<input type="checkbox"/>	MD00	MKS - InvoicePrint output		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>
<input type="checkbox"/>	MD00	MKS - InvoicePrint output		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>
<input type="checkbox"/>	MD00	MKS - InvoicePrint output		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>
<input type="checkbox"/>	MD00	MKS - InvoicePrint output		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>
<input type="checkbox"/>	MD00	MKS - InvoicePrint output		<input type="checkbox"/>	BP 1600	DE	<input type="checkbox"/>

Give these values and click on save



Once again go to the same screen to find the IDOC no.



Invoice (F2) 90036895 (F2) Change: Output

Communication method Processing log Further data Repeat output Change output

Invoice (F2) 0090036895

Sta.	Output	Description	Medium	Fun.	Partner	Lan.	Ch.
MD00	MKS -InvoiceEDI			BP	1600	DE	

Output Processing analysis for proc. mks billing proce.

Type	Message text
Object	0090036895
Output type:	MKS -Invoice O/P typ
Processing log for program	RSMAS TED routine EDI_PROCESSING
IDoc	000000000819906 was created and forwarded for transmission

Idoc details are shown here

Here the IDoc is 000000000819906

Monitoring IDoc

In WE02 transaction gives the details as shown and finds the generated IDoc.

Program Edit Goto System Help

IDoc List

Default Additional EDI

Created At	00:00:00	to	24:00:00	
Created On	26.06.2009	to	26.06.2009	
Last Changed at	00:00:00	to	24:00:00	
Last Changed on		to		
Direction				
IDoc Number	819906	to		
Current Status		to		
Basic Type		to		
Enhancement		to		
Logical Message		to		
Message Variant		to		
Message Function		to		
Partner Port		to		
Partner Number		to		
Partner Type		to		
Partner Role		to		

The screenshot shows the SAP IDoc display interface. On the left, a tree view shows the IDoc structure: IDoc 000000000819906, Control Rec., Data records, Status records, 03, IDoc sent to SAP system, 30, Receiver exists, No fi., and 01. A red box highlights the '03' status record. On the right, the 'Technical short info' panel shows: Direction 1 (Outbox), Current status 03, Basic type INVOIC02, Extension ZTRGIDOC, Message type ZTRGMSG, Partner 1600, Partn.Type KU, and Port PORT_BT. A callout box points to the 'Current status 03' field with the text: 'The Status is '03' which means Idoc is successful'.

The screenshot shows the SAP IDoc display interface with the 'Content of selected segment' table. The table has two columns: 'Fld name' and 'Fld cont.'. The data rows are: KTGRD 01 and MANSP A. A red box highlights this table. A callout box points to the table with the text: 'The Extended Idoc data'.

So the Extended IDOC is Created and successfully posted from Outbound.

In most cases basic Idoc will be suitable for your applications and if it is lacking with few set of fields then you can extend the Idoc by inserting with the fields you required. This helps the easy usage of the existing standard Idoc with little customization.

Related Content

[Extending an IDoc Type](#)

[Extension of IDoc types and Processing](#)

[Idoc homepage](#)

For more information, visit the [ABAP homepage](#).

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.