Entire ALE Examples

1.1.1. SUBHENDU MAJUMDAR

Technical Consultant, IBM

Contents

1.

Preface

The purpose of this document is to help the ABAPers get an idea about how to set up all the necessary configurations to send IDoc from one system to another. After studying all the theoretical aspects of ALE/IDoc, a developer often wonders how to start and where to start. This document will provide them a good starting point.

Scope of scenarios on EDI systems is outside the scope of this document. Also, sending Idoc by message control mechanism is not described here. Not there is any discussion of sending Idoc using BAPI.

All the scenarios discussed over here cover almost 90% of the development/configuration requirements we receive from the client. I plan to modify this document with additional scenarios as I encounter them in future.

I will be happy if this document mentors you at the time of your requirement. Contact me at **subhendu_mj@hotmail.com** in case you have any queries.

2. Client to Client ALE Setup

2.1. Introduction

ALE technology is used to transfer information from one SAP R/3 to another R/3. Here, information on vendor master is being transferred from SAP system: Shatadru, client 555 to SAP system: Shatadru, client 777. All the necessary configurations and settings required are shown below along with adequate screen shots.

This documentation assumes that the reader is already acquainted with the tools and terms of ALE:-

Logical System
RFC Destination
Customer Distribution Model
Port
Partner Profile.

The purpose of the documentation is to get one beginner a head-start , where he can see how ALE setup is done in SAP R/3 system .

2.2. Steps

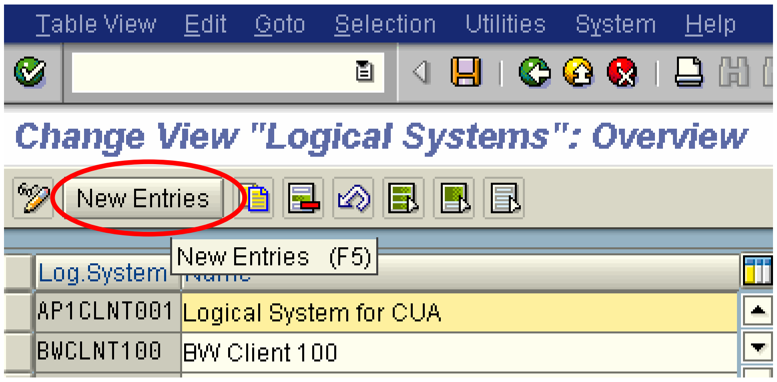
The steps to be followed consecutively to accomplish the mission are detailed below with adequate screenshots.

2.2.1. Defining Logical System

In	Shatadru 555
Logical systems	SEND555 RECEIVE777
Tcode	BD54
Purpose	To create logical systems for the two SAP systems, between which vendor master information will be shared.

Procedure:-

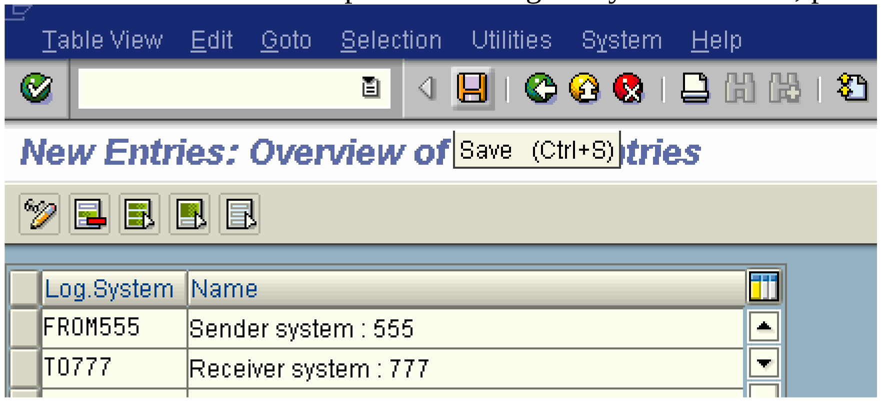
Go to transaction BD54 . Press the pushbutton as shown below to create new logical systems.



Change View "Logical Systems": Overview

Log.System	Name
AP1CLNT001	Logical System for CUA
BWCLNT100	BW Client 100

Enter the name and description of the logical systems . Then , press **Save.**



New Entries: Overview of [Save (Ctrl+S)]tries

Log.System	Name
FROM555	Sender system : 555
TO777	Receiver system : 777

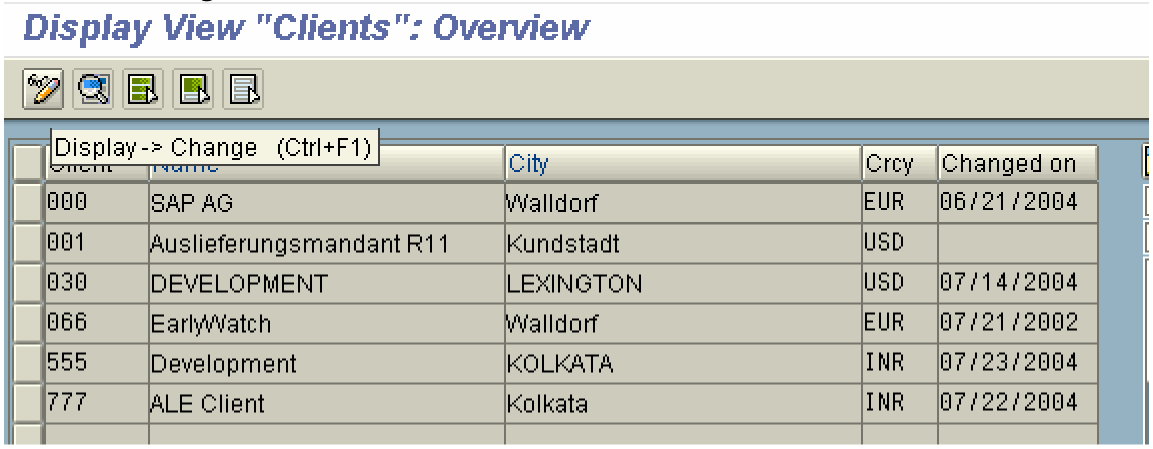
Please remember, this is an one-time activity. Logical system is built only once for two SAP systems involved.

2.2.2. Assign Logical System to Client

In	Shatadru 555
Logical systems	FROM555 assigned to client 555 TO777 assigned to client 777
Tcode	SCC4
Purpose	To assign the logical systems to the client.

Process

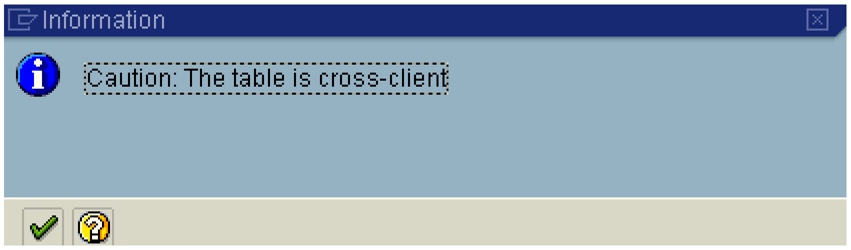
Enter into change mode.



Display View "Clients": Overview

Display -> Change (Ctrl+F1)

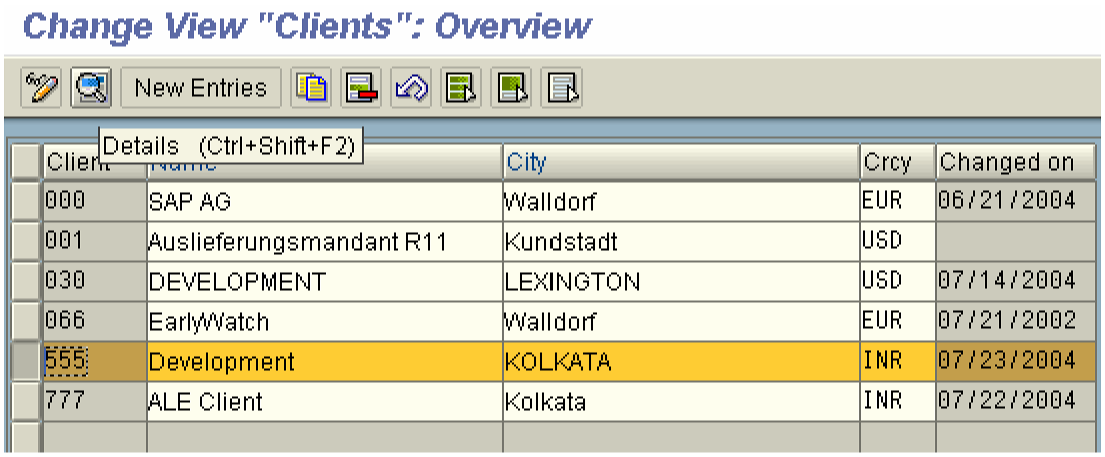
Client	Name	City	Crcy	Changed on
000	SAP AG	Walldorf	EUR	06/21/2004
001	Auslieferungsmandant R11	Kundstadt	USD	
030	DEVELOPMENT	LEXINGTON	USD	07/14/2004
066	EarlyWatch	Walldorf	EUR	07/21/2002
555	Development	KOLKATA	INR	07/23/2004
777	ALE Client	Kolkata	INR	07/22/2004



Information

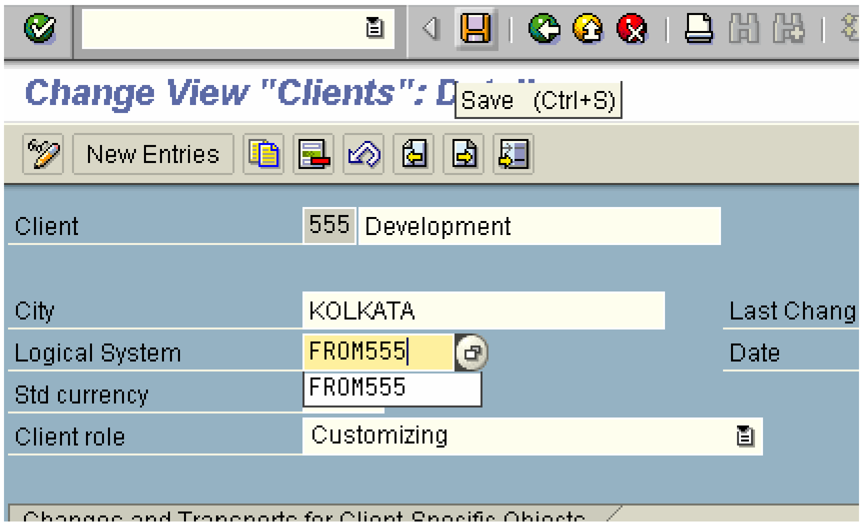
Caution: The table is cross-client

Select a client and choose: **Details.**



Change View "Clients": Overview

Client	Name	City	Crcy	Changed on
000	SAP AG	Walldorf	EUR	06/21/2004
001	Auslieferungsmandant R11	Kundstadt	USD	
030	DEVELOPMENT	LEXINGTON	USD	07/14/2004
066	EarlyWatch	Walldorf	EUR	07/21/2002
555	Development	KOLKATA	INR	07/23/2004
777	ALE Client	Kolkata	INR	07/22/2004

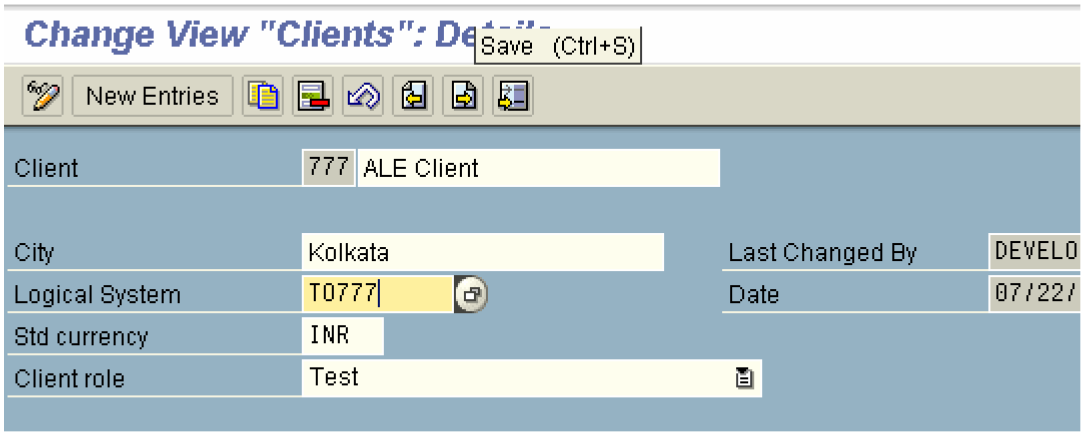


Change View "Clients": Details

Client	555 Development
City	KOLKATA
Logical System	FROM555
Std currency	FROM555
Client role	Customizing

Enter the name of the logical system which you want to assign. Press : **Save**.

Similarly, enter the name of the logical system for another system(in this case, it is client 777).



Change View "Clients": Details

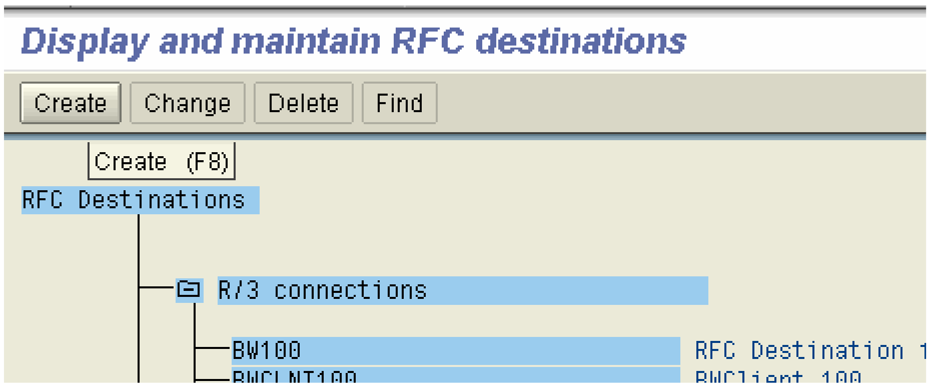
Client	777 ALE Client
City	Kolkata
Logical System	TO777
Std currency	INR
Client role	Test
Last Changed By	DEVELO
Date	07/22/

2.2.3. Create RFC Destination

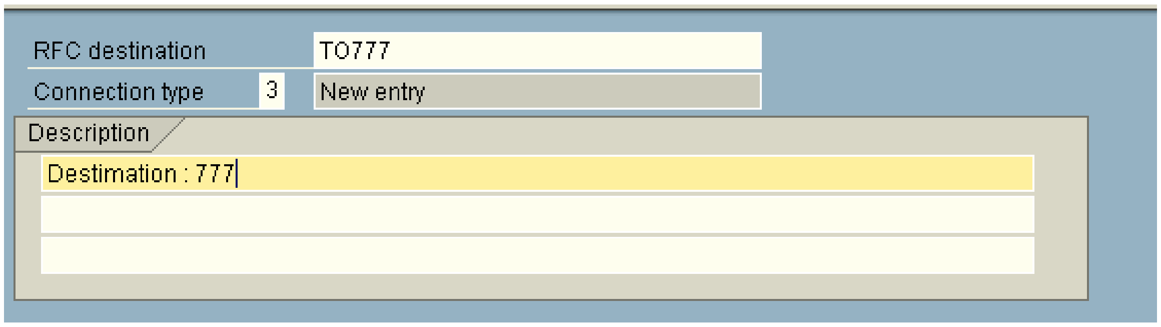
In	Shatadru , 555
Destination	TO777
Tcode	SM59
In	Shatadru , 777
Destination	FROM555
Tcode	SM59

Process

Go to transaction : **SM59** in sender system(Shatadru,555). Press the pushbutton : **Create** from the application toolbar.



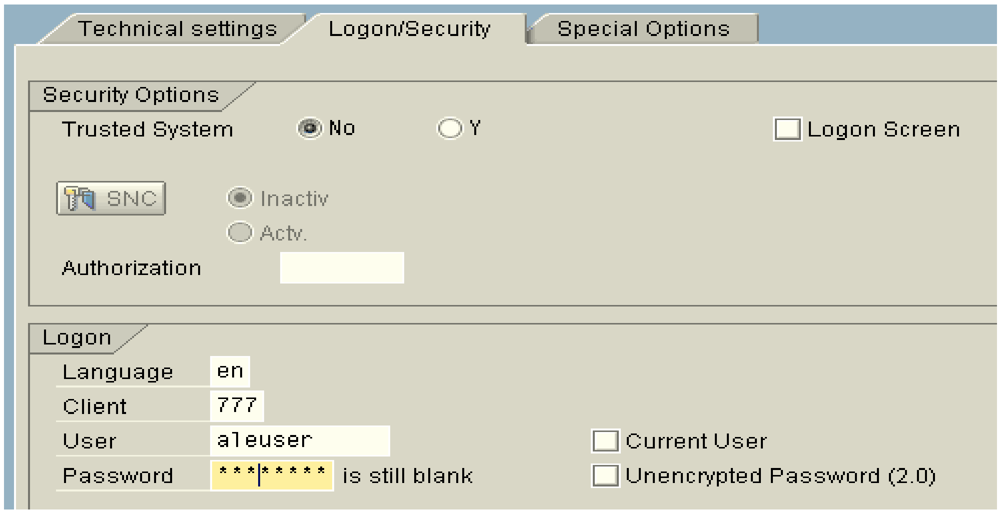
Enter the name of the logical destination., same as that of receiver logical system. Select connection type : 3 for R/3 connection.



RFC destination		TO777
Connection type	3	New entry

Description

Destimation : 777



Security Options

Trusted System [✓ No] Y ☐ Logon Screen

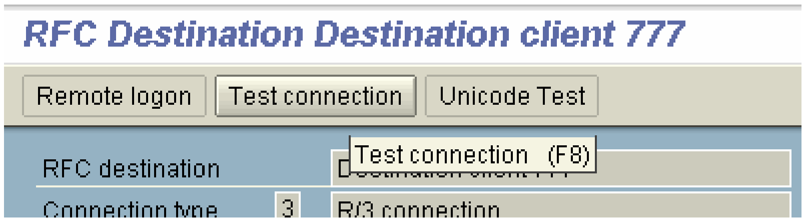
SNC [✓ Inactiv] Actv.

Authorization

Logon

Language	en	
Client	777	
User	aleuser	☐ Current User
Password	******** is still blank	☐ Unencrypted Password (2.0)

Go to the tab page : **Logon/Security**. Enter the logon details.
Press Save.

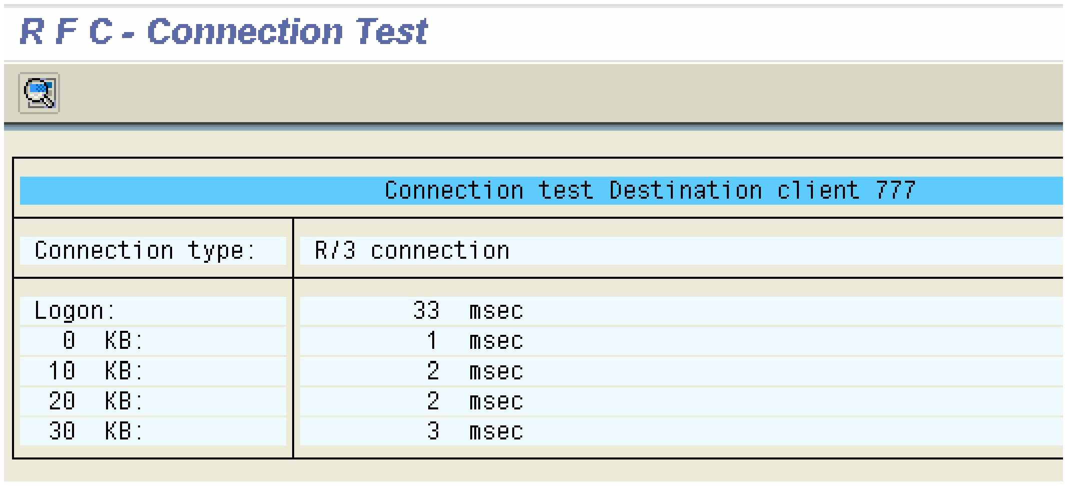


RFC Destination Destination client 777

RFC destination

Connection type 3 R/3 connection

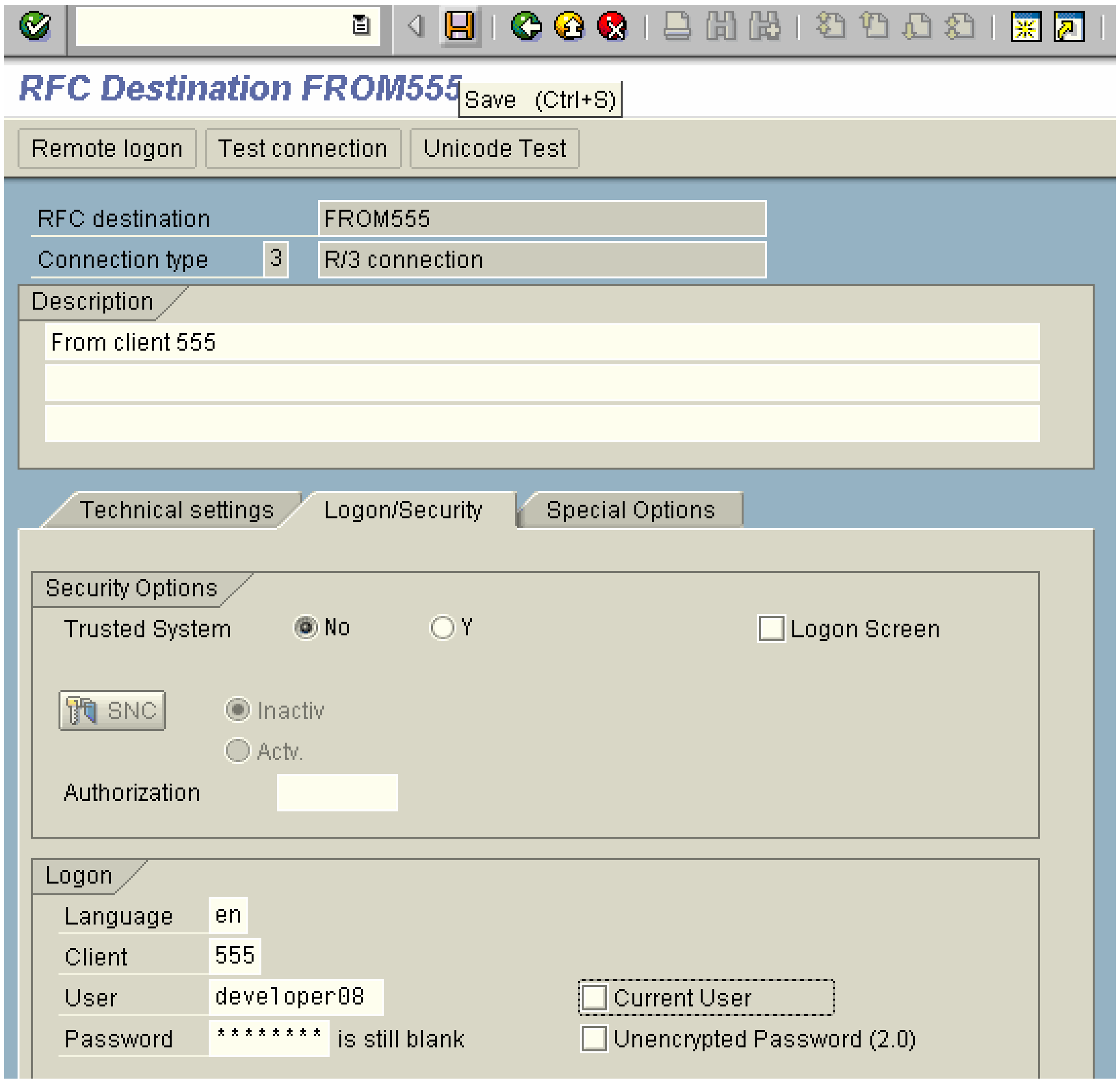
Test the connection by pressing the pushbutton : **Test connection** from the application toolbar.



R F C - Connection Test

```
                    Connection test Destination client 777

Connection type:     R/3 connection

Logon:                       33  msec
  0  KB:                      1  msec
 10  KB:                      2  msec
 20  KB:                      2  msec
 30  KB:                      3  msec
```

Similarly, create RFC destination in client 777 for client 555



RFC Destination FROM555

RFC destination	FROM555
Connection type	3 R/3 connection

Description

From client 555

Security Options

Trusted System [✓ No] Y ☐ Logon Screen

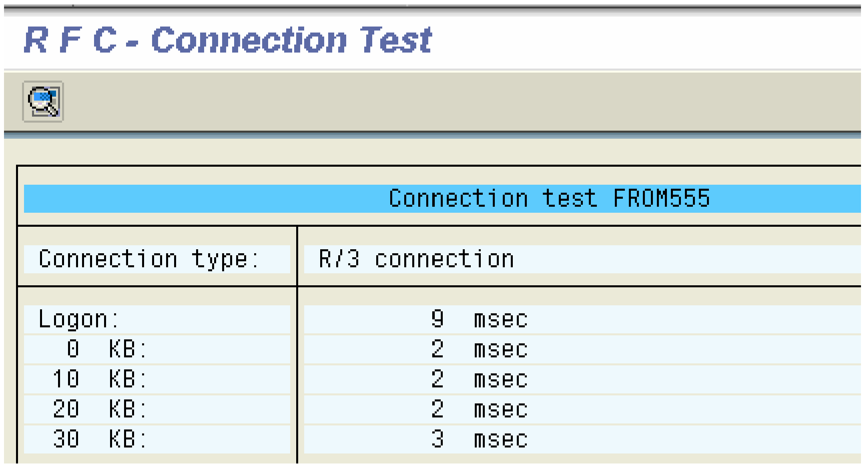
[✓ Inactiv] Actv.

Authorization

Logon

Language	en	
Client	555	
User	developer08	☐ Current User
Password	******** is still blank	☐ Unencrypted Password (2.0)

Save and test.



R F C - Connection Test

```
                 Connection test FROM555

 Connection type:      R/3 connection

 Logon:                        9  msec
   0  KB:                      2  msec
  10  KB:                      2  msec
  20  KB:                      2  msec
  30  KB:                      3  msec
```

2.2.4. Prepare Customer Distribution Model

In	Shatadru , 555
Model name	555TO777
Tcode	BD64

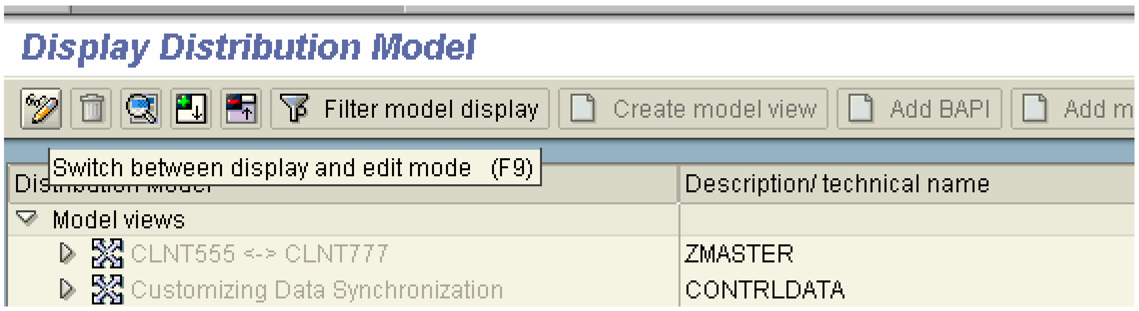
In the Customer Distribution model, you first define a technical system.

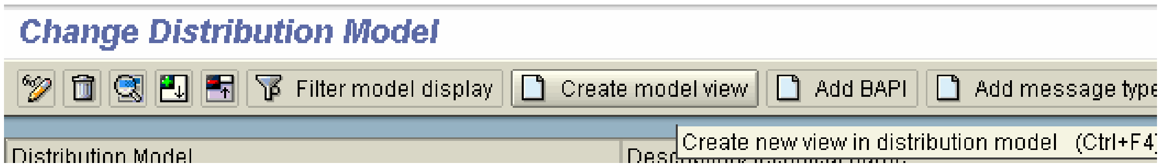
Then for that CDM, you add messages those are likely to be shared between systems.

For each message transferred, you specify the sender and receiver of the message.

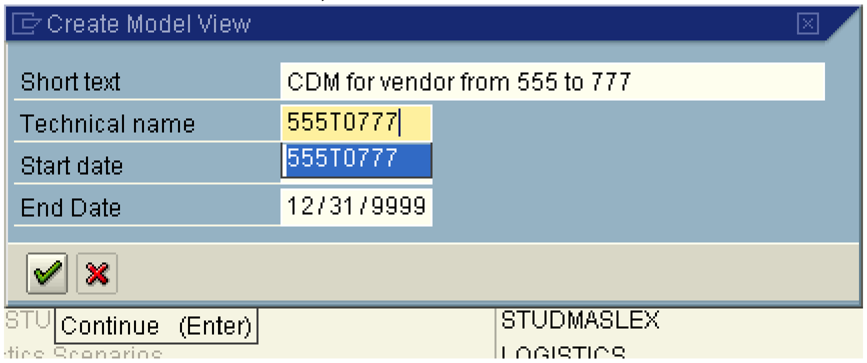
Process

Go to Change mode. Then press the button : **Create Model View** to create a customer distribution model.


Display Distribution Model
Filter model display
Create model view
Add BAPI
Add m
Switch between display and edit mode (F9)
Description/ technical name
Model views
CLNT555 <-> CLNT777
ZMASTER
Customizing Data Synchronization
CONTRLDATA

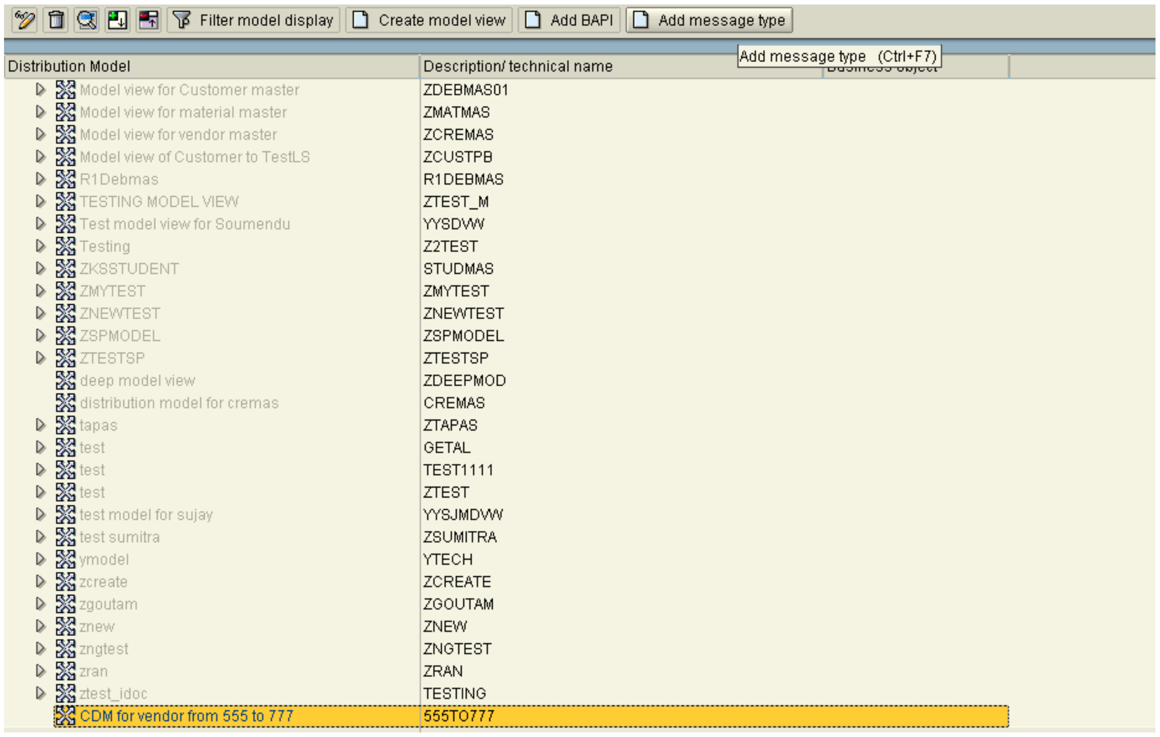


Enter the technical name, short text for the new CDM.



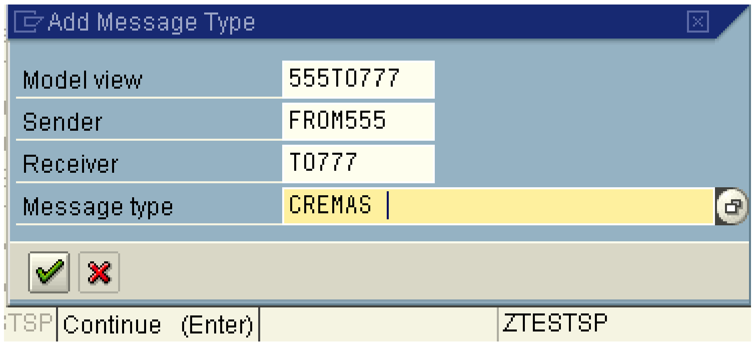
Short text	CDM for vendor from 555 to 777
Technical name	555TO777
Start date	555TO777
End Date	12/31/9999

Then, select the customer distribution model and press the button : **Add message type** from the application toolbar.



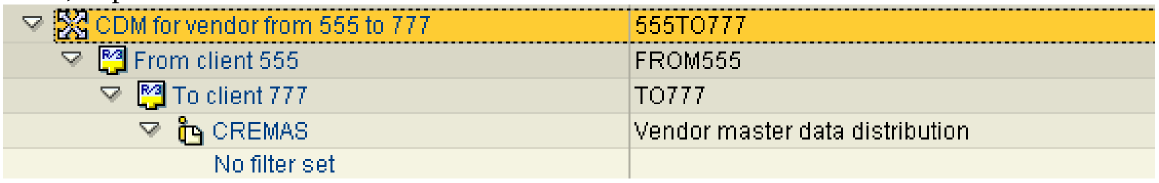
Distribution Model	Description/ technical name
Model view for Customer master	ZDEBMAS01
Model view for material master	ZMATMAS
Model view for vendor master	ZCREMAS
Model view of Customer to TestLS	ZCUSTPB
R1Debmas	R1DEBMAS
TESTING MODEL VIEW	ZTEST_M
Test model view for Soumendu	YYSDVW
Testing	Z2TEST
ZKSSTUDENT	STUDMAS
ZMYTEST	ZMYTEST
ZNEWTEST	ZNEWTEST
ZSPMODEL	ZSPMODEL
ZTESTSP	ZTESTSP
deep model view	ZDEEPMOD
distribution model for cremas	CREMAS
tapas	ZTAPAS
test	GETAL
test	TEST1111
test	ZTEST
test model for sujay	YYSJMDVW
test sumitra	ZSUMITRA
ymodel	YTECH
zcreate	ZCREATE
zgoutam	ZGOUTAM
znew	ZNEW
zngtest	ZNGTEST
zran	ZRAN
ztest_idoc	TESTING
CDM for vendor from 555 to 777	555TO777

Add message type, sender and the receiver.



Model view	555TO777
Sender	FROM555
Receiver	TO777
Message type	CREMAS

Now, explore the CDM to view details.



CDM for vendor from 555 to 777	555TO777
From client 555	FROM555
To client 777	TO777
CREMAS	Vendor master data distribution
No filter set	

Save.

2.2.5. Generate Partner Profile

In	Shatadru , 555
Model name	555TO777
Tcode	BD82

Partner profile is built for both the systems between which messages are to be transferred. So, for two systems communicating, two partners are to be configured in R/3 . The transaction for setting up partner profile manually is WE20. But, as per the current setup, if you generate partner profile using transaction BD82 , these setups are automatically done by SAP R/3.

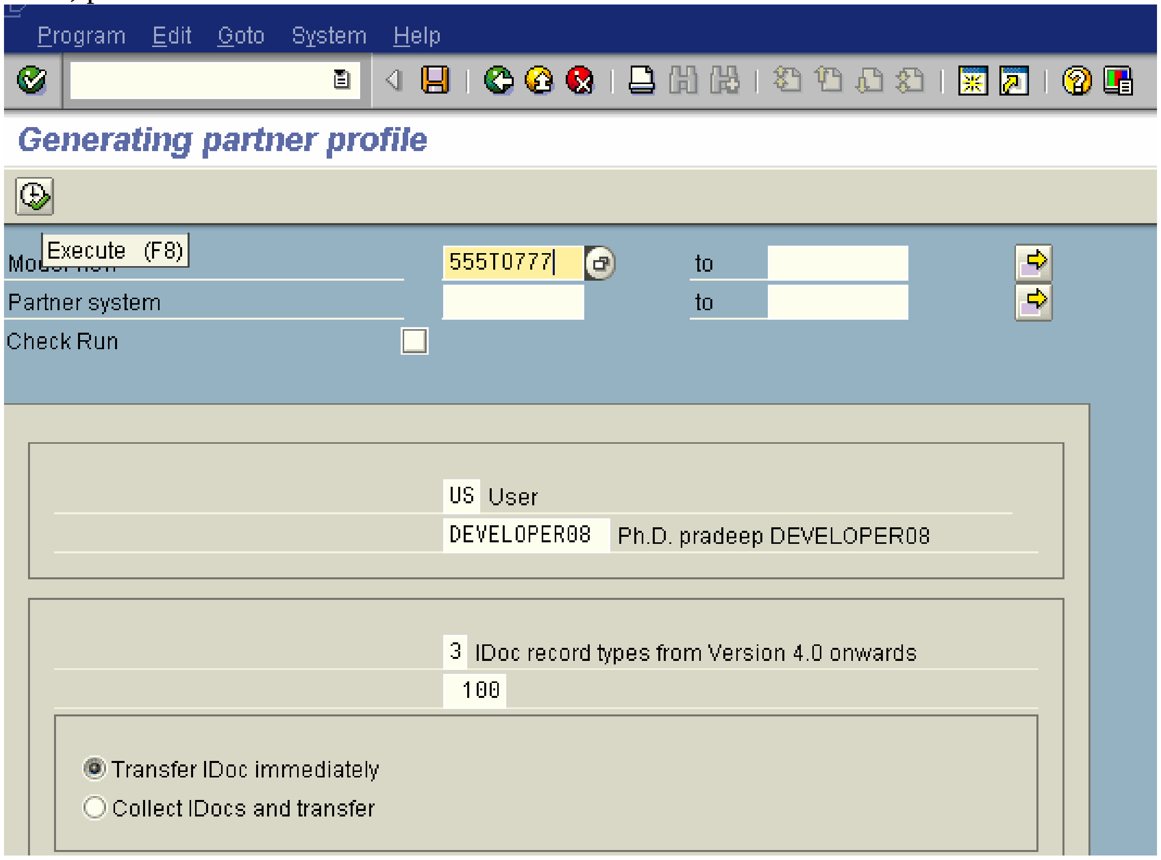
In transaction BD82, you enter the name of the technical system in the selection screen and execute the program. The R/3 system automatically:-

1. Creates partners for the two logical systems.
2. Creates inbound and outbound parameters for different messages in the partner profile for the receiver system.
3. Creates a t-RFC port automatically.

Process

Enter the name of the technical system(that u entered in customer distribution model). Choose to process the IDoc immediately/collect IDoc and transfer by checking the relevant radio buttons in the selection-screen.

Then, press Execute.



Generating partner profile

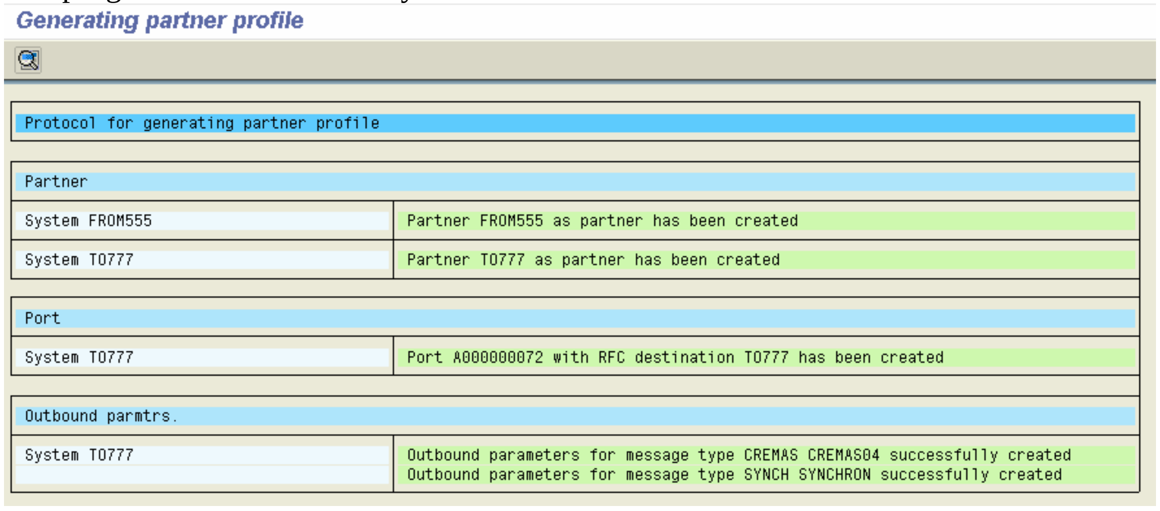
Model view	555T0777	to	
Partner system		to	
Check Run	☐		

US User
DEVELOPER08 Ph.D. pradeep DEVELOPER08

3 IDoc record types from Version 4.0 onwards
100

[✓ Transfer IDoc immediately]
Collect IDocs and transfer

The program does the necessary and furnishes the information.



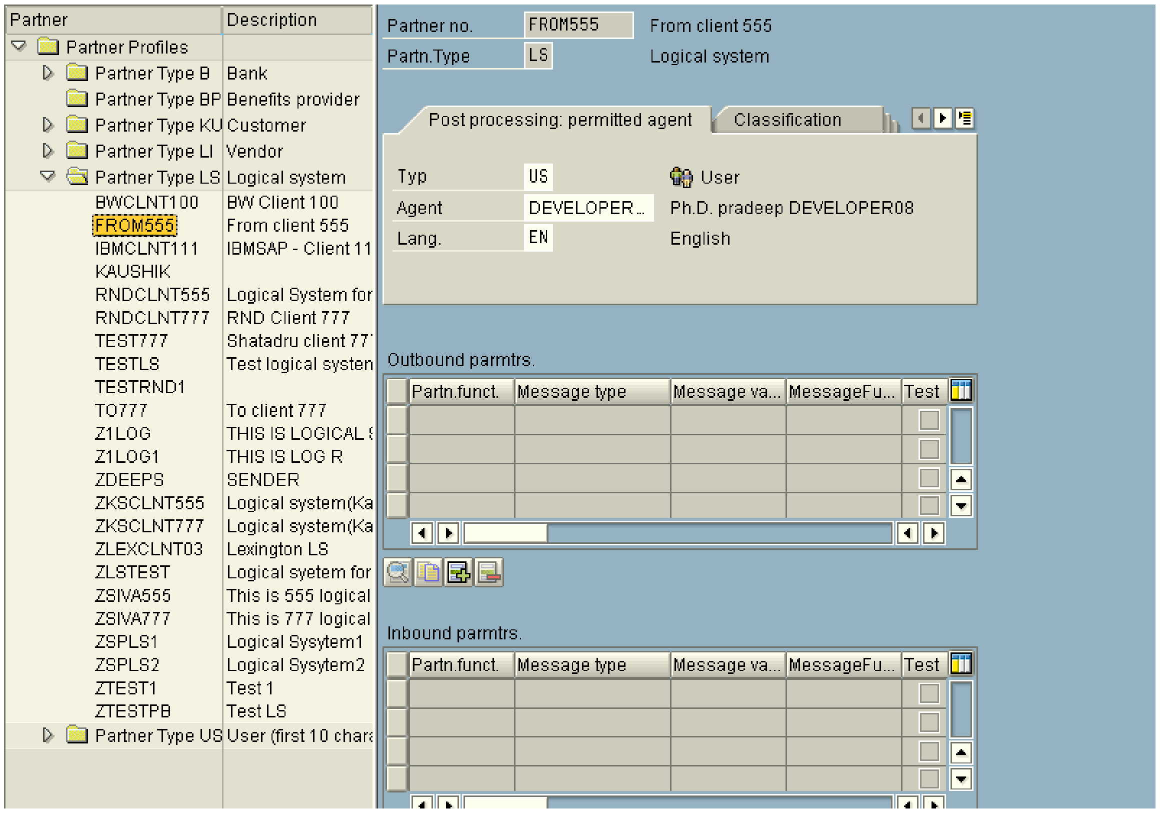
Generating partner profile

```
Protocol for generating partner profile

Partner
System FROM555     Partner FROM555 as partner has been created
System T0777       Partner T0777 as partner has been created

Port
System T0777       Port A000000072 with RFC destination T0777 has been created

Outbound parmtrs.
System T0777       Outbound parameters for message type CREMAS CREMAS04 successfully created
                   Outbound parameters for message type SYNCH SYNCHRON successfully created
```

2.2.6. View Partner Profile created in the sender system

Partner profile is usually created using tcode : WE20. This is the place where you have to create two partners – one for the receiver and one for the sender system.

In the partner profile of the sender system(in this case,Shatadru 555), the partner profile for the sender system contains only the partner definition.



Partner	Description
Partner Profiles	
Partner Type B	Bank
Partner Type BP	Benefits provider
Partner Type KU	Customer
Partner Type LI	Vendor
Partner Type LS	Logical system
BWCLNT100	BW Client 100
FROM555	From client 555
IBMCLNT111	IBMSAP - Client 11
KAUSHIK	
RNDCLNT555	Logical System for
RNDCLNT777	RND Client 777
TEST777	Shatadru client 77
TESTLS	Test logical systen
TESTRND1	
TO777	To client 777
Z1LOG	THIS IS LOGICAL S
Z1LOG1	THIS IS LOG R
ZDEEPS	SENDER
ZKSCLNT555	Logical system(Ka
ZKSCLNT777	Logical system(Ka
ZLEXCLNT03	Lexington LS
ZLSTEST	Logical syetem for
ZSIVA555	This is 555 logical
ZSIVA777	This is 777 logical
ZSPLS1	Logical Sysytem1
ZSPLS2	Logical Sysytem2
ZTEST1	Test 1
ZTESTPB	Test LS
Partner Type US	User (first 10 chara

Partn.no.	FROM555	From client 555
Partn.Type	LS	Logical system

Post processing: permitted agent | Classification

Typ	US	User
Agent	DEVELOPER...	Ph.D. pradeep DEVELOPER08
Lang.	EN	English

Outbound parmtrs.

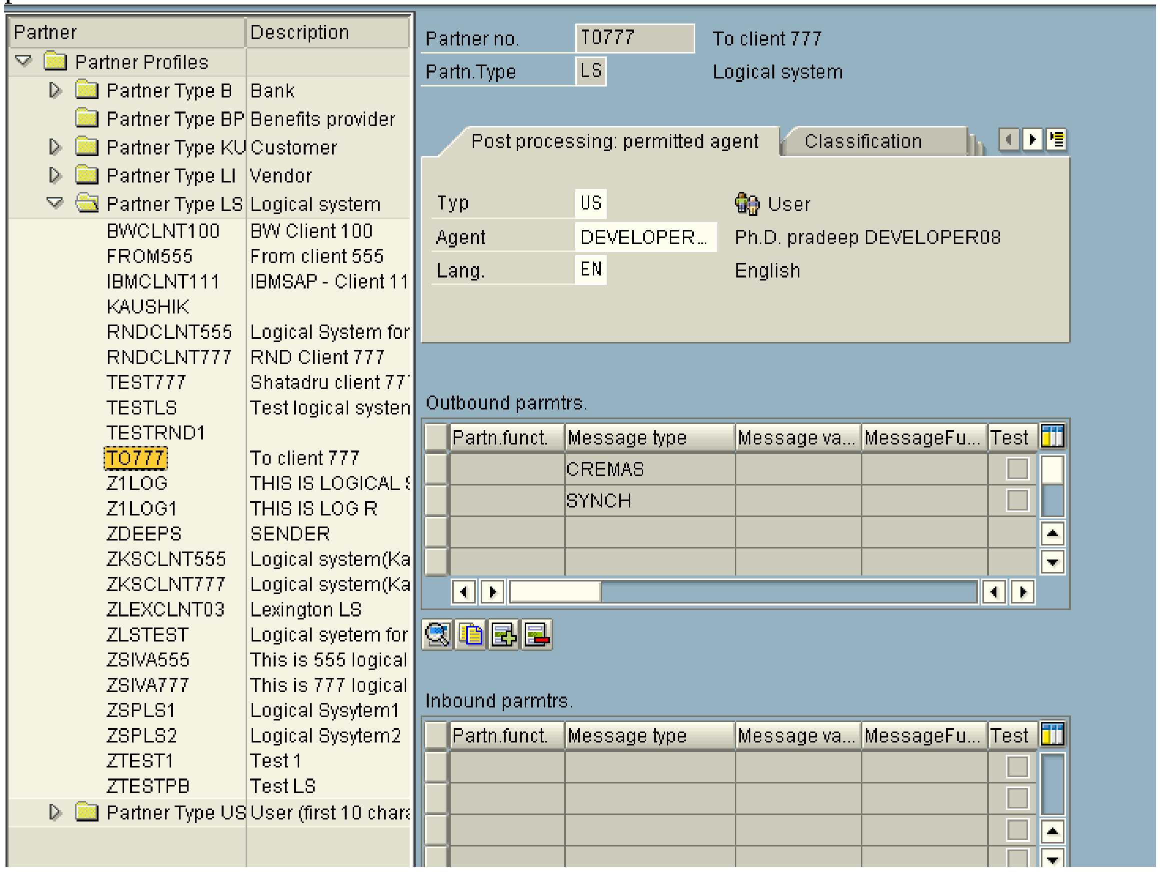
Partn.funct.	Message type	Message va...	MessageFu...	Test

Inbound parmtrs.

Partn.funct.	Message type	Message va...	MessageFu...	Test

In the sender system, the partner profile for the receiver system(Shatadru 777) is maintained as follows:-

The message to be transferred to the receiver system is maintained as outbound parameter.



Partner	Description
Partner Profiles	
Partner Type B	Bank
Partner Type BP	Benefits provider
Partner Type KU	Customer
Partner Type LI	Vendor
Partner Type LS	Logical system
BWCLNT100	BW Client 100
FROM555	From client 555
IBMCLNT111	IBMSAP - Client 11
KAUSHIK	
RNDCLNT555	Logical System for
RNDCLNT777	RND Client 777
TEST777	Shatadru client 77
TESTLS	Test logical systen
TESTRND1	
TO777	To client 777
Z1LOG	THIS IS LOGICAL
Z1LOG1	THIS IS LOG R
ZDEEPS	SENDER
ZKSCLNT555	Logical system(Ka
ZKSCLNT777	Logical system(Ka
ZLEXCLNT03	Lexington LS
ZLSTEST	Logical syetem for
ZSIVA555	This is 555 logical
ZSIVA777	This is 777 logical
ZSPLS1	Logical Sysytem1
ZSPLS2	Logical Sysytem2
ZTEST1	Test 1
ZTESTPB	Test LS
Partner Type US	User (first 10 chara

Partner no.	TO777	To client 777
Partn.Type	LS	Logical system

Post processing: permitted agent | Classification

Typ	US	User
Agent	DEVELOPER...	Ph.D. pradeep DEVELOPER08
Lang.	EN	English

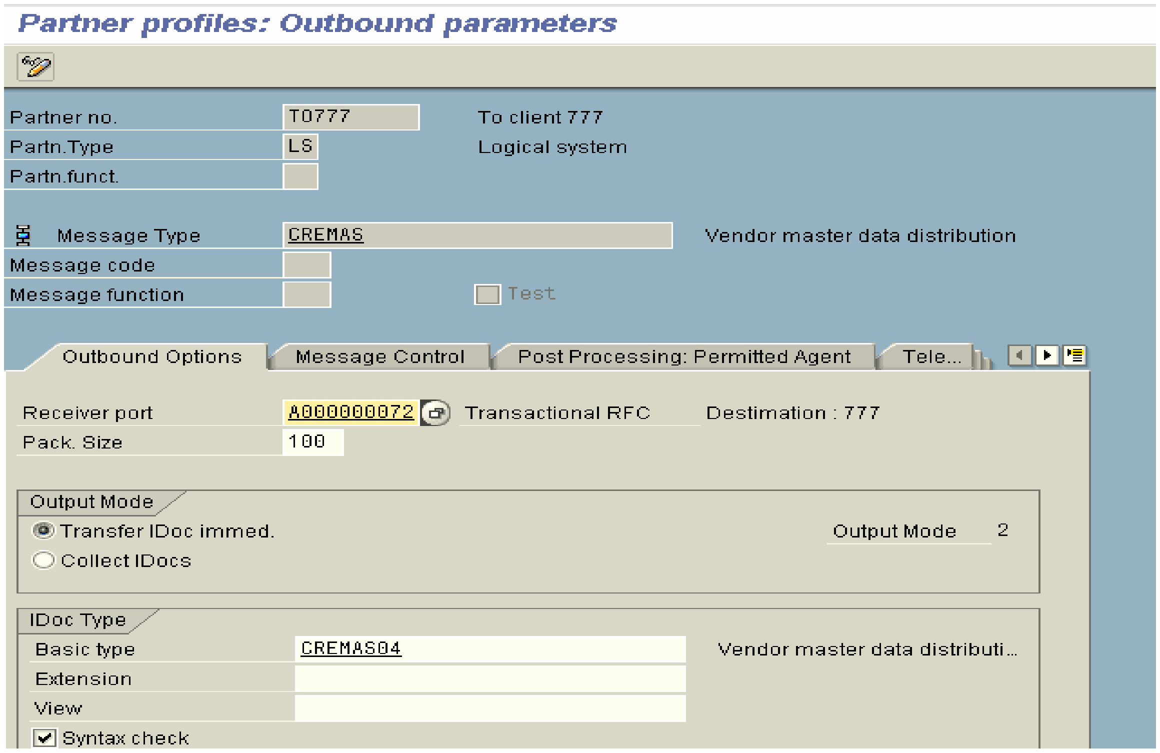
Outbound parmtrs.

Partn.funct.	Message type	Message va...	MessageFu...	Test
	CREMAS			
	SYNCH			

Inbound parmtrs.

Partn.funct.	Message type	Message va...	MessageFu...	Test

Then, for that message type, the receiver port, the basic Idoc type is mentioned. Also, the packet size of the Idocs(in case they are collected and transferred) are specified too.


Partner profiles: Outbound parameters

Partner no.	T0777	To client 777	
Partn.Type	LS	Logical system	
Partn.funct.			
Message Type	CREMAS	Vendor master data distribution	
Message code			
Message function		Test	

Outbound Options | Message Control | Post Processing: Permitted Agent | Tele...

Receiver port	A000000072	Transactional RFC	Destimation : 777
Pack. Size	100		

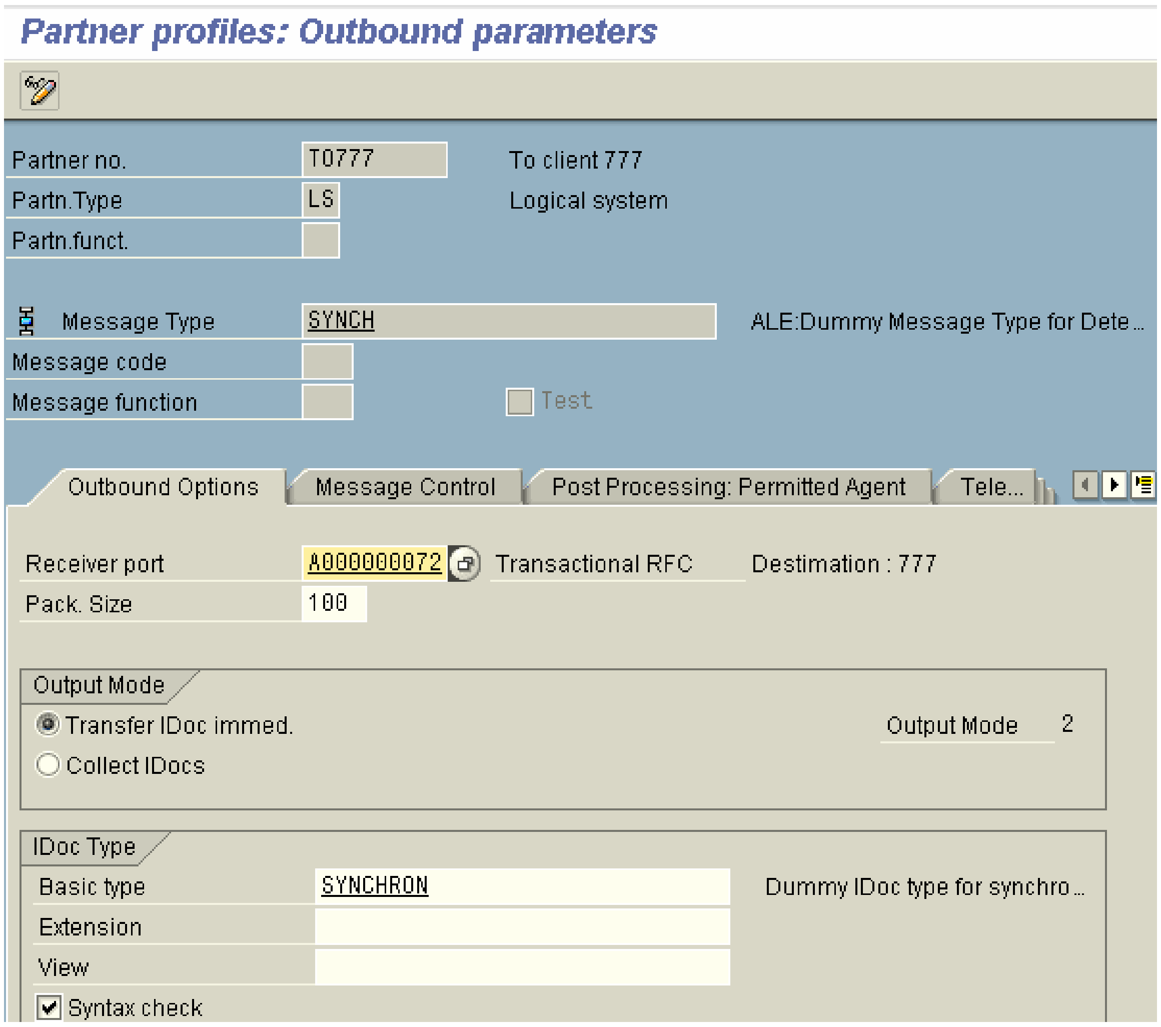
Output Mode
(● Transfer IDoc immed.) ○ Collect IDocs — Output Mode 2

IDoc Type

Basic type	CREMAS04	Vendor master data distributi...
Extension		
View		

☑ Syntax check

But, if you generate partner profile using tcode : BD82, these jobs are done by SAP itself.


Partner profiles: Outbound parameters

Partner no.	T0777	To client 777
Partn.Type	LS	Logical system
Partn.funct.		
Message Type	SYNCH	ALE:Dummy Message Type for Dete...
Message code		
Message function		Test [ ]

Outbound Options | Message Control | Post Processing: Permitted Agent | Tele...

Receiver port	A000000072	Transactional RFC	Destination : 777
Pack. Size	100		

Output Mode
[✓ Transfer IDoc immed.] Output Mode 2
Collect IDocs

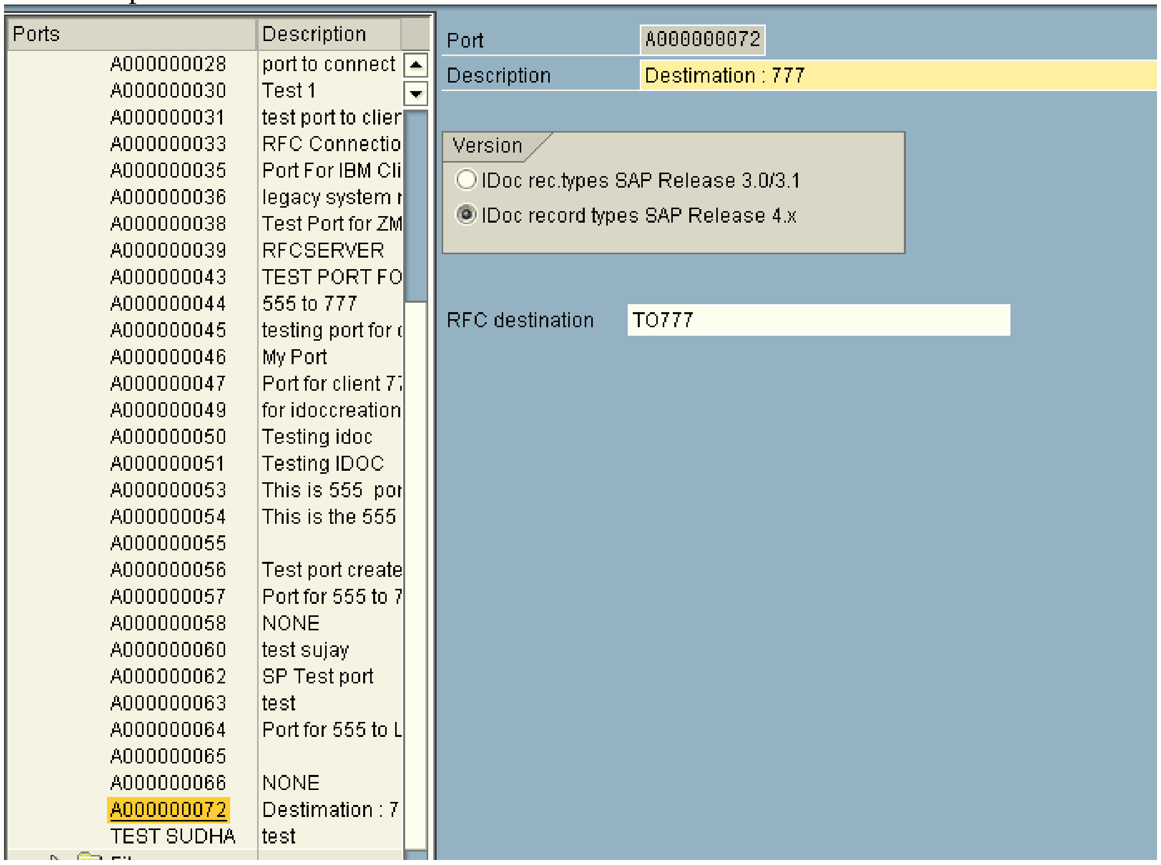
IDoc Type

Basic type	SYNCHRON	Dummy IDoc type for synchro...
Extension		
View		

[✓] Syntax check

2.2.7. View port information in sender

View the port information from transaction WE21.



Ports	Description
A000000028	port to connect
A000000030	Test 1
A000000031	test port to clier
A000000033	RFC Connectio
A000000035	Port For IBM Cli
A000000036	legacy system r
A000000038	Test Port for ZM
A000000039	RFCSERVER
A000000043	TEST PORT FO
A000000044	555 to 777
A000000045	testing port for (
A000000046	My Port
A000000047	Port for client 77
A000000049	for idoccreation
A000000050	Testing idoc
A000000051	Testing IDOC
A000000053	This is 555 por
A000000054	This is the 555
A000000055	
A000000056	Test port create
A000000057	Port for 555 to 7
A000000058	NONE
A000000060	test sujay
A000000062	SP Test port
A000000063	test
A000000064	Port for 555 to L
A000000065	
A000000066	NONE
A000000072	Destimation : 7
TEST SUDHA	test

Port A000000072

Description Destimation : 777

Version

( ) IDoc rec.types SAP Release 3.0/3.1

(●) IDoc record types SAP Release 4.x

RFC destination TO777

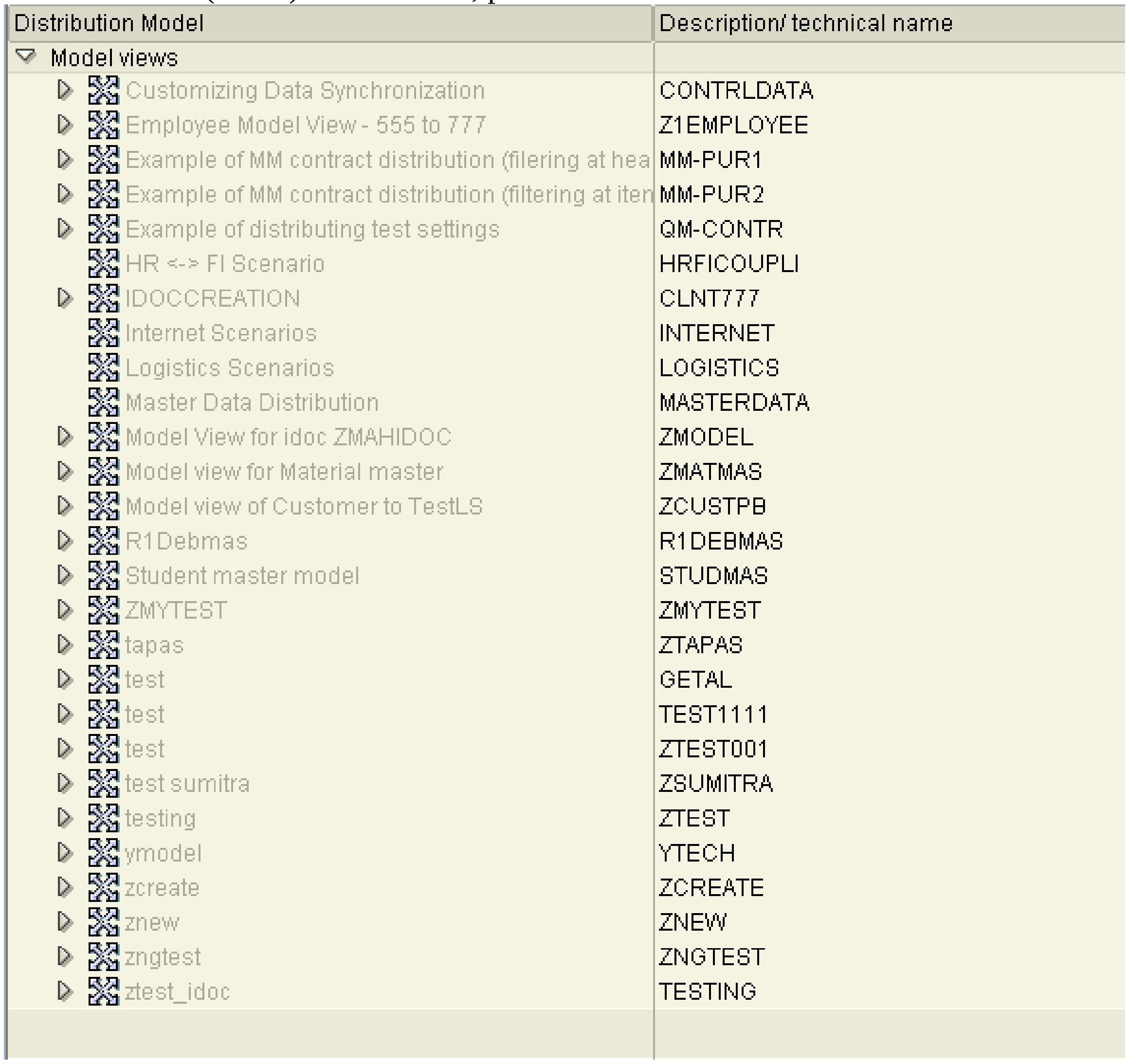
2.2.8. Distribute Customer Distribution Model

In	Shatadru 555
Tcode	BD64

Now, we are ready with customer distribution model, partner profiles, ports in the sender system(Shatadru,555) . But, the customer distribution model is not maintaied in reciver system. We need to maintain CDM in receiver system, too with the same customer distribution model, same message type, sender and receiver.

For that, one can distribute the customer distribution model from client 555 . Note, if you do that, the CDM will be replicated in client 777.

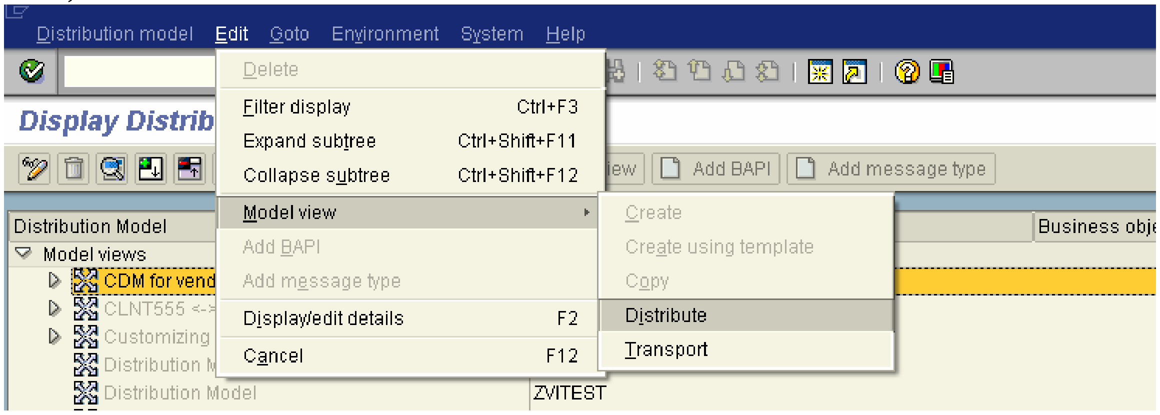
Picture of CDM(BD64) in client 777 , prior to distribution:-



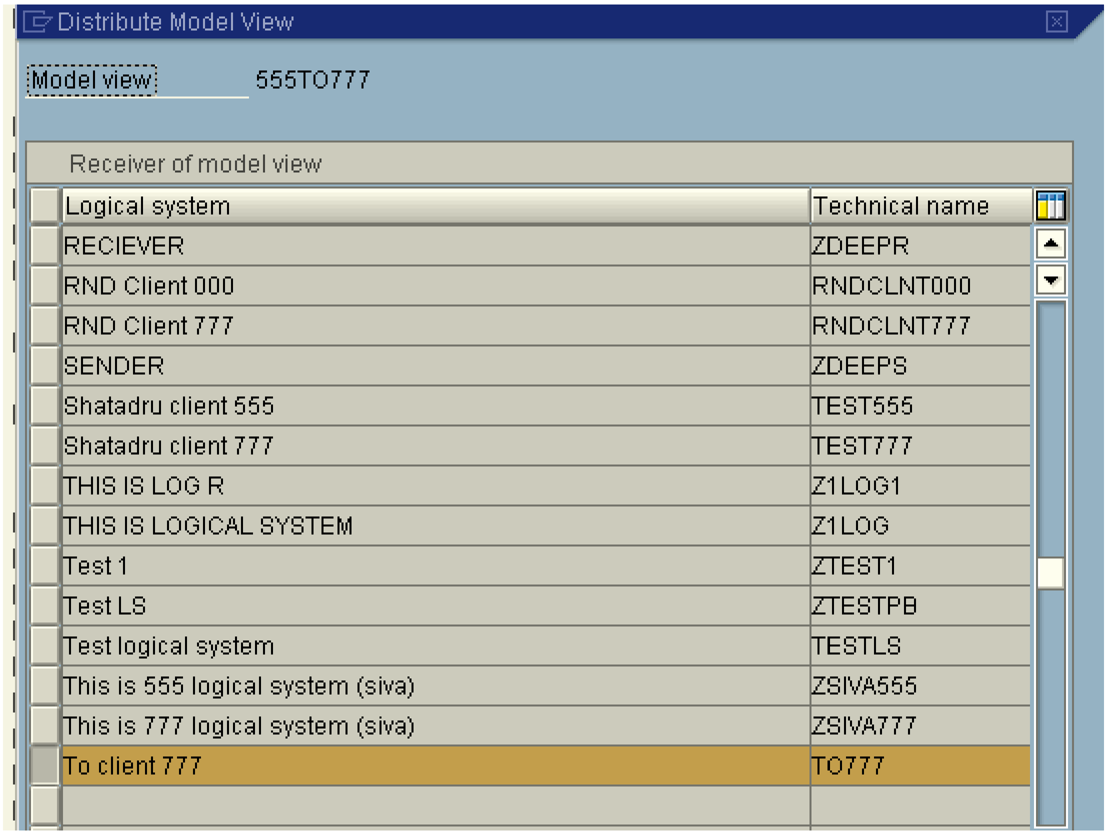
Distribution Model	Description/ technical name
Model views	
Customizing Data Synchronization	CONTRLDATA
Employee Model View - 555 to 777	Z1EMPLOYEE
Example of MM contract distribution (filering at hea	MM-PUR1
Example of MM contract distribution (filtering at iten	MM-PUR2
Example of distributing test settings	QM-CONTR
HR <-> FI Scenario	HRFICOUPLI
IDOCCREATION	CLNT777
Internet Scenarios	INTERNET
Logistics Scenarios	LOGISTICS
Master Data Distribution	MASTERDATA
Model View for idoc ZMAHIDOC	ZMODEL
Model view for Material master	ZMATMAS
Model view of Customer to TestLS	ZCUSTPB
R1Debmas	R1DEBMAS
Student master model	STUDMAS
ZMYTEST	ZMYTEST
tapas	ZTAPAS
test	GETAL
test	TEST1111
test	ZTEST001
test sumitra	ZSUMITRA
testing	ZTEST
ymodel	YTECH
zcreate	ZCREATE
znew	ZNEW
zngtest	ZNGTEST
ztest_idoc	TESTING

Note that, our CDM is not there.

Now, distribute the CDM in client 555 as shown below:-



(Select the CDM , then from the menu path, choose :- Edit→Model View→Distribute)

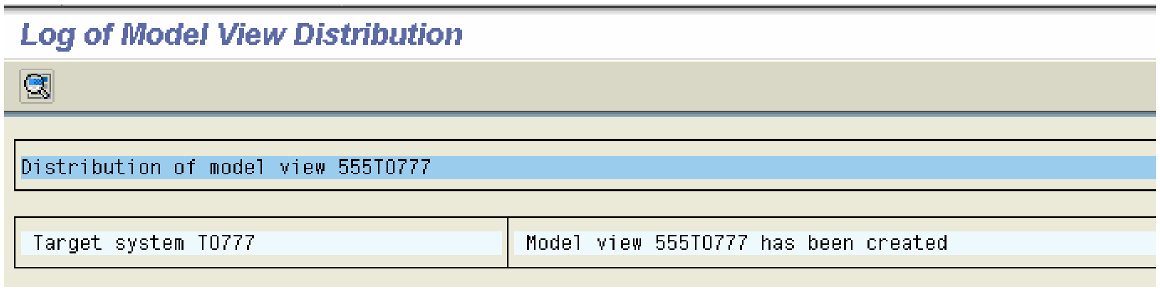

Distribute Model View

Model view 555TO777

Receiver of model view

Logical system	Technical name
RECIEVER	ZDEEPR
RND Client 000	RNDCLNT000
RND Client 777	RNDCLNT777
SENDER	ZDEEPS
Shatadru client 555	TEST555
Shatadru client 777	TEST777
THIS IS LOG R	Z1LOG1
THIS IS LOGICAL SYSTEM	Z1LOG
Test 1	ZTEST1
Test LS	ZTESTPB
Test logical system	TESTLS
This is 555 logical system (siva)	ZSIVA555
This is 777 logical system (siva)	ZSIVA777
To client 777	TO777

Choose the destination from the list and press Enter.

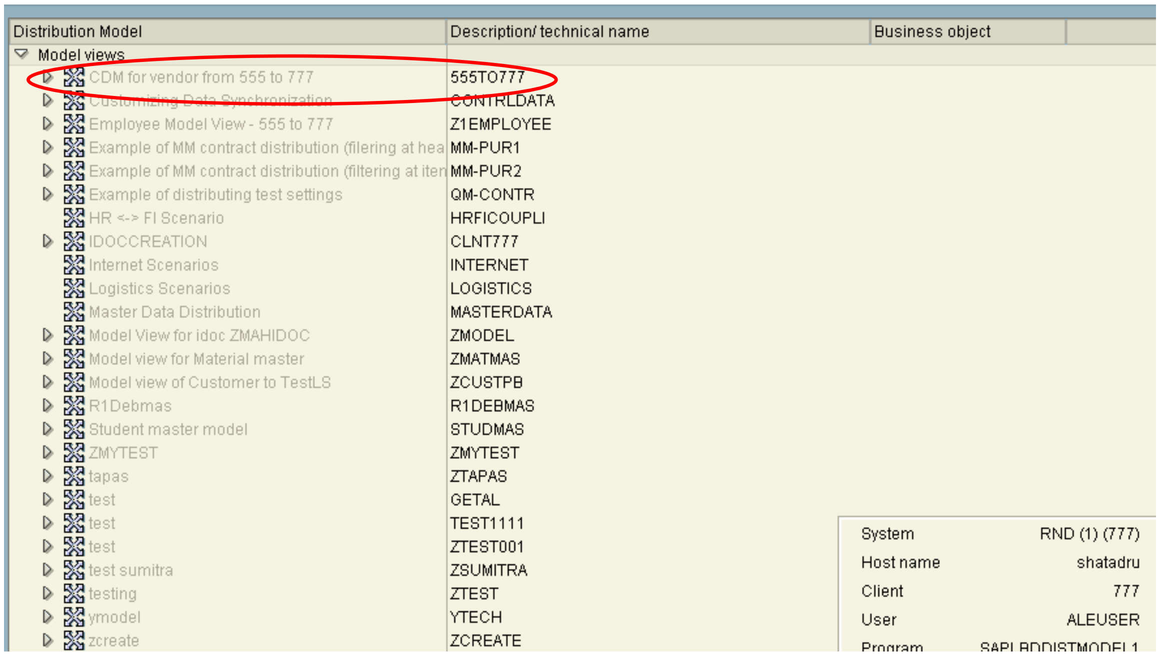


Log of Model View Distribution

```
Distribution of model view 555T0777

Target system T0777          Model view 555T0777 has been created
```

The model view now gets created in client 777.

Look at the CDM list on client 777 now from transaction BD64.



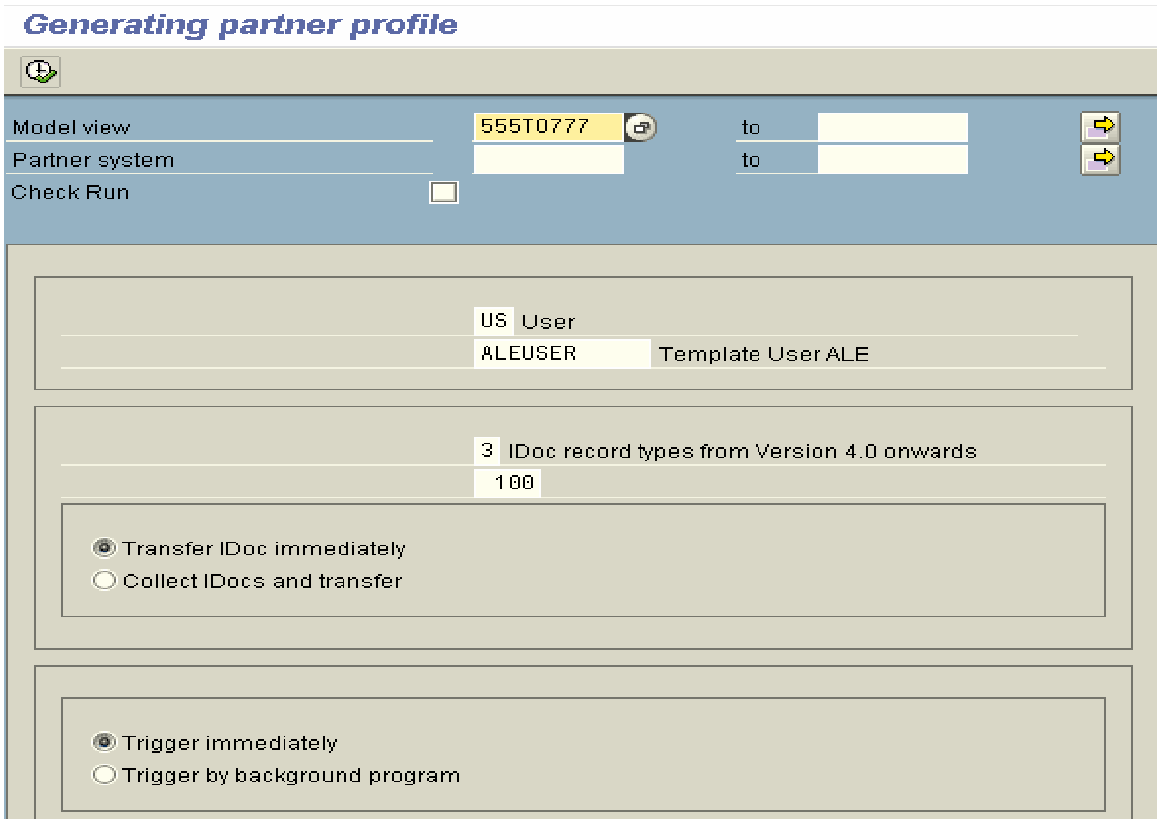
Distribution Model	Description/ technical name	Business object
Model views		
CDM for vendor from 555 to 777	555T0777	
Customizing Data Synchronization	CONTRLDATA	
Employee Model View - 555 to 777	Z1EMPLOYEE	
Example of MM contract distribution (filering at hea	MM-PUR1	
Example of MM contract distribution (filtering at iten	MM-PUR2	
Example of distributing test settings	QM-CONTR	
HR <-> FI Scenario	HRFICOUPLI	
IDOCCREATION	CLNT777	
Internet Scenarios	INTERNET	
Logistics Scenarios	LOGISTICS	
Master Data Distribution	MASTERDATA	
Model View for idoc ZMAHIDOC	ZMODEL	
Model view for Material master	ZMATMAS	
Model view of Customer to TestLS	ZCUSTPB	
R1Debmas	R1DEBMAS	
Student master model	STUDMAS	
ZMYTEST	ZMYTEST	
tapas	ZTAPAS	
test	GETAL	
test	TEST1111	
test	ZTEST001	
test sumitra	ZSUMITRA	
testing	ZTEST	
ymodel	YTECH	
zcreate	ZCREATE	

System	RND (1) (777)
Host name	shatadru
Client	777
User	ALEUSER
Program	SAPLBDDISTMODEL1

It is created now.

2.2.9. Generate Partner Profile in client 777

In	Shatadru 777
Tcode	BD82



Generating partner profile

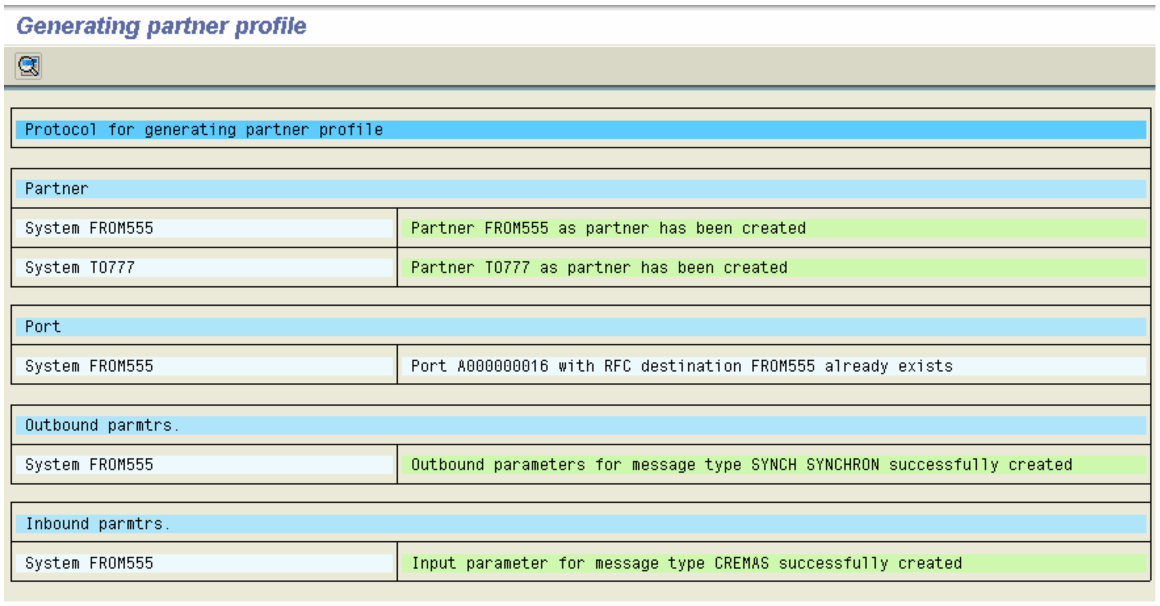
Model view	555T0777	to	
Partner system		to	
Check Run	☐		

US User
ALEUSER Template User ALE

3 IDoc record types from Version 4.0 onwards
100

[✓ Transfer IDoc immediately]
Collect IDocs and transfer

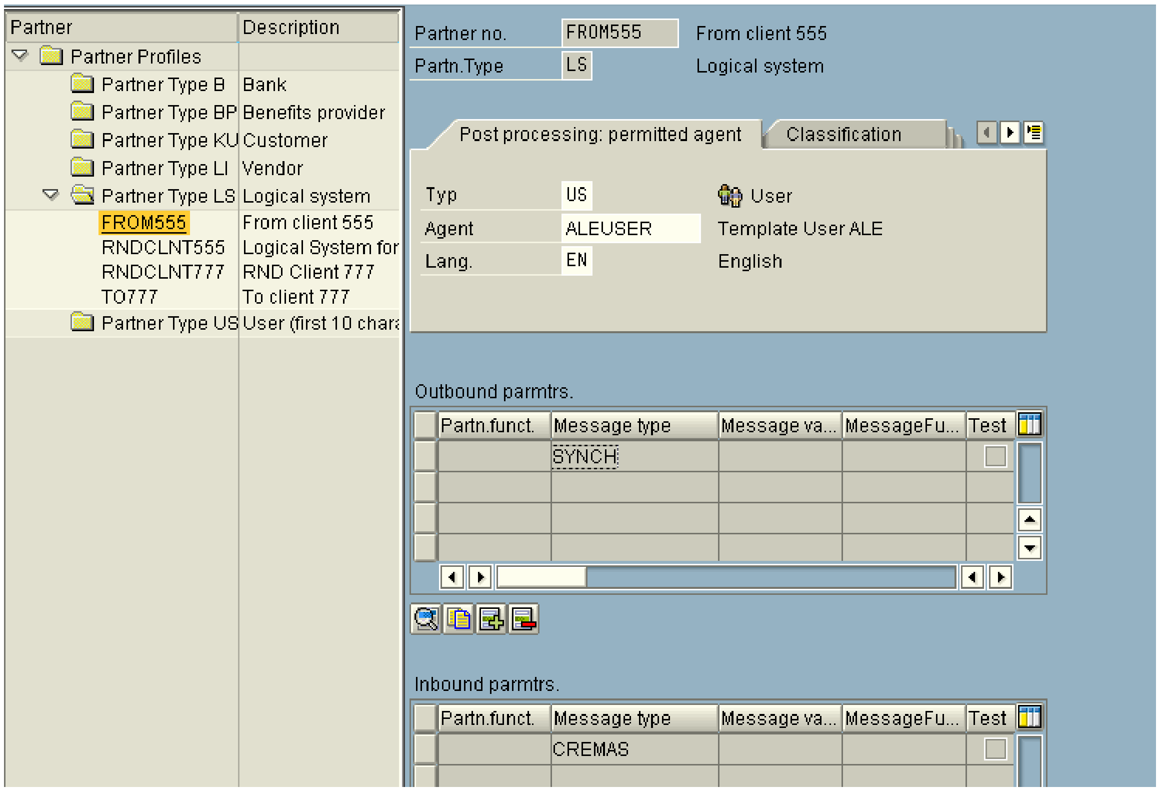
[✓ Trigger immediately]
Trigger by background program



Generating partner profile

```
Protocol for generating partner profile

Partner
System FROM555      Partner FROM555 as partner has been created
System T0777        Partner T0777 as partner has been created

Port
System FROM555      Port A000000016 with RFC destination FROM555 already exists

Outbound parmtrs.
System FROM555      Outbound parameters for message type SYNCH SYNCHRON successfully created

Inbound parmtrs.
System FROM555      Input parameter for message type CREMAS successfully created
```

Look at the partner profile of FROM555 and TO777 from transaction WE20 in client 777:-



Partner	Description
Partner Profiles	
Partner Type B	Bank
Partner Type BP	Benefits provider
Partner Type KU	Customer
Partner Type LI	Vendor
Partner Type LS	Logical system
FROM555	From client 555
RNDCLNT555	Logical System for
RNDCLNT777	RND Client 777
TO777	To client 777
Partner Type US	User (first 10 chara

Partner no. FROM555 From client 555
Partn.Type LS Logical system

Post processing: permitted agent | Classification

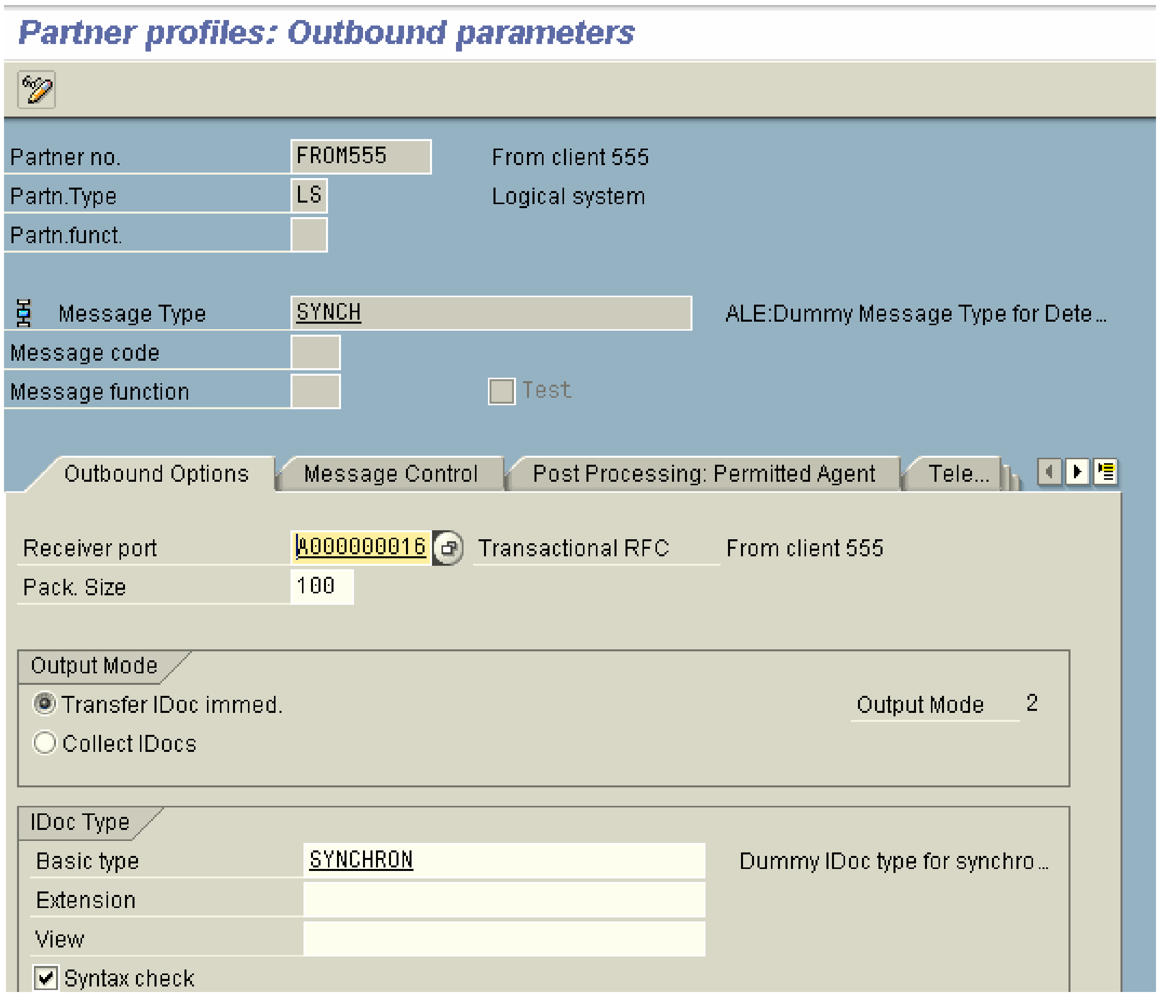
Typ	US	User
Agent	ALEUSER	Template User ALE
Lang.	EN	English

Outbound parmtrs.

Partn.funct.	Message type	Message va...	MessageFu...	Test
	SYNCH			

Inbound parmtrs.

Partn.funct.	Message type	Message va...	MessageFu...	Test
	CREMAS			



Partner profiles: Outbound parameters

Partner no.	FROM555	From client 555
Partn.Type	LS	Logical system
Partn.funct.		
Message Type	SYNCH	ALE:Dummy Message Type for Dete...
Message code		
Message function		Test

Receiver port	A000000016	Transactional RFC	From client 555
Pack. Size	100		

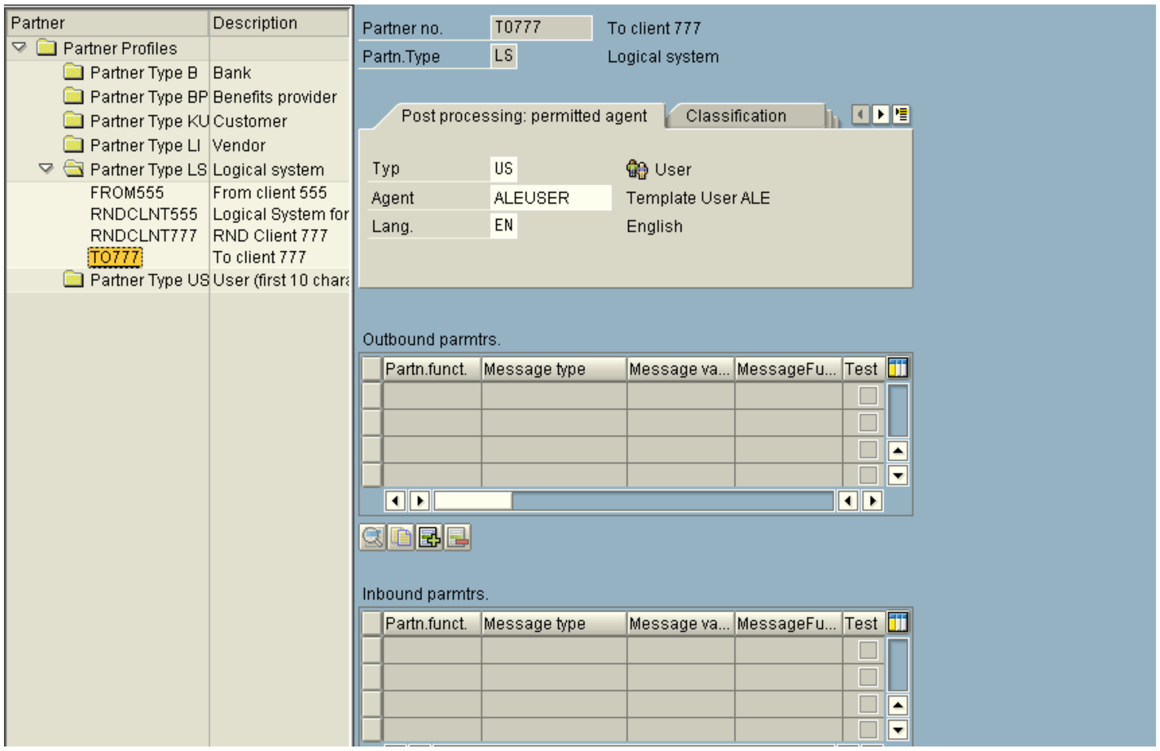
Output Mode
- [✓ Transfer IDoc immed.]
- Collect IDocs

Output Mode 2

IDoc Type

Basic type	SYNCHRON	Dummy IDoc type for synchro...
Extension		
View		

Syntax check ✓



Partner	Description
Partner Profiles	
Partner Type B	Bank
Partner Type BP	Benefits provider
Partner Type KU	Customer
Partner Type LI	Vendor
Partner Type LS	Logical system
FROM555	From client 555
RNDCLNT555	Logical System for
RNDCLNT777	RND Client 777
TO777	To client 777
Partner Type US	User (first 10 chara

Partner no. T0777 To client 777
Partn.Type LS Logical system

Post processing: permitted agent | Classification

Typ	US	User
Agent	ALEUSER	Template User ALE
Lang.	EN	English

Outbound parmtrs.

Partn.funct.	Message type	Message va...	MessageFu...	Test
				☐
				☐
				☐
				☐

Inbound parmtrs.

Partn.funct.	Message type	Message va...	MessageFu...	Test
				☐
				☐
				☐
				☐

2.2.10. Create and Distribute Material Master

All your settings are done. Now, create one material and then use tcode: BD14 to distribute the Vendor master by ALE. Then, follow the status of the IDoc from tcode : WE05.

Go to the receiver system and view the status of the IDoc by WE05. If successful, view the vendor created by XK03.

3. System to System ALE Setup

You have configured client to client ALE setup previously. Now, let us have a system-to-system ALE setup. The scenario used is tabulated below:-

Sender	IBMSAP(9.182.150.5) Client 111
Receiver	Shatadru(9.182.150.33) Client 555
Message	MATMAS (Material master)

3.1. Steps

Steps to be followed:-

Step no	**Process**	**Transaction Code**
1	Create Logical Systems for both sender and Receiver	BD54
2	Assign Logical Systems to Clients	SCC4
3	Create RFC Destination for Receiver in the Sender system and vice versa	SM59
4	Maintain Customer Distribution Model	BD64
5	Generate Partner Profile	BD82
6	Distribute Customer Model to the receiving logical system	BD64
7	Generation of Partner Profile in the receiving logical system	BD82
8	Check consistency of Customer Distribution Model	BDC5
9	Create/change Vendor master records	XK01/XK02
10	Distributing records to the receiving logical system	BD14
12	Checking for the material in the receiving system	MM03

3.1.1. Create Logical Systems for both sender and Receiver

Tcode	BD54
Done in	IBMSAP Client 111 and Shatadru(555)
Process	1. Login to IBMSAP using client 111(sender) as: aleuser. Access transaction BD54. From the Application Toolbar, press the pushbutton: New Entries (F5). Create logical system TO555, IBMCLNT111 and save. 2. Login to Shatadru in client **555**(receiver) as: developer08. Access transaction **BD54**. From the Application Toolbar, press the pushbutton: **New Entries** (F5). Create logical system TO555, IBMCLNT111 and save. 3. Save and come out

3.1.2. Assign Logical Systems to Clients

Tcode	SCC4
Done in	IBMSAP Client 111
Process	1. Access transaction SCC4. Press the Change (Ctrl-F1) pushbutton from the Application Toolbar. It will bring the change mode so that the records can be changed. 2. Select the line for the sender client(111) from the table control and press the **Details** pushbutton (Ctrl+Shift+F2) pushbutton from the Application Toolbar. 3. This will bring the Details entry screen for the client. In the field for Logical System, enter the name of the logical system created for the Sender (ALECLNT111) 4. Save and come out. Do the same for the receiver client also.
Tcode	SCC4
Done in	Shatadru Client 555
	4. Access transaction SCC4. Press the Change (Ctrl-F1) pushbutton from the Application Toolbar. It will bring the change mode so that the records can be changed. 5. Select the line for the sender client(555) from the table control and press the **Details** pushbutton (Ctrl+Shift+F2) pushbutton from the Application Toolbar. 6. This will bring the Details entry screen for the client. In the field for Logical System, enter the name of the logical system created for the Sender(TO555).

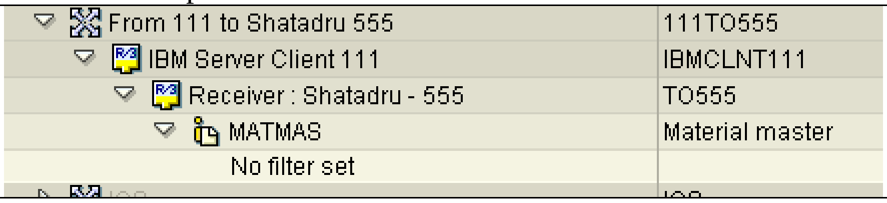
	4. Save and come out. Do the same for the receiver client also.

3.1.3. Create RFC Destination

IBMCLNT111 in Shatadru(555)
TO555 in IBMSAP(111)

Tcode	SM59
Done in	Create RNDCLNT100 in Shatadru 500 and ALESYS500 in PwCSAP 100
Process	See the documentation on Client to client communication

3.1.4. Maintain Customer Distribution Model

Tcode	BD64
Done in	IBMSAP 111
Process	Follow the steps as shown in the documentation : Client to Client communication.
Screen shot	 From 111 to Shatadru 555 — 111TO555; IBM Server Client 111 — IBMCLNT111; Receiver : Shatadru - 555 — TO555; MATMAS — Material master; No filter set

3.1.5. Generate Partner Profile

Tcode	BD82
Done in	IBMSAP 111
Process	Enter the name of the model in the selection-screen and execute. The report generated will tell you about successful creation of the partner profile(creation of partner,port,outbound,inbound parameters etc)

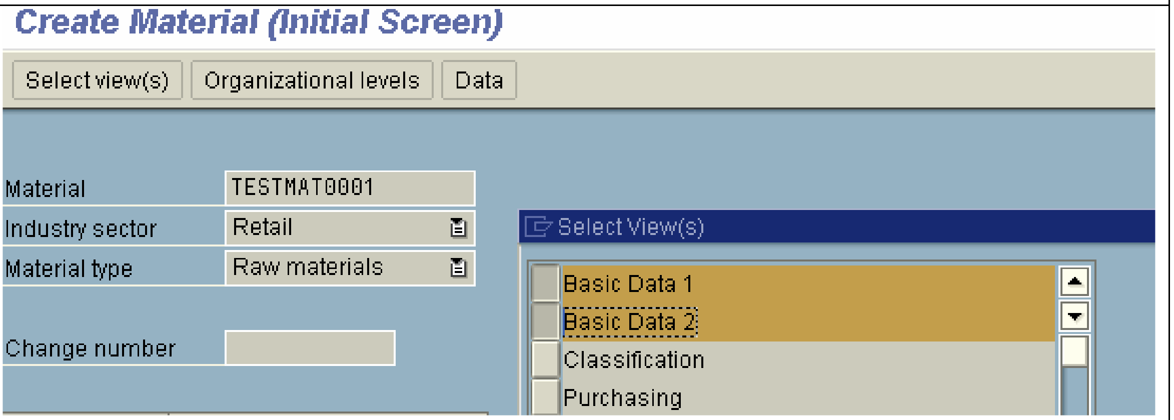
3.1.6. Distribute Customer Distribution Model

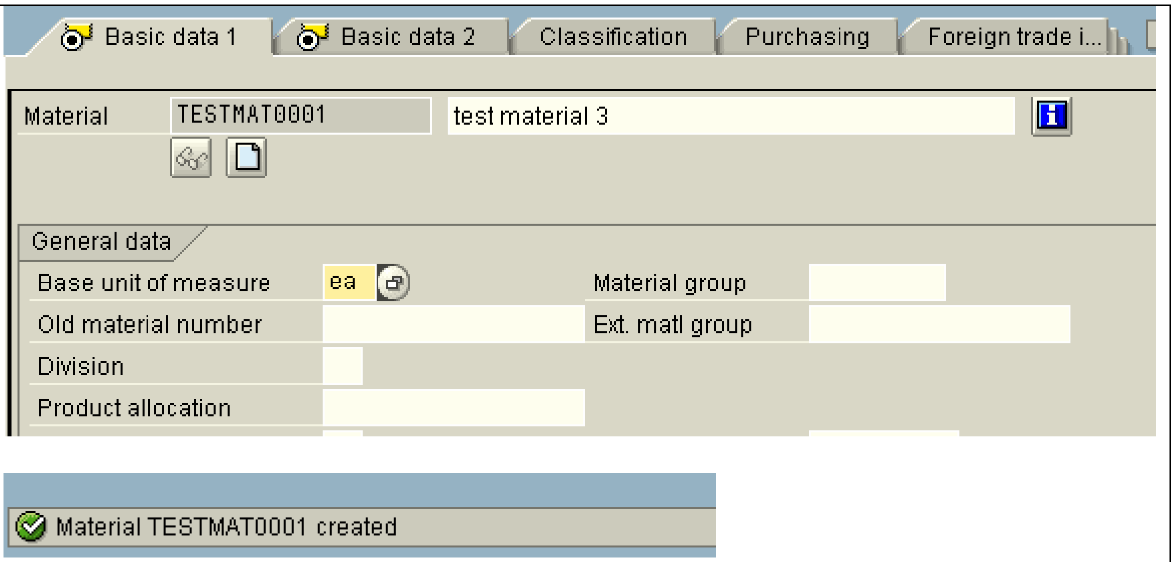
Tcode	BD64
Done in	IBMSAP 111
Process	See documentation of : Client to client communication

3.1.7. Generating Partner Profile in the Receiving Logical Systems

Tcode	BD82
Done in	Shatadru 555
Process	. Enter The name of the customer distribution model and press **Execute(F8)** button. A report will be published, informing you about the generation of the partner profile in the receiving logical system.

3.1.8. Create /Change Material master records

Tcode	MM01/MM02
Done in	IBMSAP 111
Process	 Create Material (Initial Screen) Material: TESTMAT0001 Industry sector: Retail Material type: Raw materials Change number: Select View(s): Basic Data 1, Basic Data 2, Classification, Purchasing



Basic data 1 | Basic data 2 | Classification | Purchasing | Foreign trade i...

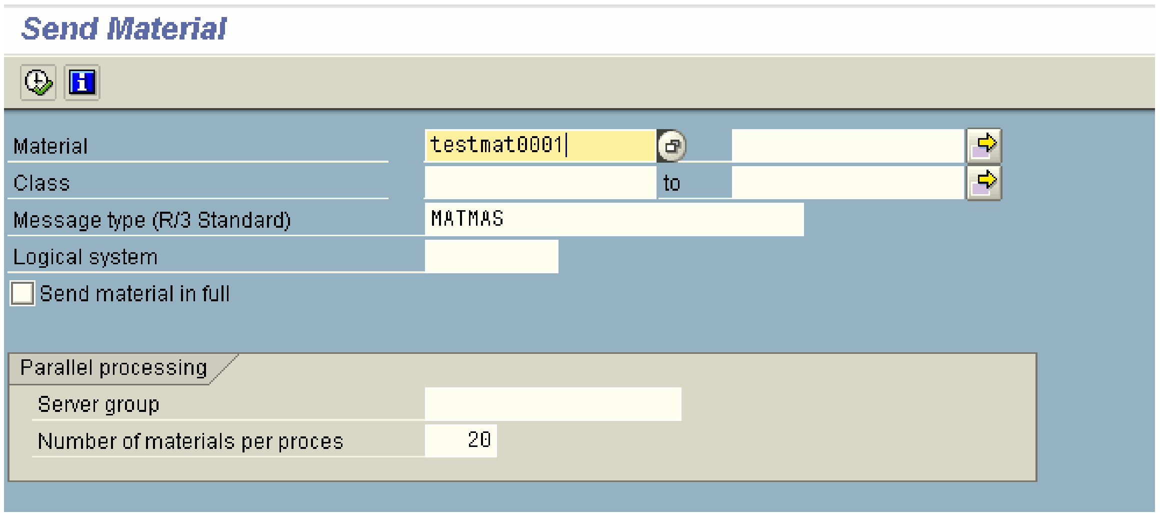
Material TESTMAT0001 test material 3

General data

Base unit of measure	ea	Material group	
Old material number		Ext. matl group	
Division			
Product allocation			

Material TESTMAT0001 created

3.1.9. Distributing Records in the Receiving logical system

Tcode	BD10
Done in	IBMSAP 111
Process	. 

Send Material

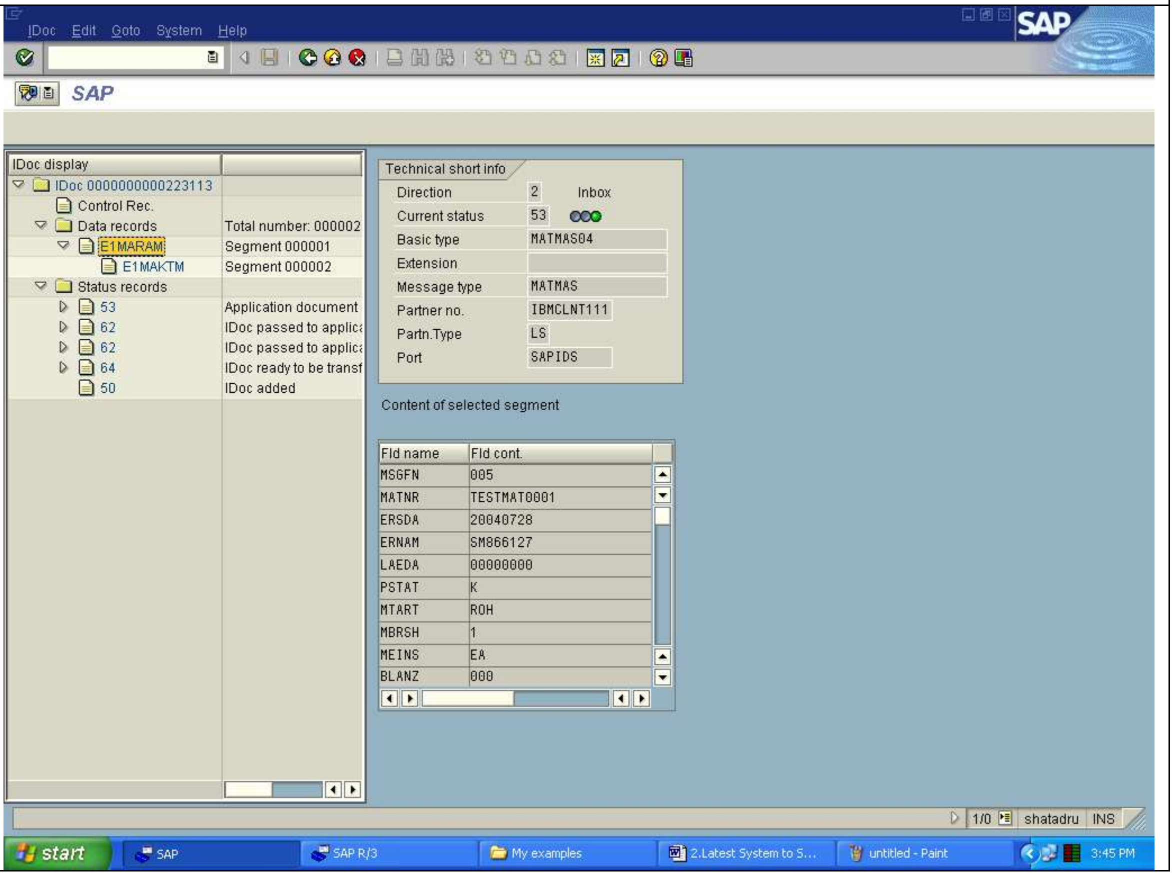
Material	testmat0001		
Class		to	
Message type (R/3 Standard)	MATMAS		
Logical system			

☐ Send material in full

Parallel processing

Server group	
Number of materials per proces	20

3.1.10. Checking IDoc status in the receiver system

Tcode	WE05
Done in	IBMSAP 111
	

IDoc display	
IDoc 0000000000223113	
Control Rec.	
Data records	Total number: 000002
E1MARAM	Segment 000001
E1MAKTM	Segment 000002
Status records	
53	Application document
62	IDoc passed to applic
62	IDoc passed to applic
64	IDoc ready to be transf
50	IDoc added

Technical short info

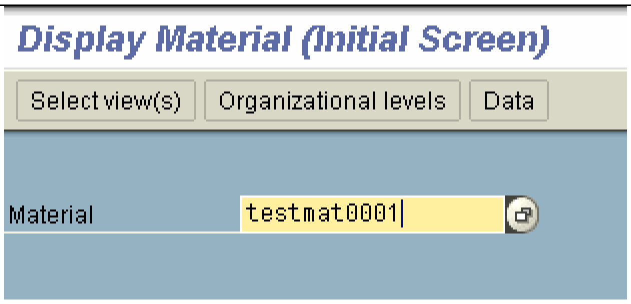
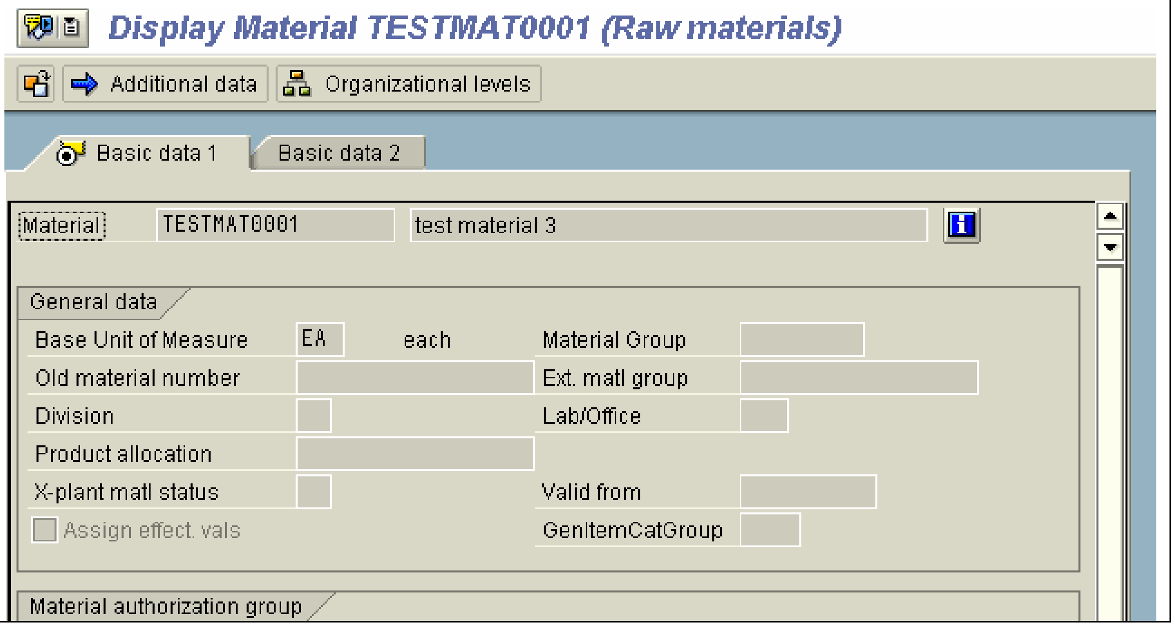
Direction	2	Inbox
Current status	53	●●●
Basic type	MATMAS04	
Extension		
Message type	MATMAS	
Partner no.	IBMCLNT111	
Partn.Type	LS	
Port	SAPIDS	

Content of selected segment

Fld name	Fld cont.
MSGFN	005
MATNR	TESTMAT0001
ERSDA	20040728
ERNAM	SM866127
LAEDA	00000000
PSTAT	K
MTART	ROH
MBRSH	1
MEINS	EA
BLANZ	000

1/0 shatadru INS
start SAP SAP R/3 My examples 2.Latest System to S... untitled - Paint 3:45 PM

3.1.11. Check Material in Receiver System

In	PwCSAP 555
Tcode	MM03
	 

Display Material (Initial Screen)

Material: testmat0001

Display Material TESTMAT0001 (Raw materials)

Material: TESTMAT0001 test material 3

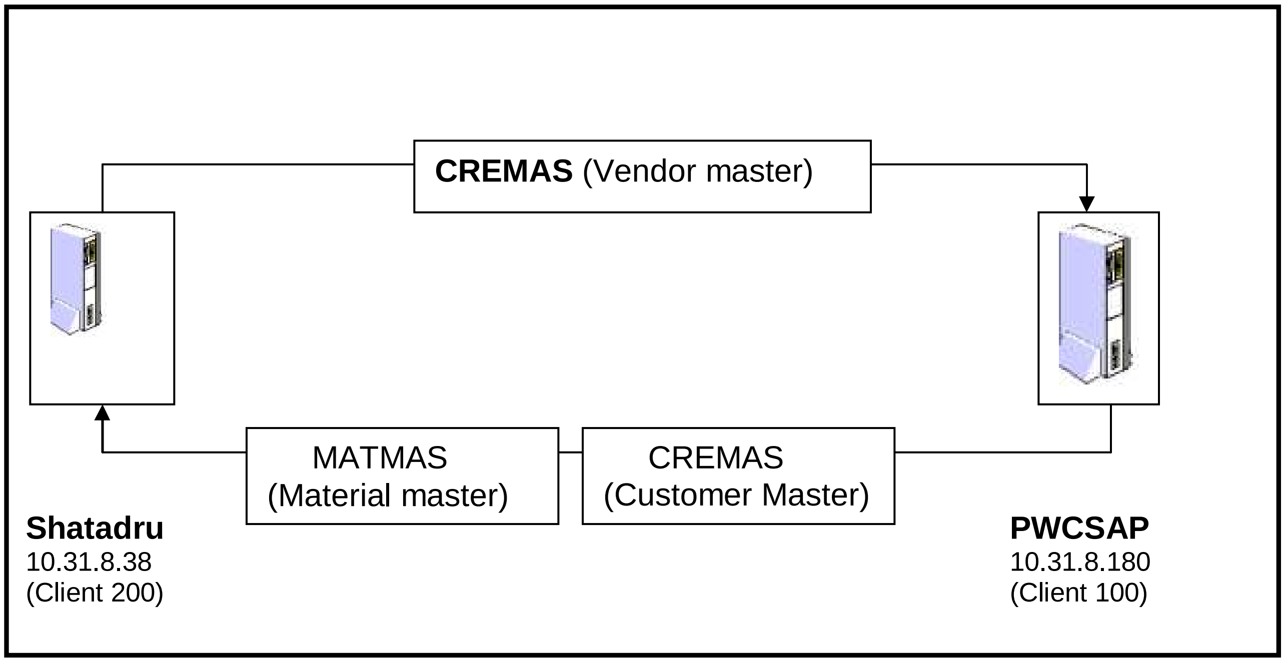
General data

Base Unit of Measure	EA each	Material Group	
Old material number		Ext. matl group	
Division		Lab/Office	
Product allocation			
X-plant matl status		Valid from	
Assign effect. vals		GenItemCatGroup	

Material authorization group

4. Two way Server to Server Communication with IDoc

This documentation assumes that the reader is acquainted with the popular terms for middleware technologies. This also assumes that the reader has gone through the earlier two documentations: Client-to-Client IDoc communication and Server-to-Server IDoc communication. Here, we shall discuss a more realistic approach; both the servers will be sending some message type to other. The details of the servers and the message types and IDoc types they exchange are shown below:-



As details of proceeding through the transactions have been provided in the earlier documentations, we shall mainly mention the steps, transaction code and screen shots for this project.

This documentation is mainly divided into three sections:-

- Customizing for Sending Vendor master from Shatadru to PWCSAP
- Additional customizations to be made for sending material master from PWCSAP to Shatadru
- Additional customizations to be made for sending Customer master(DEBMAS) from PWCSAP to Shatadru

Steps

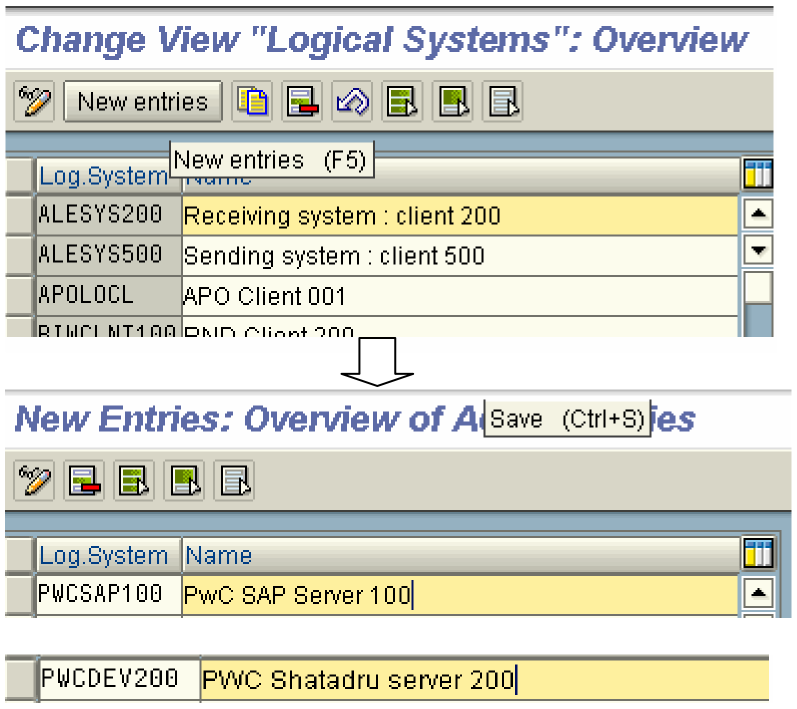
The steps to be followed are outlined below:-

4.1. Customizing for Sending Vendor master from Shatadru to PWCSAP

4.1.1. Create Logical system PWCSAP100 and PWCDEV200 both in Shatadru (200) and PWCSAP (100).

Transaction code: BD54

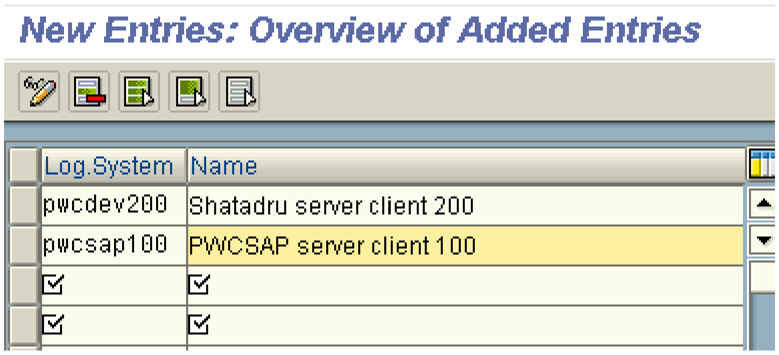
Create Logical System in Shatadru :-


Change View "Logical Systems": Overview
New entries
New entries (F5)
New Entries: Overview of A
Save (Ctrl+S)
es

Log.System	Name
ALESYS200	Receiving system : client 200
ALESYS500	Sending system : client 500
APOLOCL	APO Client 001
BIWCLNT100	BND Client 200

Log.System	Name
PWCSAP100	PwC SAP Server 100
PWCDEV200	PWC Shatadru server 200

Similarly, create the two logical systems in PWCSAP



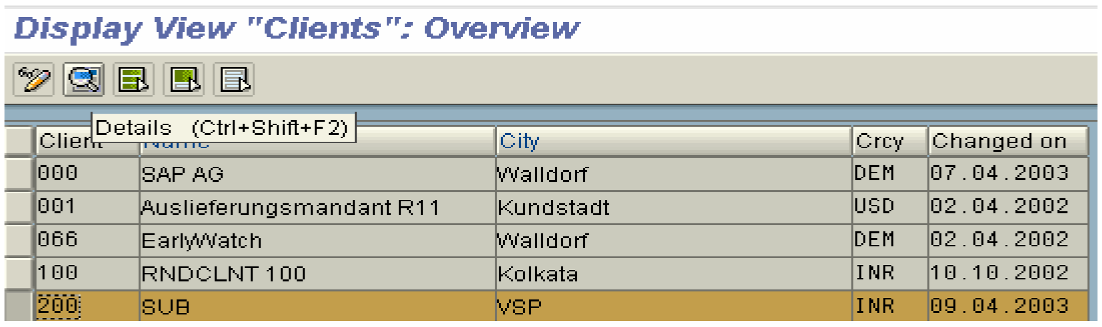
New Entries: Overview of Added Entries

Log.System	Name
pwcdev200	Shatadru server client 200
pwcsap100	PWCSAP server client 100
☑	☑
☑	☑

4.1.2. Assign Logical System PWCSAP100 to client 100 in PWCSAP and PWCDEV200 to client 200 in Shatadru

Transaction Code: SCC4

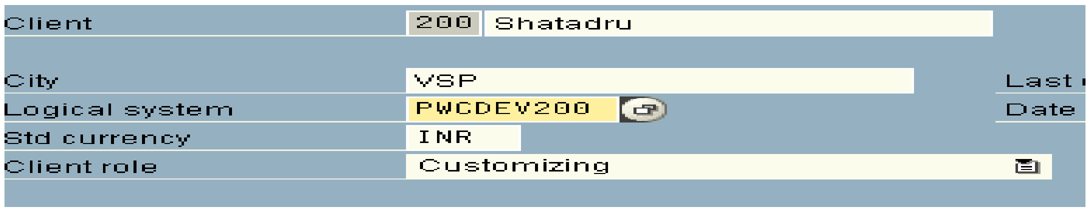
<u>Assign in Shatadru</u>



Display View "Clients": Overview

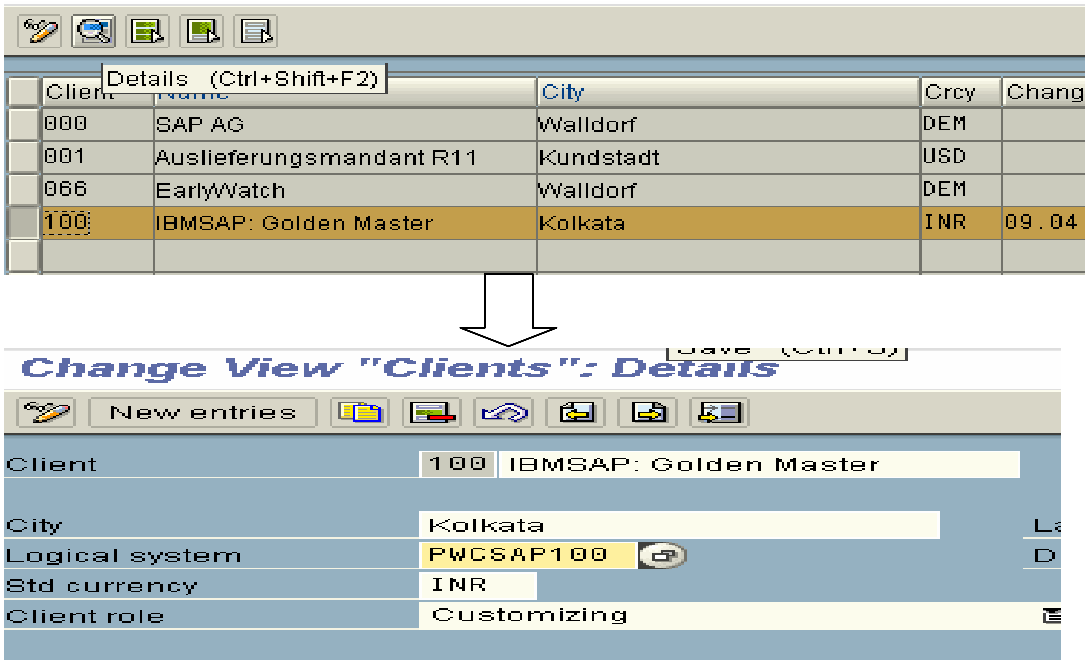
Details (Ctrl+Shift+F2)

Client	Name	City	Crcy	Changed on
000	SAP AG	Walldorf	DEM	07.04.2003
001	Auslieferungsmandant R11	Kundstadt	USD	02.04.2002
066	EarlyWatch	Walldorf	DEM	02.04.2002
100	RNDCLNT 100	Kolkata	INR	10.10.2002
200	SUB	VSP	INR	09.04.2003



Client	200 Shatadru
City	VSP
Logical system	PWCDEV200
Std currency	INR
Client role	Customizing

Assign in PWCSAP



Client	Name	City	Crcy	Chang
000	SAP AG	Walldorf	DEM	
001	Auslieferungsmandant R11	Kundstadt	USD	
066	EarlyWatch	Walldorf	DEM	
100	IBMSAP: Golden Master	Kolkata	INR	09.04

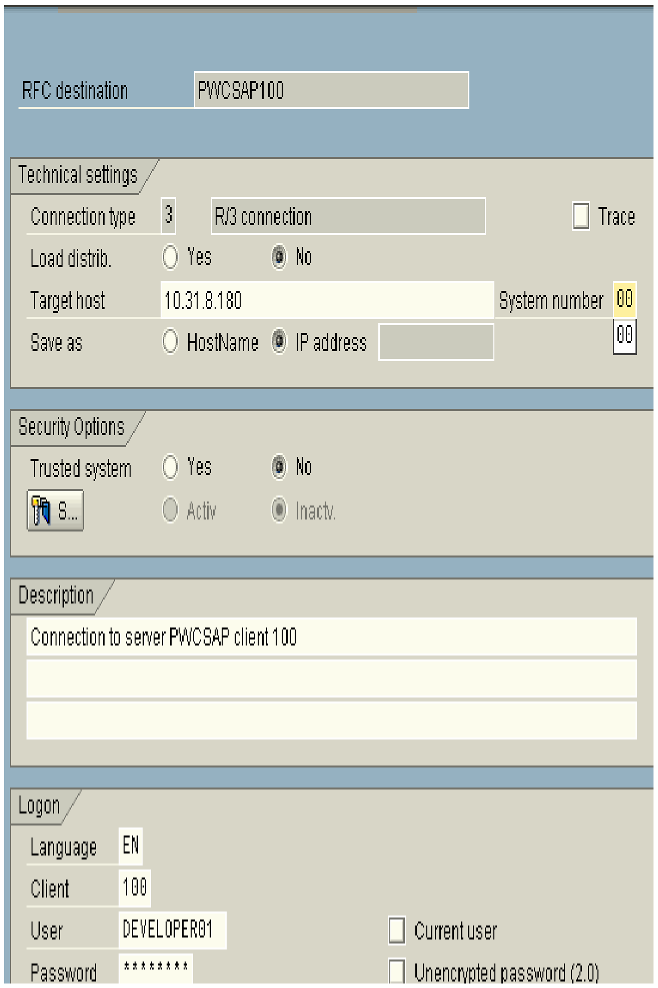
Change View "Clients": Details

Client	100 IBMSAP: Golden Master
City	Kolkata
Logical system	PWCSAP100
Std currency	INR
Client role	Customizing

4.1.3. Create RFC Destination PWCSAP100 in Shatadru and PWCDEV200 in PWCSAP

Transaction code: SM59

In Shatadru **In PWCSAP**



RFC destination: PWCSAP100

Technical settings

Connection type: 3 R/3 connection ☐ Trace

Load distrib.: Yes [✓ No]

Target host: 10.31.8.180 System number: 00

Save as: HostName [✓ IP address] 00

Security Options

Trusted system: Yes [✓ No]

S... Activ [✓ Inactv.]

Description

Connection to server PWCSAP client 100

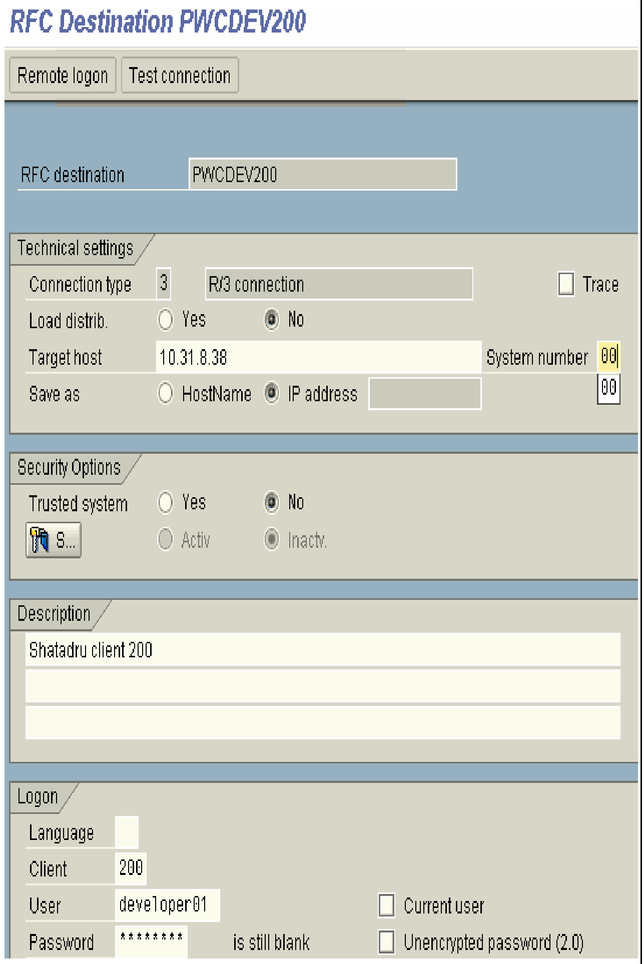
Logon

Language: EN

Client: 100

User: DEVELOPER01 ☐ Current user

Password: ******** ☐ Unencrypted password (2.0)



RFC Destination PWCDEV200

Remote logon | Test connection

RFC destination: PWCDEV200

Technical settings

Connection type: 3 R/3 connection ☐ Trace

Load distrib.: Yes [✓ No]

Target host: 10.31.8.38 System number: 00

Save as: HostName [✓ IP address] 00

Security Options

Trusted system: Yes [✓ No]

S... Activ [✓ Inactv.]

Description

Shatadru client 200

Logon

Language:

Client: 200

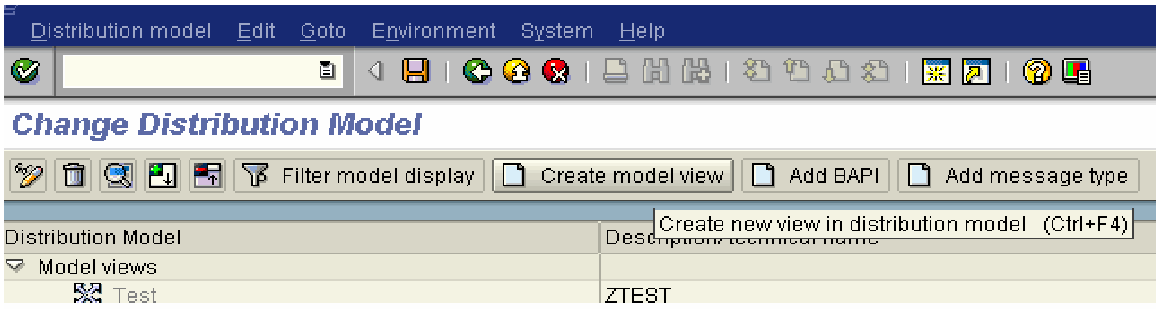
User: developer01 ☐ Current user

Password: ******** is still blank ☐ Unencrypted password (2.0)

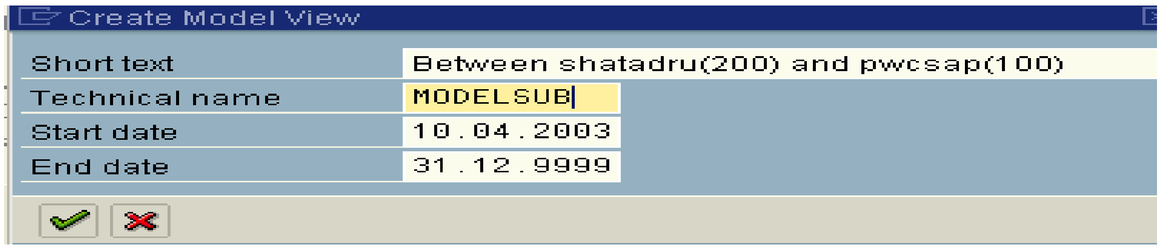
After creating and saving the RFC destinations, test the connections by pressing the pushbutton **Remote Logon** from the Application Toolbar.

4.1.4. Creating Distribution Model MODELSUB in Shatadru

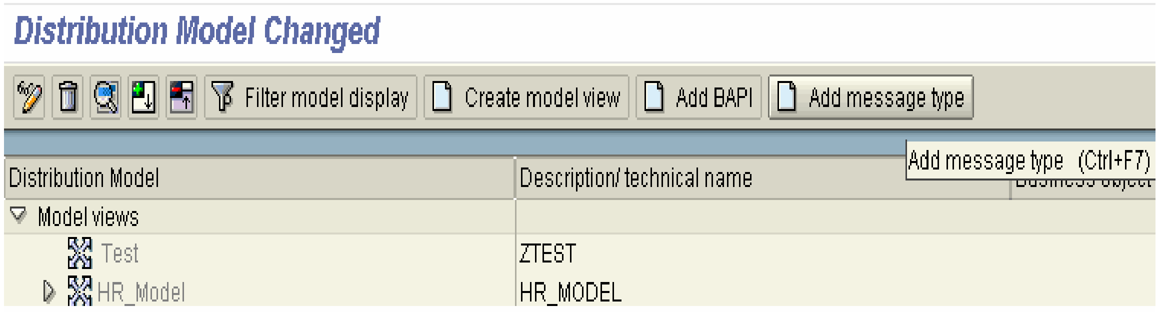
Transaction code: BD64


Distribution model Edit Goto Environment System Help
Change Distribution Model
Filter model display
Create model view
Add BAPI
Add message type
Create new view in distribution model (Ctrl+F4)

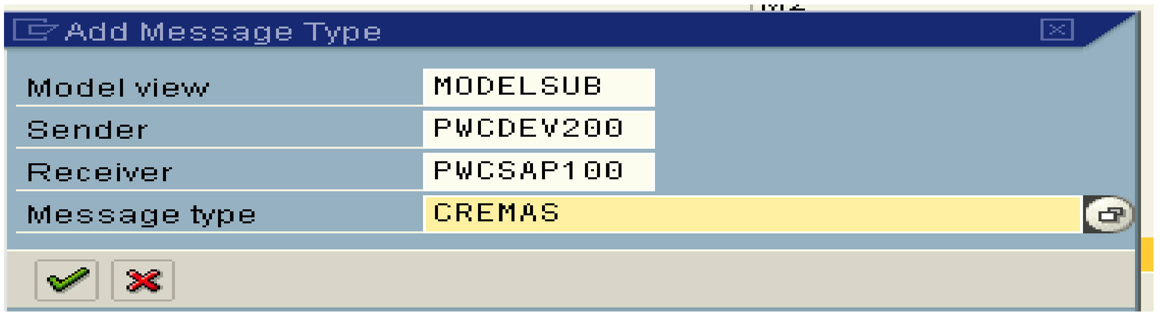
Distribution Model	Description/ technical name
Model views	
Test	ZTEST



Short text	Between shatadru(200) and pwcsap(100)
Technical name	MODELSUB
Start date	10.04.2003
End date	31.12.9999



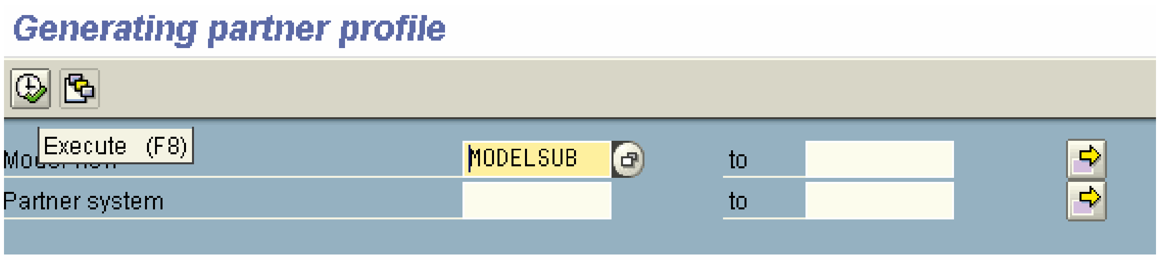
Distribution Model	Description/ technical name
Model views	
Test	ZTEST
HR_Model	HR_MODEL



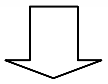
Model view	MODELSUB
Sender	PWCDEV200
Receiver	PWCSAP100
Message type	CREMAS

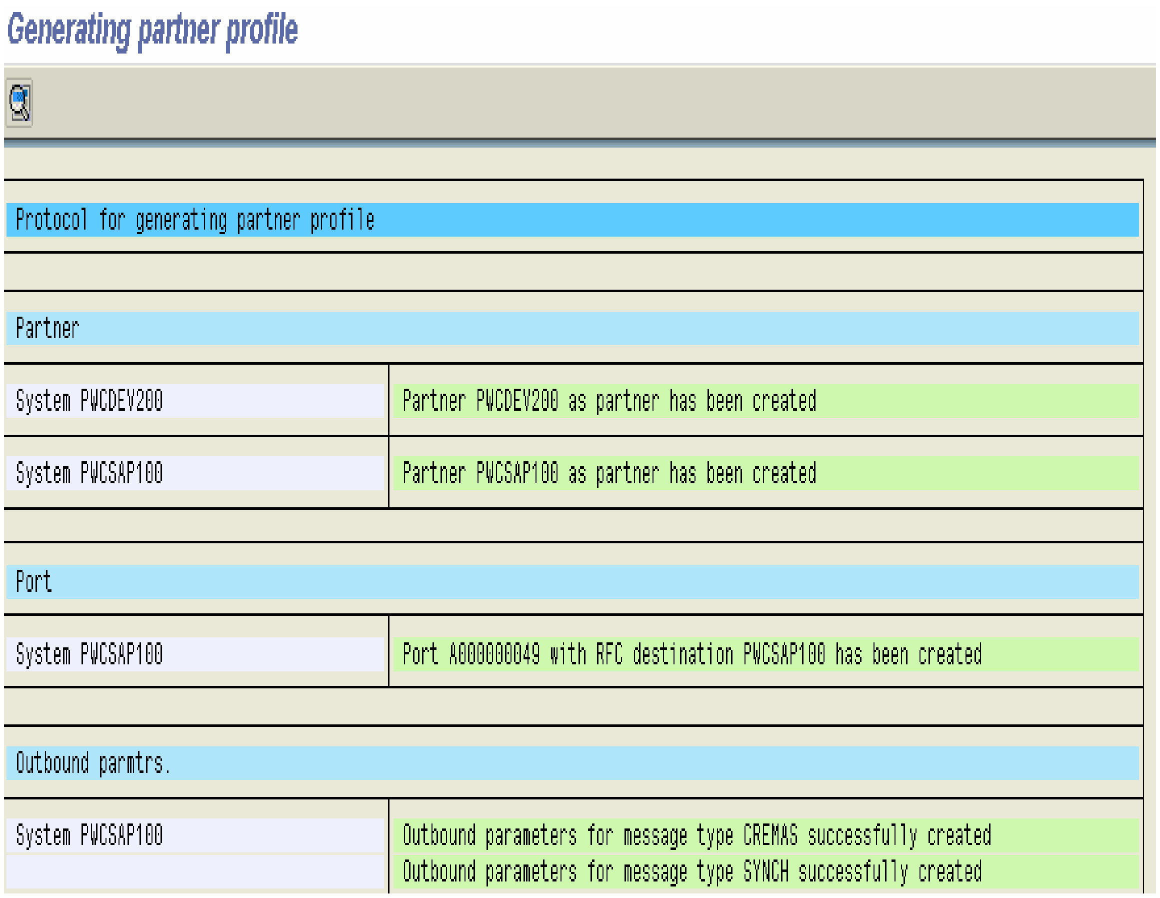
4.1.5. Creating Partner Profile in Shatadru

Transaction Code: BD82



Model view	MODELSUB	to	
Partner system		to	

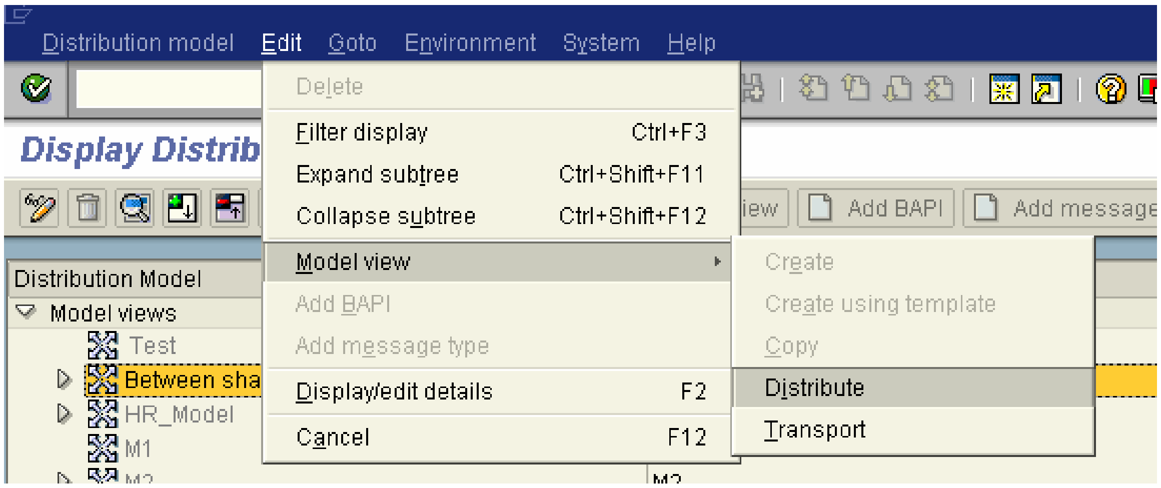


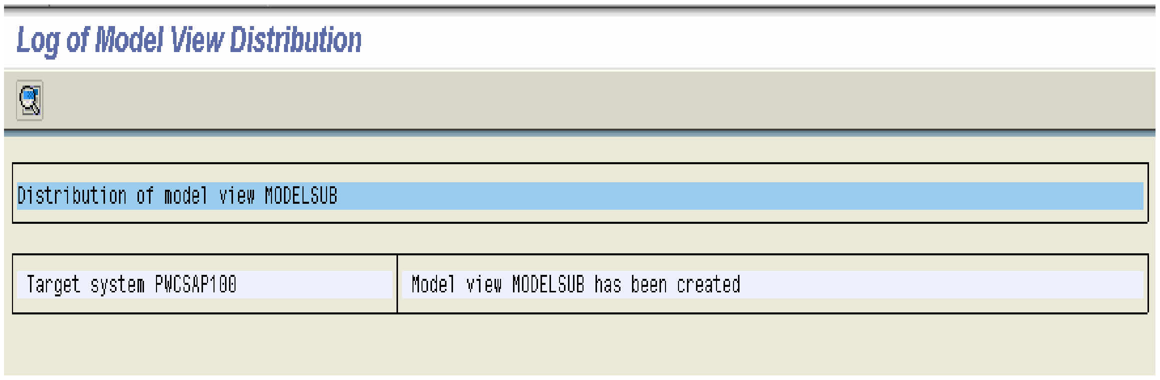


```
Protocol for generating partner profile

Partner
System PWCDEV200    Partner PWCDEV200 as partner has been created
System PWCSAP100    Partner PWCSAP100 as partner has been created

Port
System PWCSAP100    Port A000000049 with RFC destination PWCSAP100 has been created

Outbound parmtrs.
System PWCSAP100    Outbound parameters for message type CREMAS successfully created
                    Outbound parameters for message type SYNCH successfully created
```

4.1.6. Distributing Customer Distribution Model in Shatadru

Transaction Code: BD64

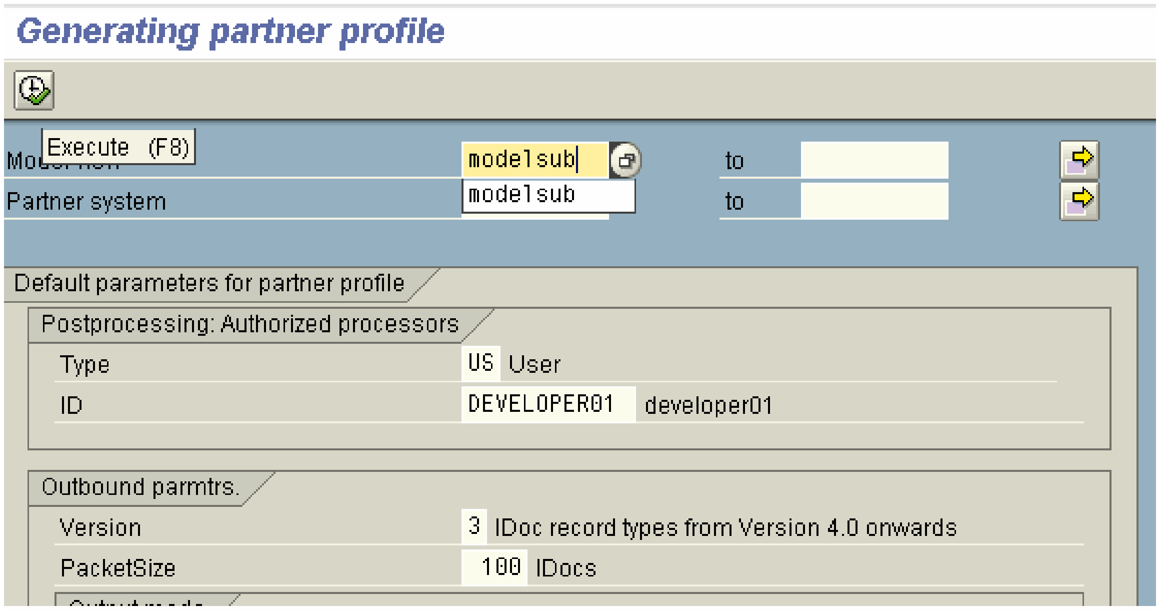




```
Distribution of model view MODELSUB

Target system PWCSAP100    Model view MODELSUB has been created
```

4.1.7. Generating Partner Profile in PWCSAP

Transaction Code: BD82



Generating partner profile

Execute (F8)

Model view	modelsub	to	
Partner system	modelsub	to	

Default parameters for partner profile

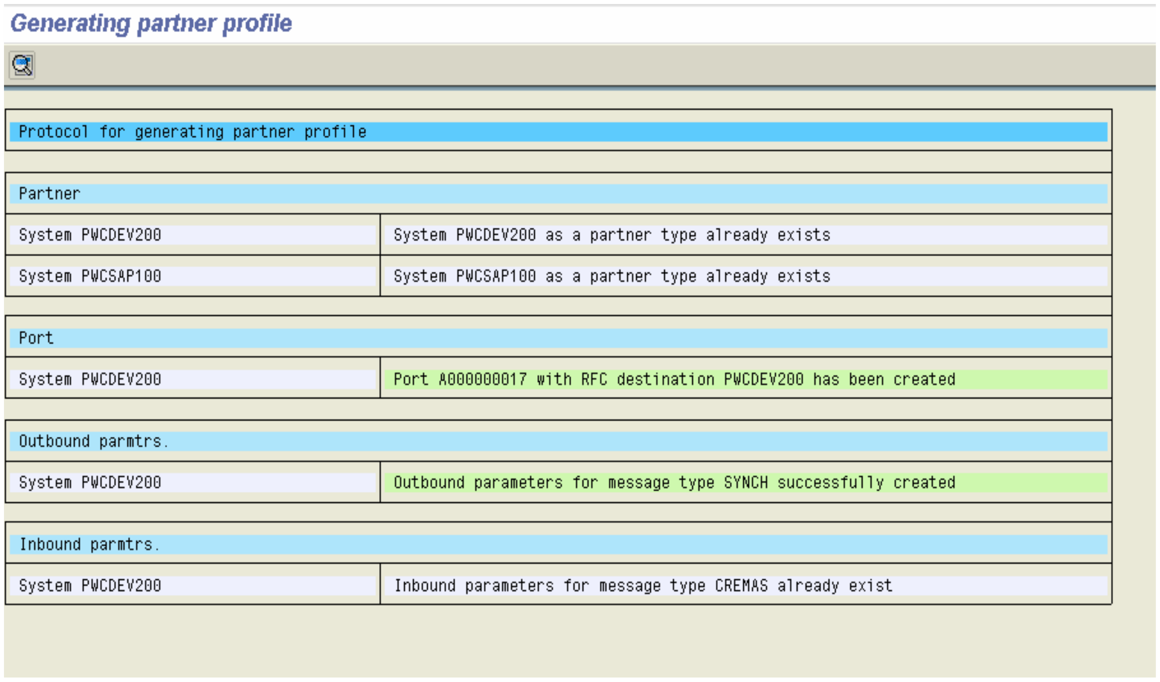
Postprocessing: Authorized processors

Type	US	User
ID	DEVELOPER01	developer01

Outbound parmtrs.

Version	3	IDoc record types from Version 4.0 onwards
PacketSize	100	IDocs

⇩

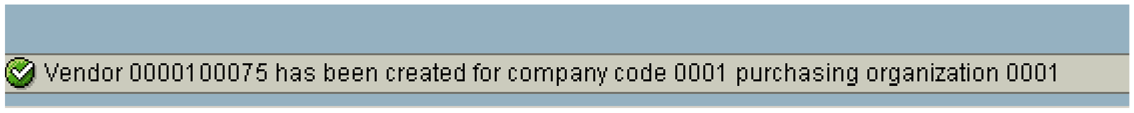


Generating partner profile

```
Protocol for generating partner profile

Partner
System PWCDEV200      System PWCDEV200 as a partner type already exists
System PWCSAP100      System PWCSAP100 as a partner type already exists

Port
System PWCDEV200      Port A000000017 with RFC destination PWCDEV200 has been created

Outbound parmtrs.
System PWCDEV200      Outbound parameters for message type SYNCH successfully created

Inbound parmtrs.
System PWCDEV200      Inbound parameters for message type CREMAS already exist
```

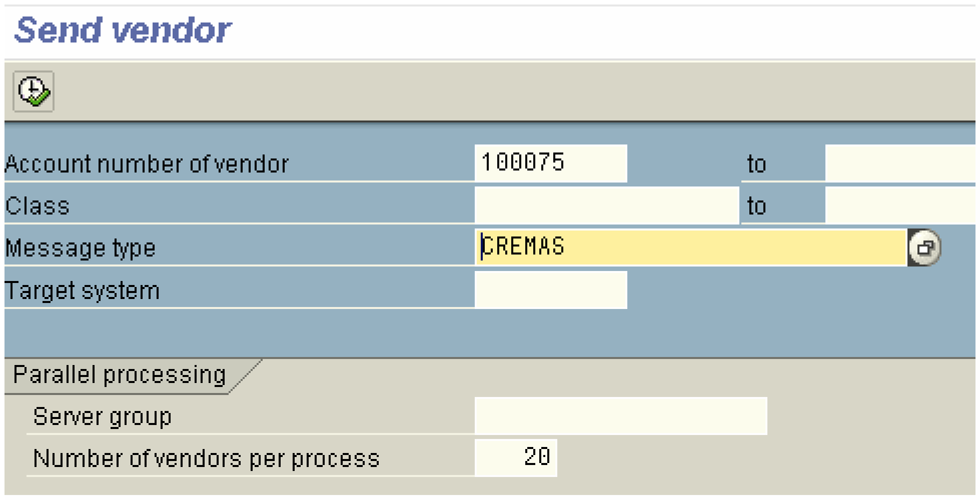
4.1.8. Creating Vendor in Shatadru

Transaction Code : XK01



Vendor 0000100075 has been created for company code 0001 purchasing organization 0001

4.1.9. Creation of IDOC in Shatadru by BD14

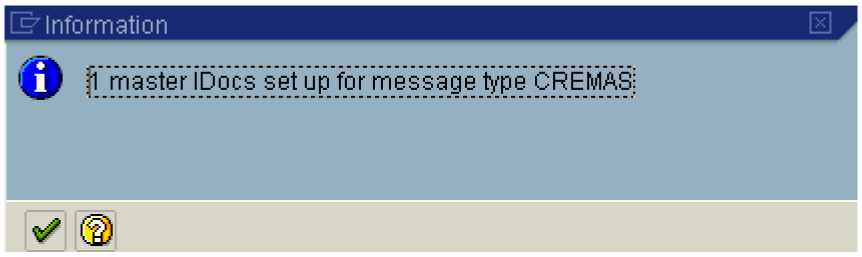


Send vendor

Account number of vendor	100075	to	
Class		to	
Message type	CREMAS		
Target system			

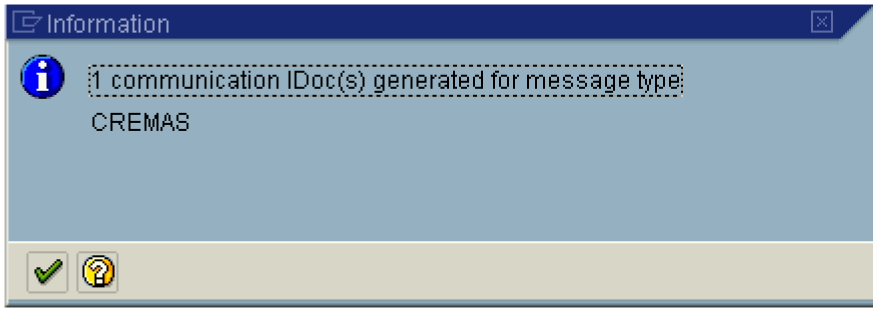
Parallel processing

Server group	
Number of vendors per process	20



Information

1 master IDocs set up for message type CREMAS

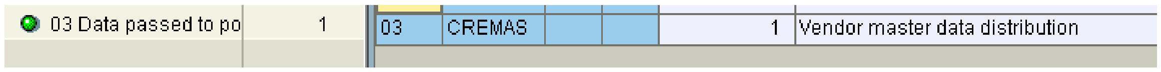


Information

1 communication IDoc(s) generated for message type CREMAS

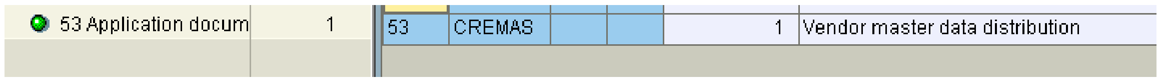
4.1.10. View Status of IDoc in Shatadru by WE05

Transaction code: WE05



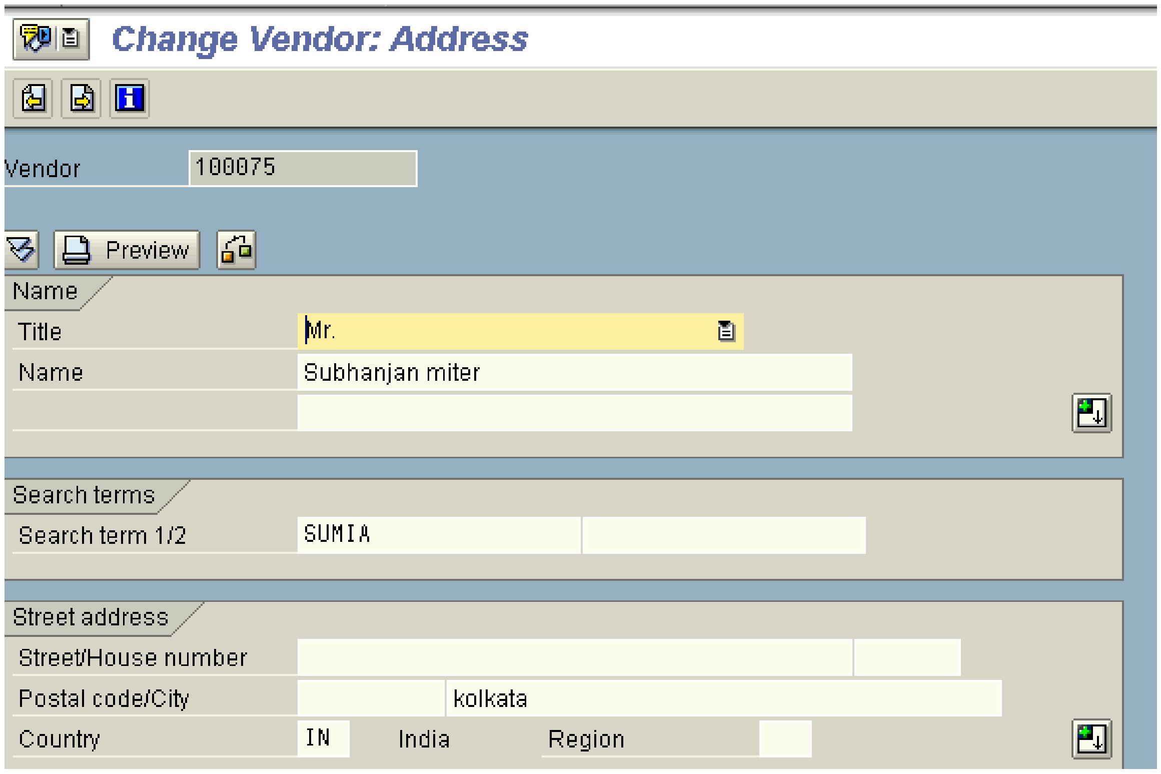
● 03 Data passed to po	1	03	CREMAS			1	Vendor master data distribution

4.1.11. View Status of IDoc in PWCSAP by WE05



● 53 Application docum	1	53	CREMAS			1	Vendor master data distribution

4.1.12. Vendor Successfully created in PWCSAP



Change Vendor: Address

Vendor 100075

Preview

Name

Title	Mr.
Name	Subhanjan miter

Search terms

Search term 1/2	SUMIA

Street address

Street/House number			
Postal code/City		kolkata	
Country	IN	India	Region

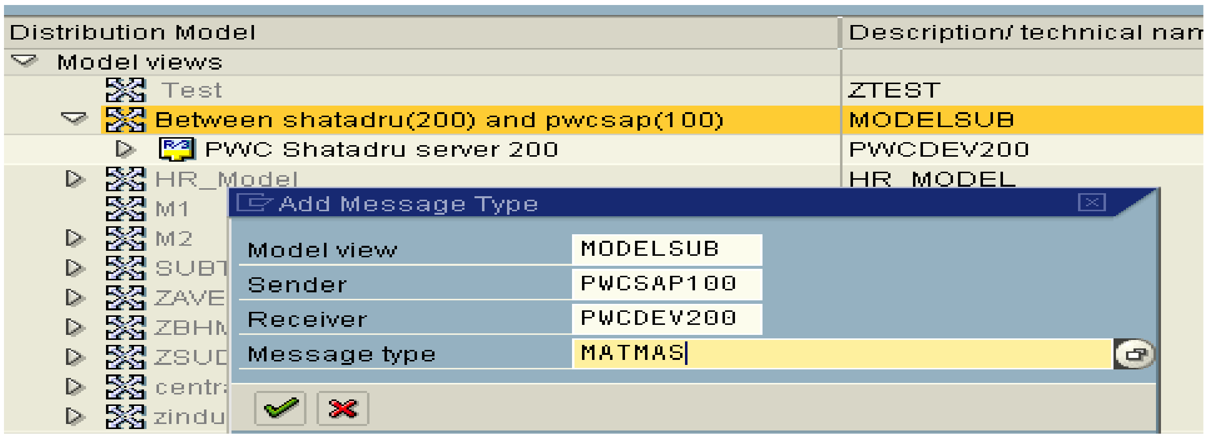
4.2. Additional customizations to be made for sending material master from PWCSAP to Shatadru

Once the settings for sending vendor master from Shatadru to PWCSAP is complete, we want to customize the system so that PWCSAP can send material master data to Shatadru. For that, we need to :-

- ❖ Create one additional message type in customer distribution model MODELSUB
- ❖ Generate the partner profile in Shatadru
- ❖ Distribute the customer distribution model MODELSUB from Shatadru
- ❖ Generate the partner profile from PWCSAP

4.2.1. Create one additional message type in customer distribution model MODELSUB in system Shatadru (200)

Transaction: BD64

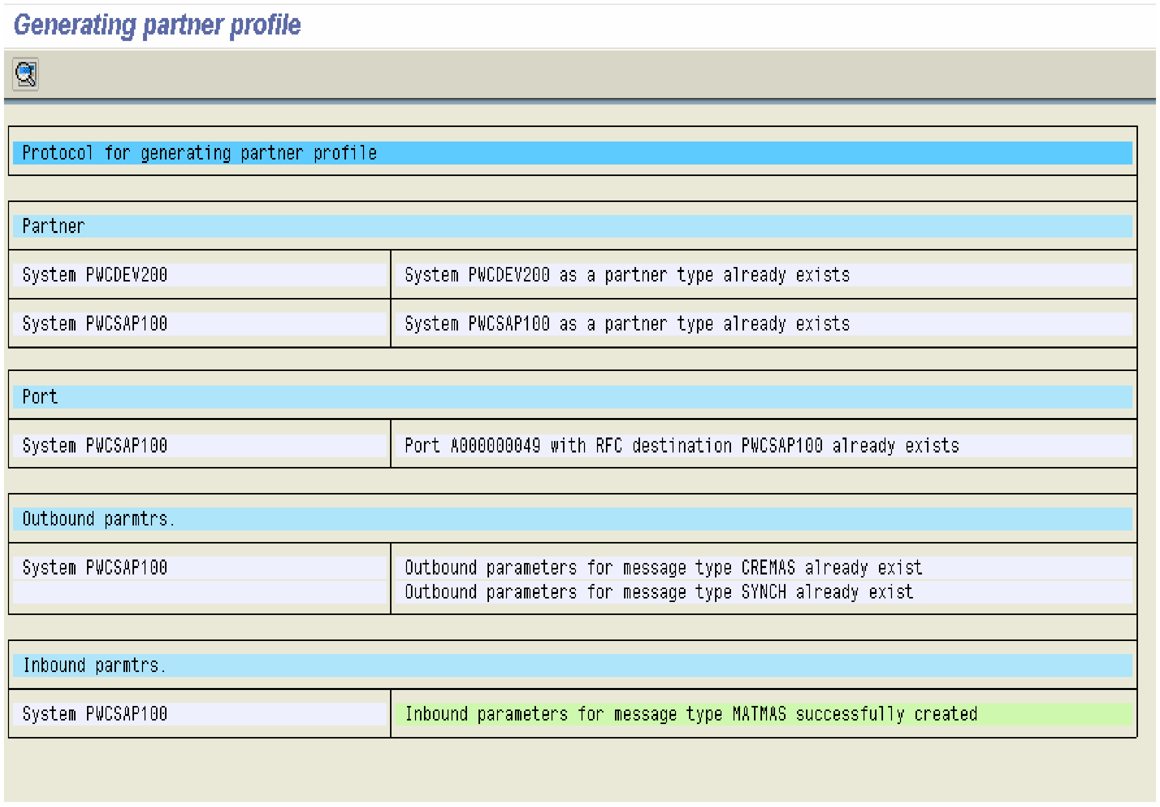


Model view	MODELSUB
Sender	PWCSAP100
Receiver	PWCDEV200
Message type	MATMAS

4.2.2. Generate the partner profile in Shatadru

Transaction: BD82

Fill in the selection-screen with the name of the customer distribution model and press **Execute**

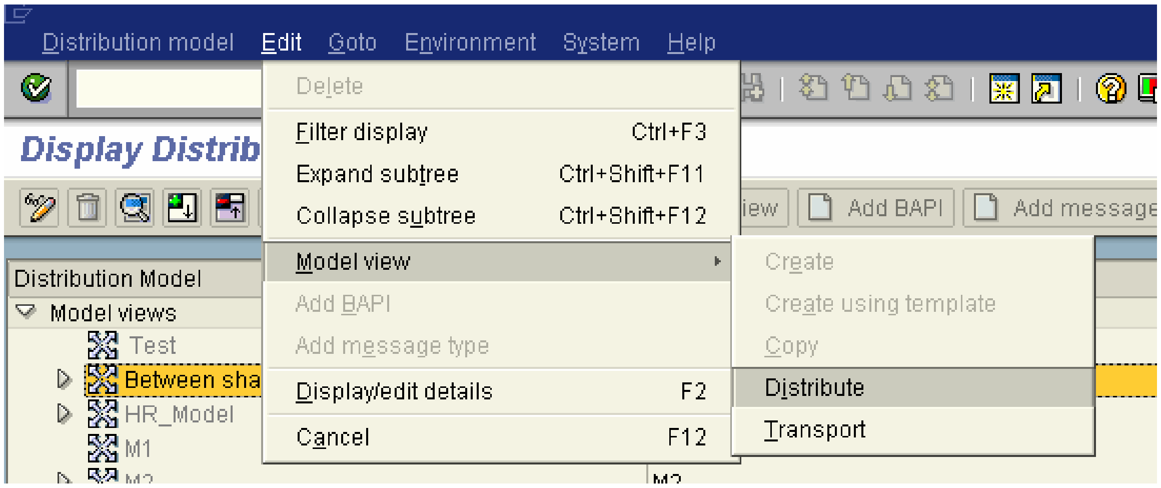


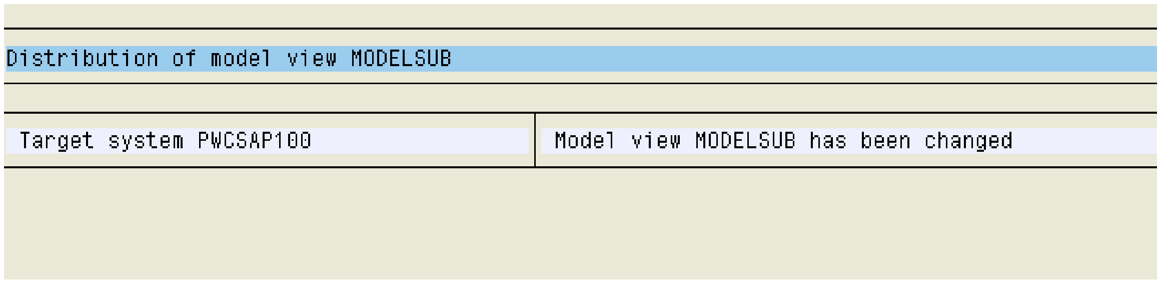
Generating partner profile

Protocol for generating partner profile	
Partner	
System PWCDEV200	System PWCDEV200 as a partner type already exists
System PWCSAP100	System PWCSAP100 as a partner type already exists
Port	
System PWCSAP100	Port A000000049 with RFC destination PWCSAP100 already exists
Outbound parmtrs.	
System PWCSAP100	Outbound parameters for message type CREMAS already exist Outbound parameters for message type SYNCH already exist
Inbound parmtrs.	
System PWCSAP100	Inbound parameters for message type MATMAS successfully created

4.2.3. Distribute the customer distribution model MODELSUB from Shatadru

Transaction: BD64

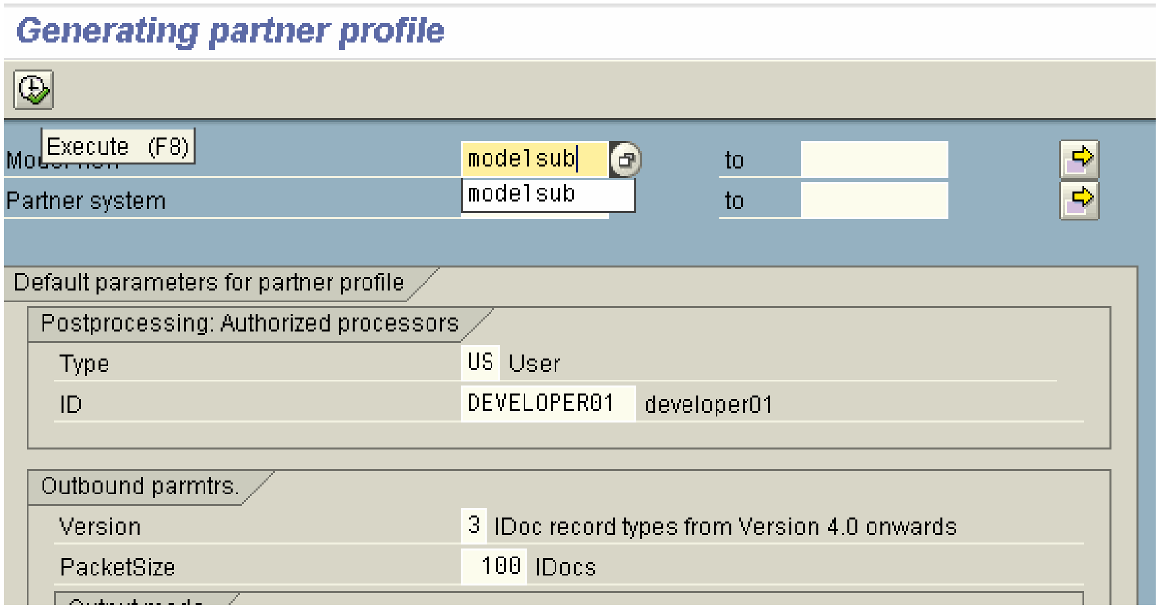




```
Distribution of model view MODELSUB

Target system PWCSAP100          Model view MODELSUB has been changed
```

4.2.4. Generate the partner profile from PWCSAP

Transaction: BD82



Generating partner profile

Execute (F8)

Model view	modelsub	to	
Partner system	modelsub	to	

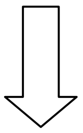
Default parameters for partner profile

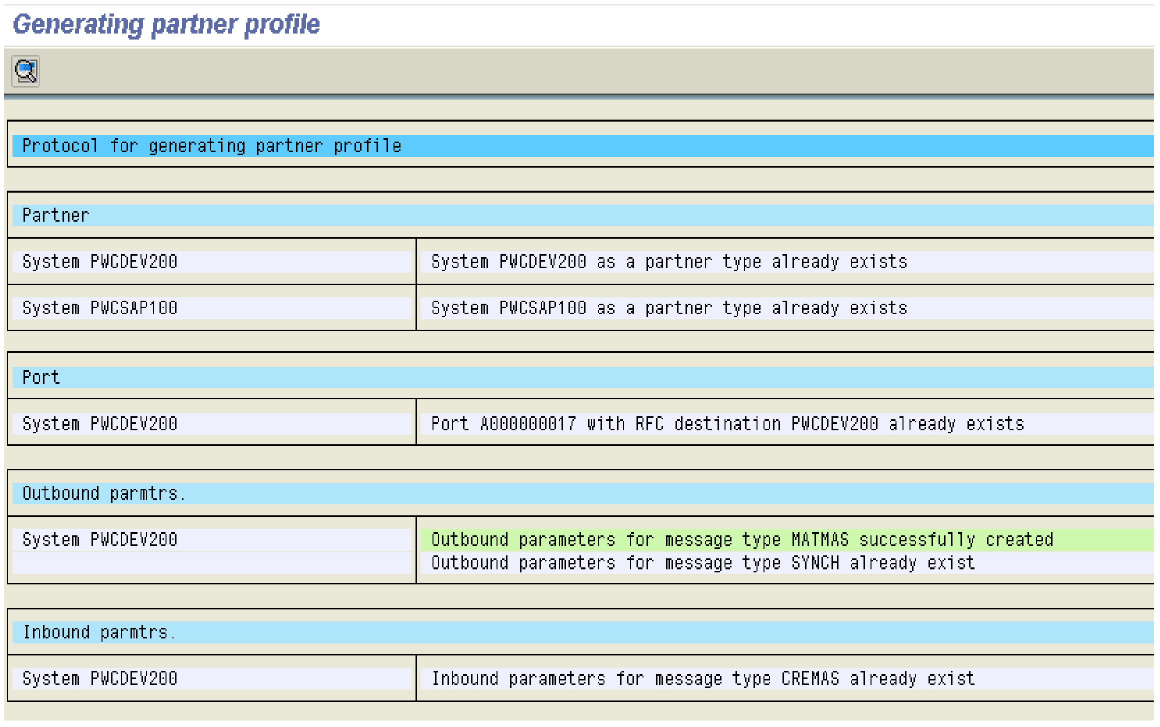
Postprocessing: Authorized processors

Type	US	User
ID	DEVELOPER01	developer01

Outbound parmtrs.

Version	3	IDoc record types from Version 4.0 onwards
PacketSize	100	IDocs



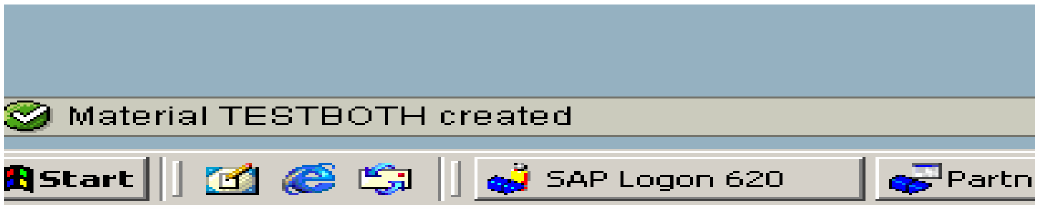


Generating partner profile

```
Protocol for generating partner profile

Partner
System PWCDEV200      System PWCDEV200 as a partner type already exists
System PWCSAP100      System PWCSAP100 as a partner type already exists

Port
System PWCDEV200      Port A000000017 with RFC destination PWCDEV200 already exists

Outbound parmtrs.
System PWCDEV200      Outbound parameters for message type MATMAS successfully created
                      Outbound parameters for message type SYNCH already exist

Inbound parmtrs.
System PWCDEV200      Inbound parameters for message type CREMAS already exist
```

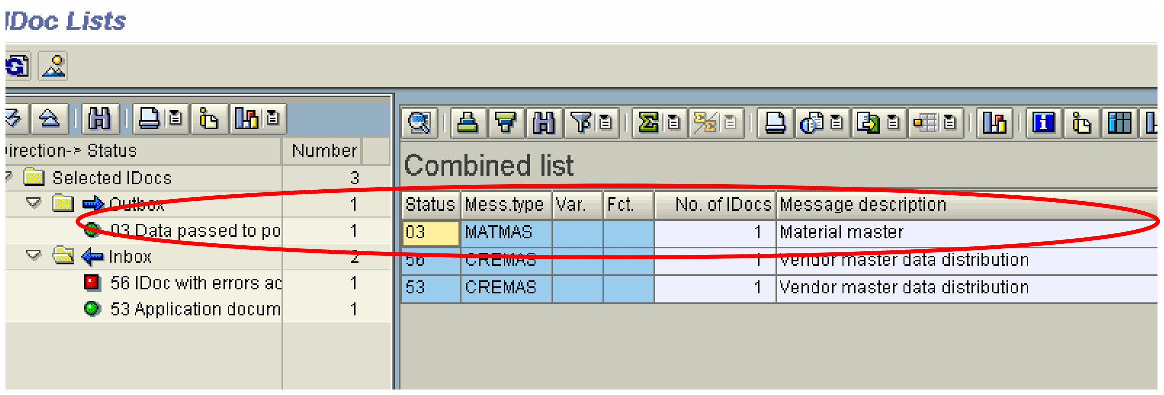
4.2.5. Create Material master in PWCSAP and distribute it using transaction BD10

Then, create a material using MM01/MMR1 and generate the IDoc using transaction code BD10. For details, you can consult my documentation on : Client to Client Communication by IDoc.


Material TESTBOTH created

4.2.6. View Status of the IDoc in PWCSAP (Sender system)

Transaction: WE05


IDoc Lists

irection-> Status	Number
Selected IDocs	3
Outbox	1
03 Data passed to po	1
Inbox	2
56 IDoc with errors ac	1
53 Application docum	1

Combined list

Status	Mess.type	Var.	Fct.	No. of IDocs	Message description
03	MATMAS			1	Material master
56	CREMAS			1	Vendor master data distribution
53	CREMAS			1	Vendor master data distribution

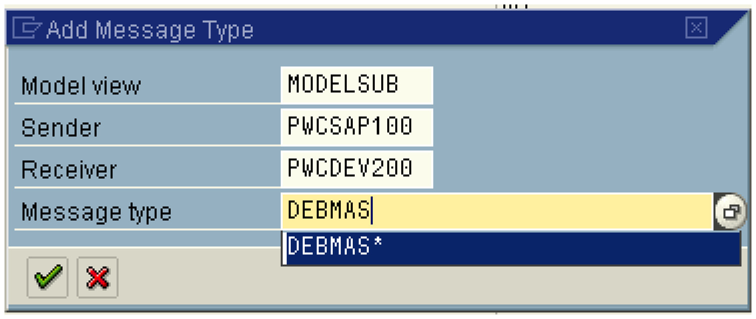
4.3. Customization for Receiving Customer Master into Shatadru from PWCSAP

The steps are similar to that done for receiving material master IDoc from PWCSAP into Shatadru. The steps are:-

- ❖ Adding one message type with PWCSAP100 as sender and PWCDEV200 as receiver
- ❖ Generating Partner profile in Shatadru
- ❖ Distributing Customer Distribution Model from Shatadru
- ❖ Generating Partner Profile from PWCSAP
- ❖ Creating a Customer Master
- ❖ Generate IDOC in PWCSAP
- ❖ Verify the status of the IDoc in PWCSAP
- ❖ Verify the status in Shatadru
- ❖ Verify the customer in Shatadru

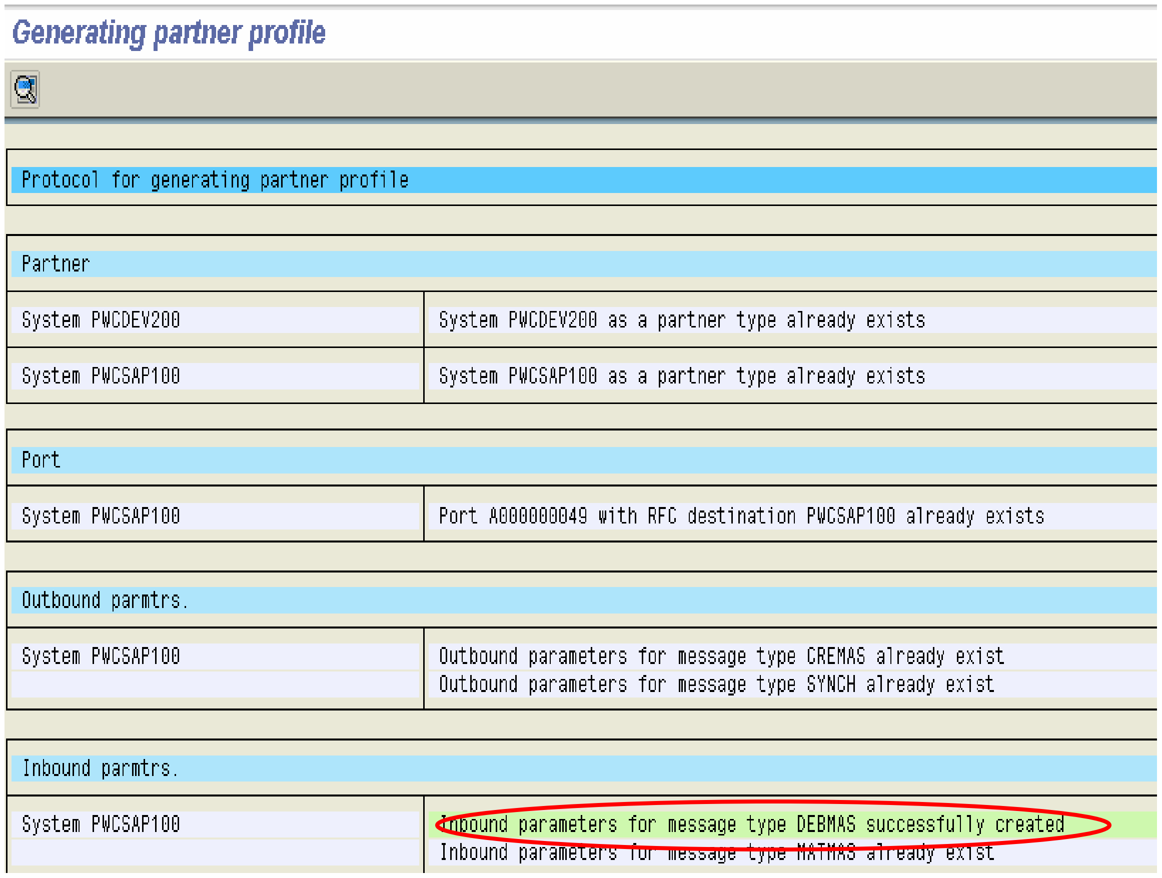
4.3.1. Adding one message type with PWCSAP100 as sender and PWCDEV200 as receiver in Shatadru

Transaction: BD64



Model view	MODELSUB
Sender	PWCSAP100
Receiver	PWCDEV200
Message type	DEBMAS

4.3.2. Generate the Partner Profile in Shatadru for the customer model



Protocol for generating partner profile

Partner	
System PWCDEV200	System PWCDEV200 as a partner type already exists
System PWCSAP100	System PWCSAP100 as a partner type already exists

Port	
System PWCSAP100	Port A000000049 with RFC destination PWCSAP100 already exists

Outbound parmtrs.	
System PWCSAP100	Outbound parameters for message type CREMAS already exist Outbound parameters for message type SYNCH already exist

Inbound parmtrs.	
System PWCSAP100	Inbound parameters for message type DEBMAS successfully created Inbound parameters for message type MATMAS already exist

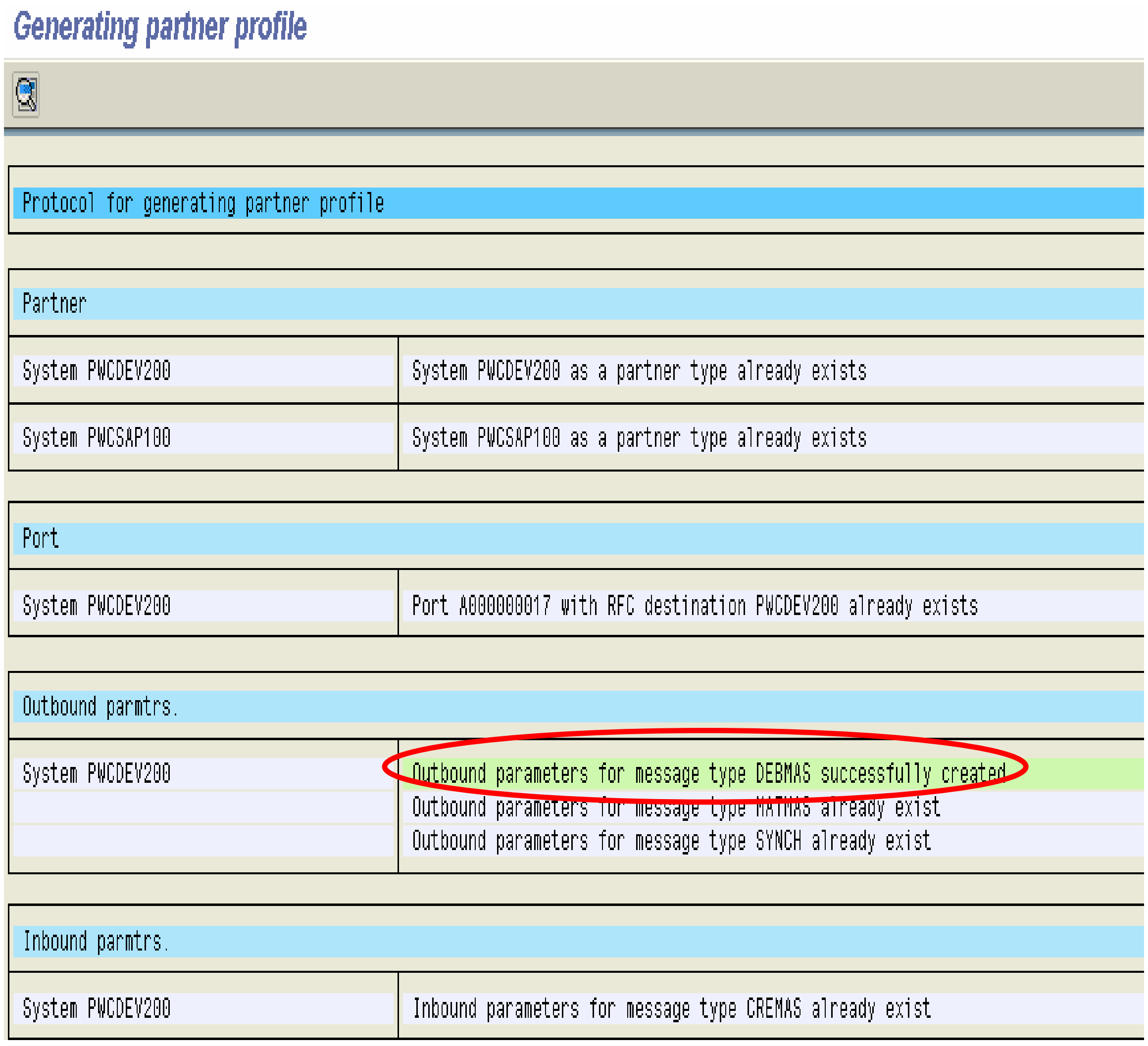
4.3.3. Distributing Customer Distribution Model from Shatadru

Transaction: BD64

Similar to the processes described earlier for distributing CDM.

4.3.4. Generating Partner Profile from PWCSAP

Transaction: BD82


Generating partner profile

Protocol for generating partner profile

Partner	
System PWCDEV200	System PWCDEV200 as a partner type already exists
System PWCSAP100	System PWCSAP100 as a partner type already exists

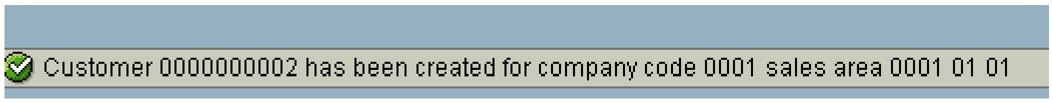
Port	
System PWCDEV200	Port A000000017 with RFC destination PWCDEV200 already exists

Outbound parmtrs.	
System PWCDEV200	Outbound parameters for message type DEBMAS successfully created
	Outbound parameters for message type MATMAS already exist
	Outbound parameters for message type SYNCH already exist

Inbound parmtrs.	
System PWCDEV200	Inbound parameters for message type CREMAS already exist

4.3.5. Create Customer Master in PWCSAP

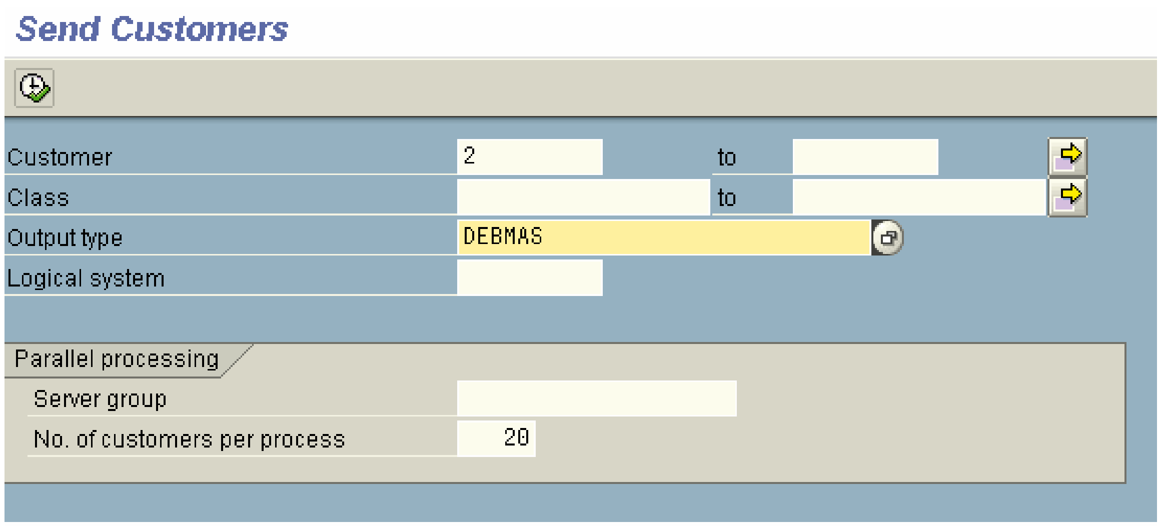
Transaction: XD01



Customer 0000000002 has been created for company code 0001 sales area 0001 01 01

4.3.6. Create IDoc for Customer Master in PWCSAP

Transaction: BD12



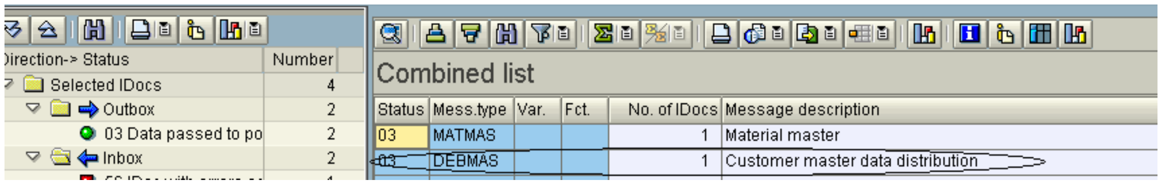
Send Customers

Customer	2	to	
Class		to	
Output type	DEBMAS		
Logical system			

Parallel processing

Server group	
No. of customers per process	20

Press Execute. One Master IDoc and one Communication IDoc will be created in the system. Verify the status of the IDoc in PWCSAP using transaction WE05



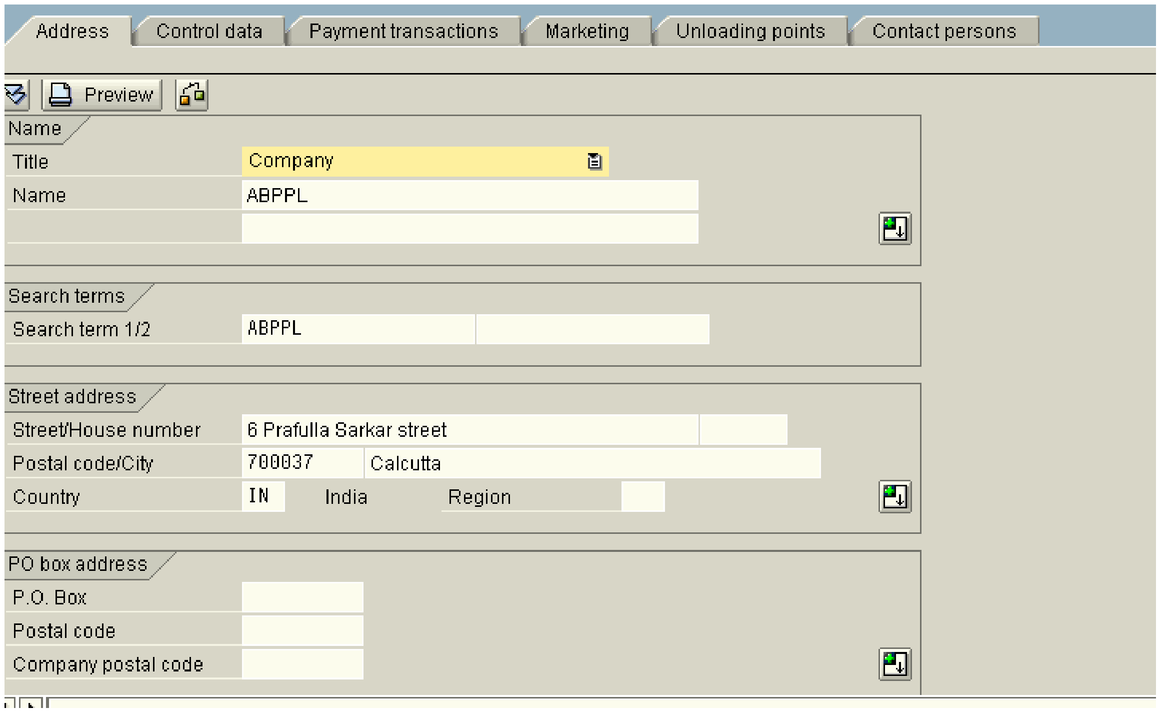
Direction-> Status	Number
Selected IDocs	4
Outbox	2
03 Data passed to po	2
Inbox	2

Combined list

Status	Mess.type	Var.	Fct.	No. of IDocs	Message description
03	MATMAS			1	Material master
03	DEBMAS			1	Customer master data distribution

4.3.7. Verify the Customer in the Receiver (Shatadru) system

Transaction: XD02/XD03



Name

Title	Company	
Name	ABPPL	

Search terms

Search term 1/2	ABPPL	

Street address

Street/House number	6 Prafulla Sarkar street		
Postal code/City	700037	Calcutta	
Country	IN	India	Region

PO box address

P.O. Box	
Postal code	
Company postal code	

5. Some important topics on IDoc

5.1. T-RFC Reporting

Program: RBDMOIND, transaction code: BD75 is scheduled or executed online to determine whether the communication of IDoc from sending to receiving system is successful. If the IDoc is dispatched to the destination system, the status becomes 12(Dispatch OK) . Else, status remains 03.

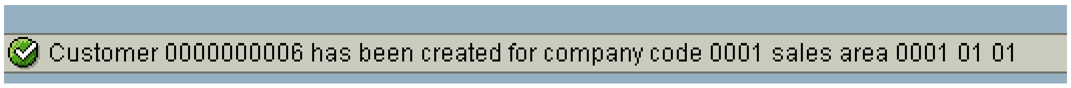
The selection-screen has two parameters :

IDoc creation date (from)
IDoc per commit work: Specifies number of IDoc to be checked before a commit is performed. Users are advised to stick with default values.

Example :

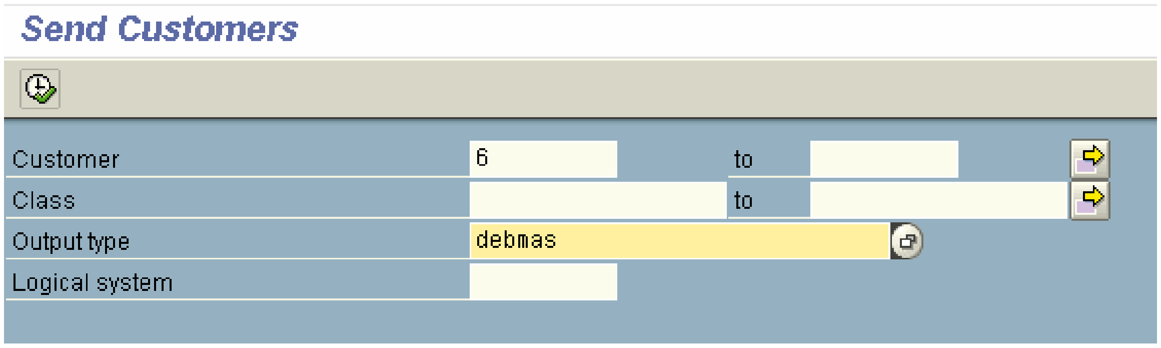
System PwCSAP , client 100 sends customer IDoc(message type DEBMAS) to system : Shatadru, client 200. So, we shall follow the steps defined below:-

1) Create customer master in PwCSAP, 100 using transaction code XD01.



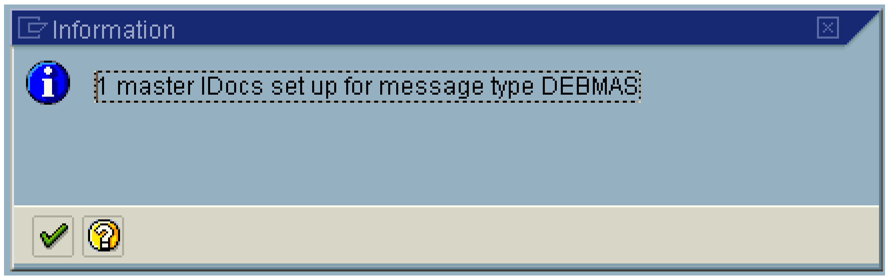
Customer 0000000006 has been created for company code 0001 sales area 0001 01 01

2) Use transaction BD12 to create communication and master IDoc in PwCSAP,100



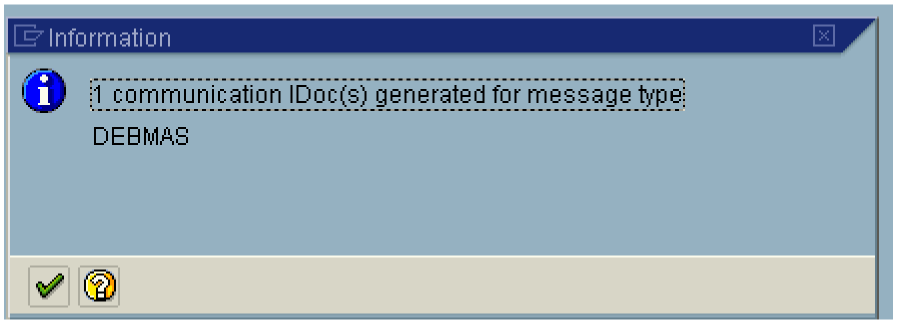
Send Customers

Customer	6	to	
Class		to	
Output type	debmas		
Logical system			



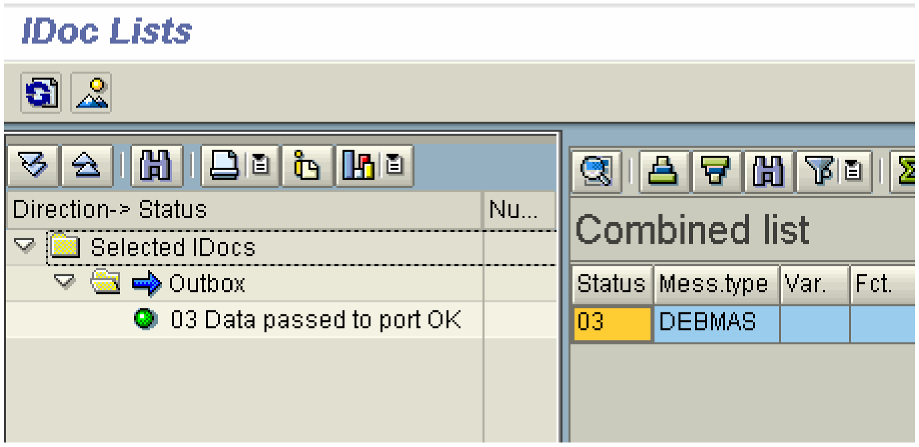
Information

1 master IDocs set up for message type DEBMAS



1 communication IDoc(s) generated for message type DEBMAS

3) See the status of the IDoc from transaction: WE05.

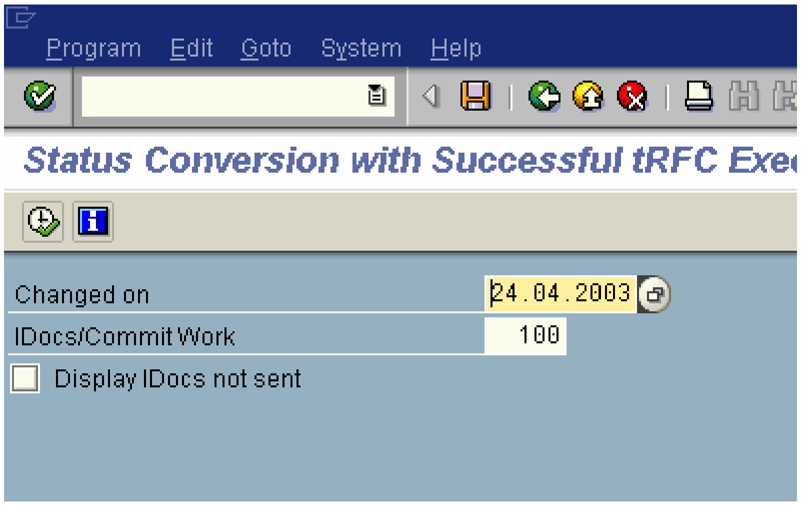


Direction-> Status	Nu...
Selected IDocs	
Outbox	
03 Data passed to port OK	

Combined list

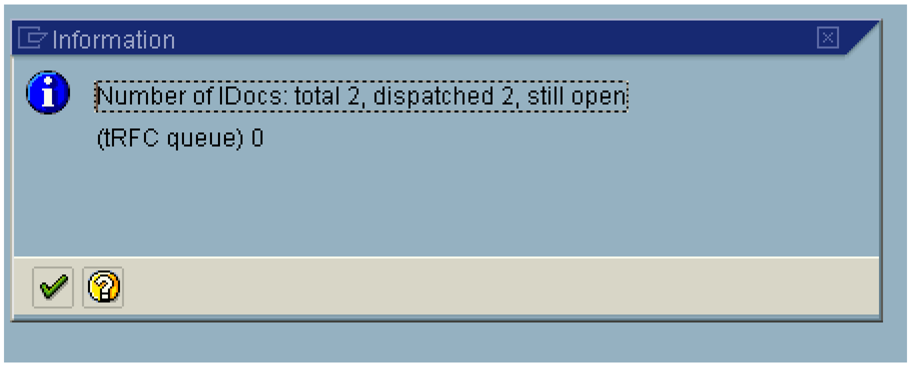
Status	Mess.type	Var.	Fct.
03	DEBMAS		

4) Execute the transaction: BD75.



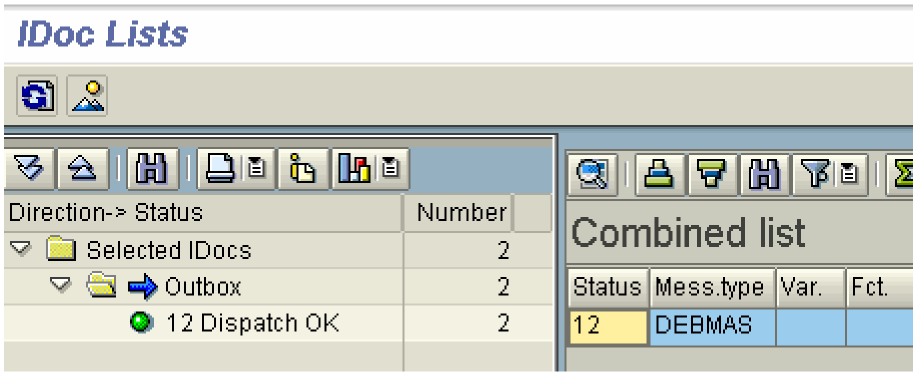
Status Conversion with Successful tRFC Exe

Changed on	24.04.2003
IDocs/Commit Work	100
Display IDocs not sent	☐


Information

Number of IDocs: total 2, dispatched 2, still open (tRFC queue) 0

5) Execute transaction WE05 to see the status


IDoc Lists

Direction-> Status	Number
Selected IDocs	2
Outbox	2
12 Dispatch OK	2

Combined list

Status	Mess.type	Var.	Fct.
12	DEBMAS		

5.2. Setting up Audit Reporting

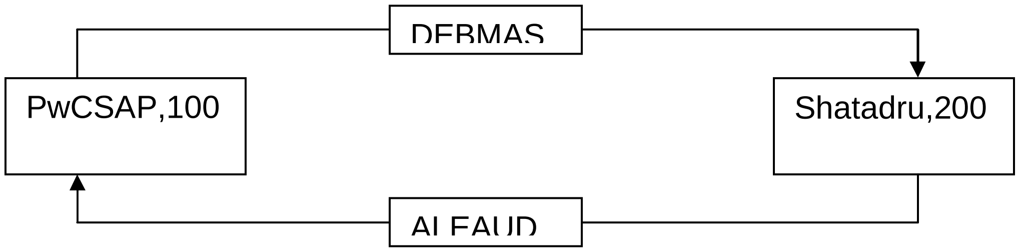
After an IDoc is dispatched to a destination system, the sender does not know the state of the process in the destination system. The system however can be configured for cross system reporting. One need to model the ALEAUD message between two systems.

For this message type, the sender is the receiver of the previous message and receiver is the sender of previous message.

There are two reports which helps in cross-system reporting:-

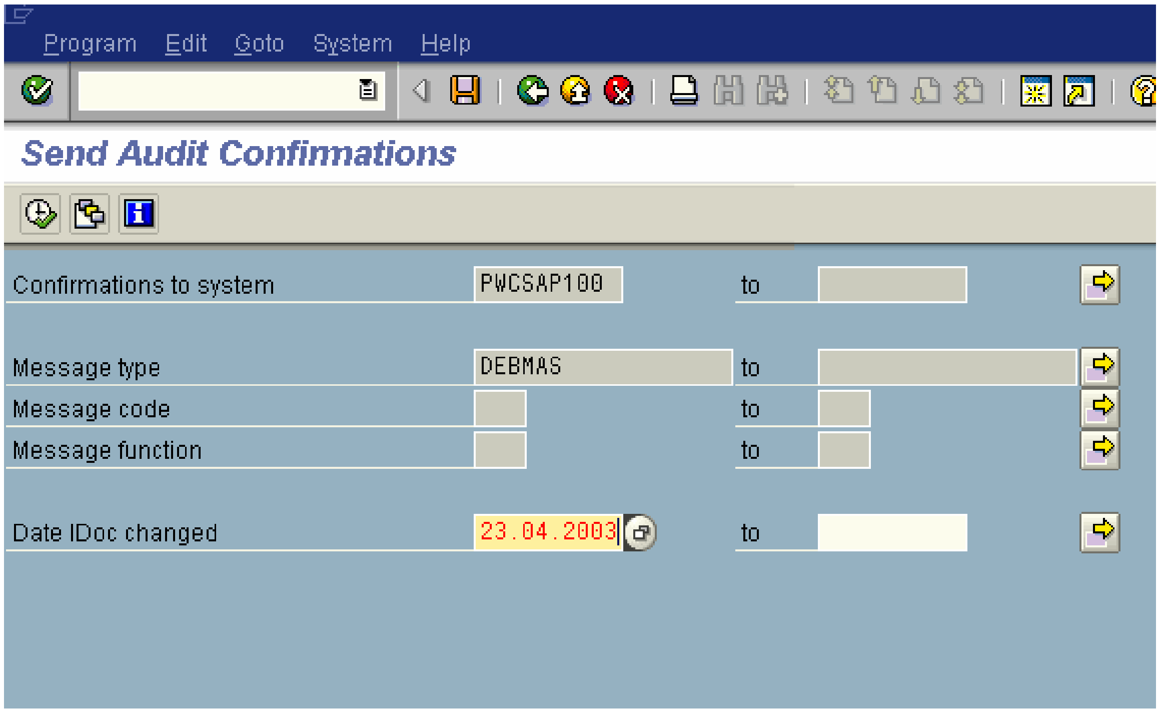
RBDSTATE(BDM8) : Is run periodically on destination system. It reports the status of incoming IDoc to sending system, using ALEAUD message and ALEAUD01 IDoc
.
RBDAUD01(BDM7) : Executed on sending system. Analyzes the audit log and displays the output as a report.

Example:-



Consider the customer master information transfer from PwCSAP,100 to Shatadru,200. Now, we will execute program RBDSTATE(tcode: BDM8) in Shatadru,200(receiver of message : DEBMAS)

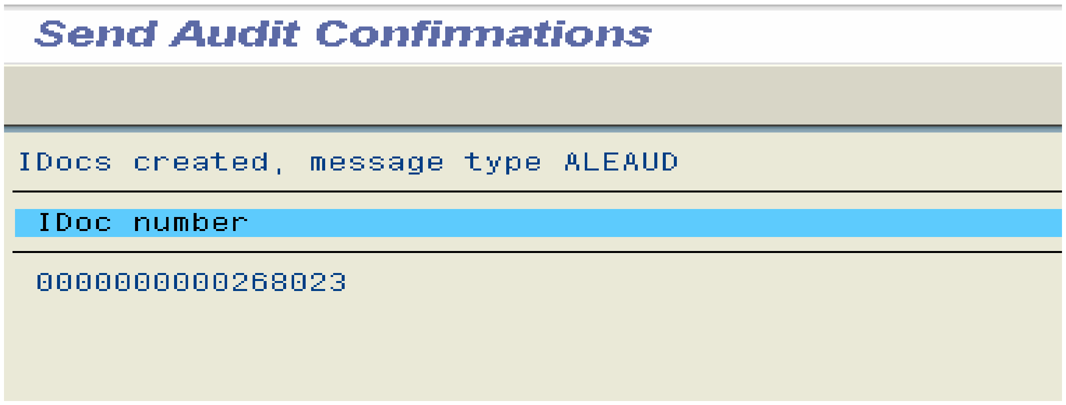
Selection-screen



Send Audit Confirmations

Confirmations to system	PWCSAP100	to	
Message type	DEBMAS	to	
Message code		to	
Message function		to	
Date IDoc changed	23.04.2003	to	

Execute the program



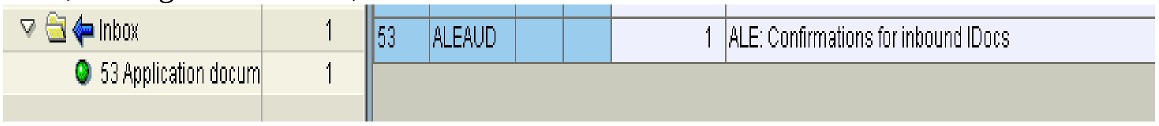
Send Audit Confirmations

IDocs created, message type ALEAUD

IDoc number
0000000000268023

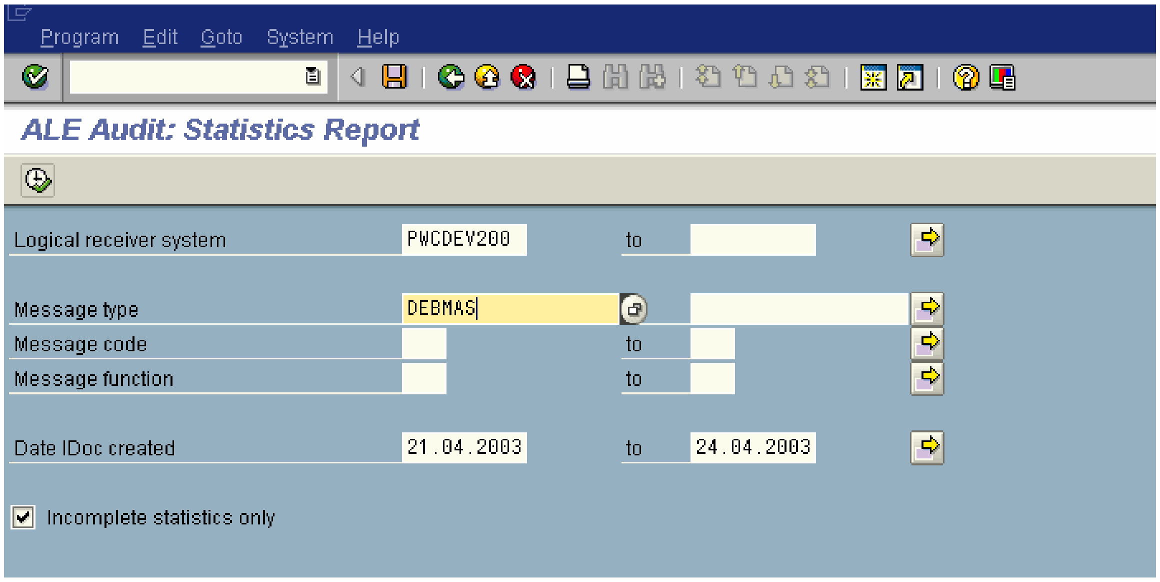
Shows the IDoc number of the IDoc ALEAUD01 for message type ALEAUD.

Now, let us go to PwCSAP, 100 and execute WE05.



Inbox	1	53	ALEAUD			1	ALE: Confirmations for inbound IDocs
53 Application docum	1						

It shows the arrival of the IDoc. Now, we will execute program RBDAUD01(tcode: BDM7) in PwCSAP,100(sender of DEBMAS to Shatadru,200)



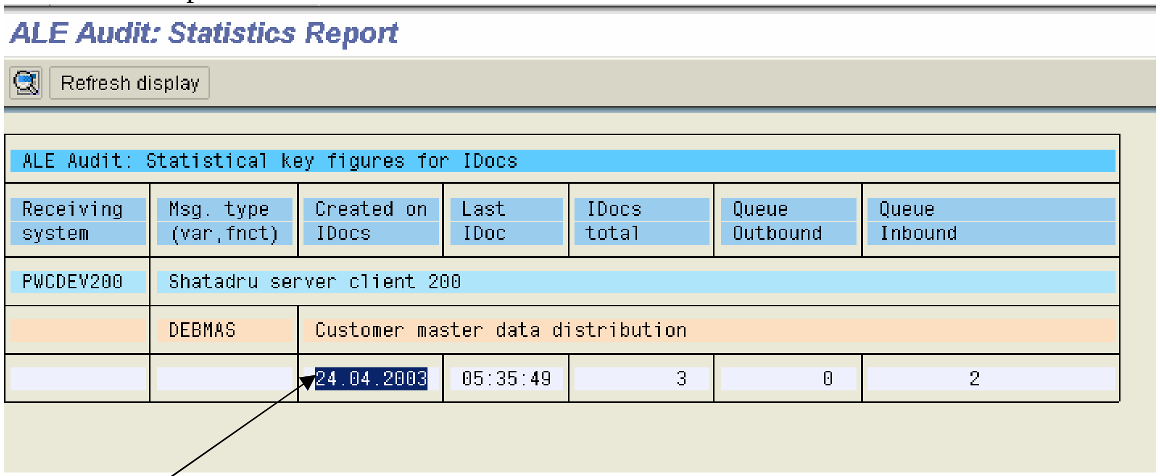
ALE Audit: Statistics Report

Logical receiver system	PWCDEV200	to	
Message type	DEBMAS		
Message code		to	
Message function		to	
Date IDoc created	21.04.2003	to	24.04.2003

[✓] Incomplete statistics only

Execute the program.

The initial output is as follows:-



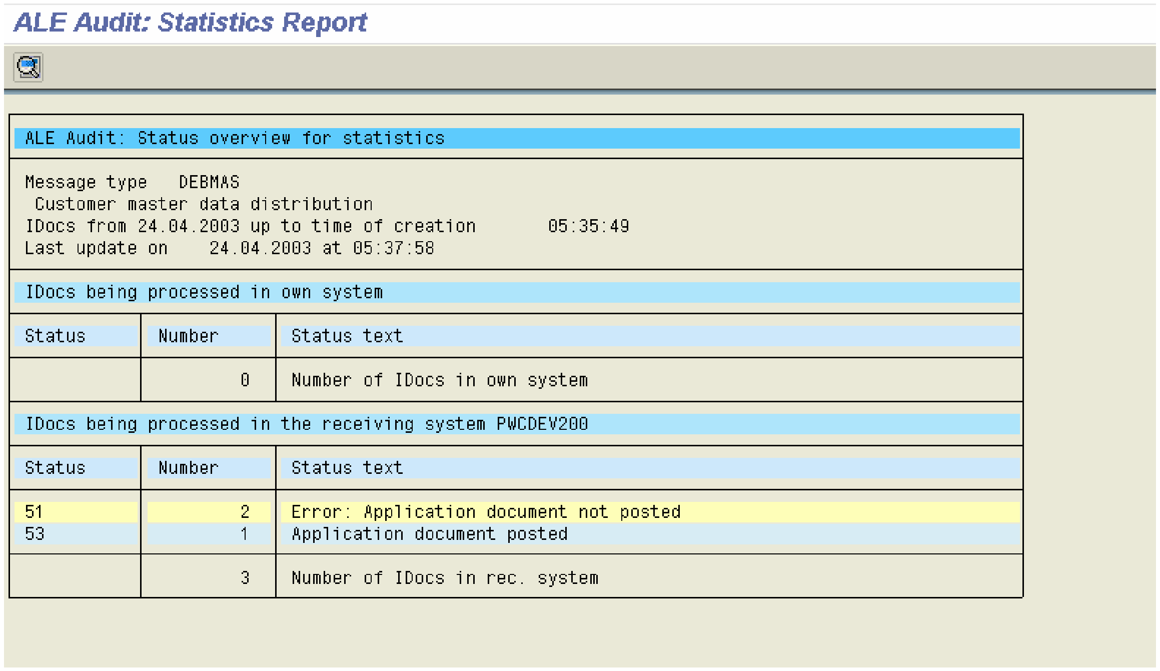
ALE Audit: Statistics Report

ALE Audit: Statistical key figures for IDocs

Receiving system	Msg. type (var,fnct)	Created on IDocs	Last IDoc	IDocs total	Queue Outbound	Queue Inbound
PWCDEV200	Shatadru server client 200					
	DEBMAS	Customer master data distribution				
		24.04.2003	05:35:49	3	0	2

Double click here.

It shows a detail of how many IDoc of the specified message type are sent; and what is the current status in the receiver system.


ALE Audit: Statistics Report

```
ALE Audit: Status overview for statistics

Message type   DEBMAS
 Customer master data distribution
IDocs from 24.04.2003 up to time of creation       05:35:49
Last update on     24.04.2003 at 05:37:58
```

IDocs being processed in own system

Status	Number	Status text
	0	Number of IDocs in own system

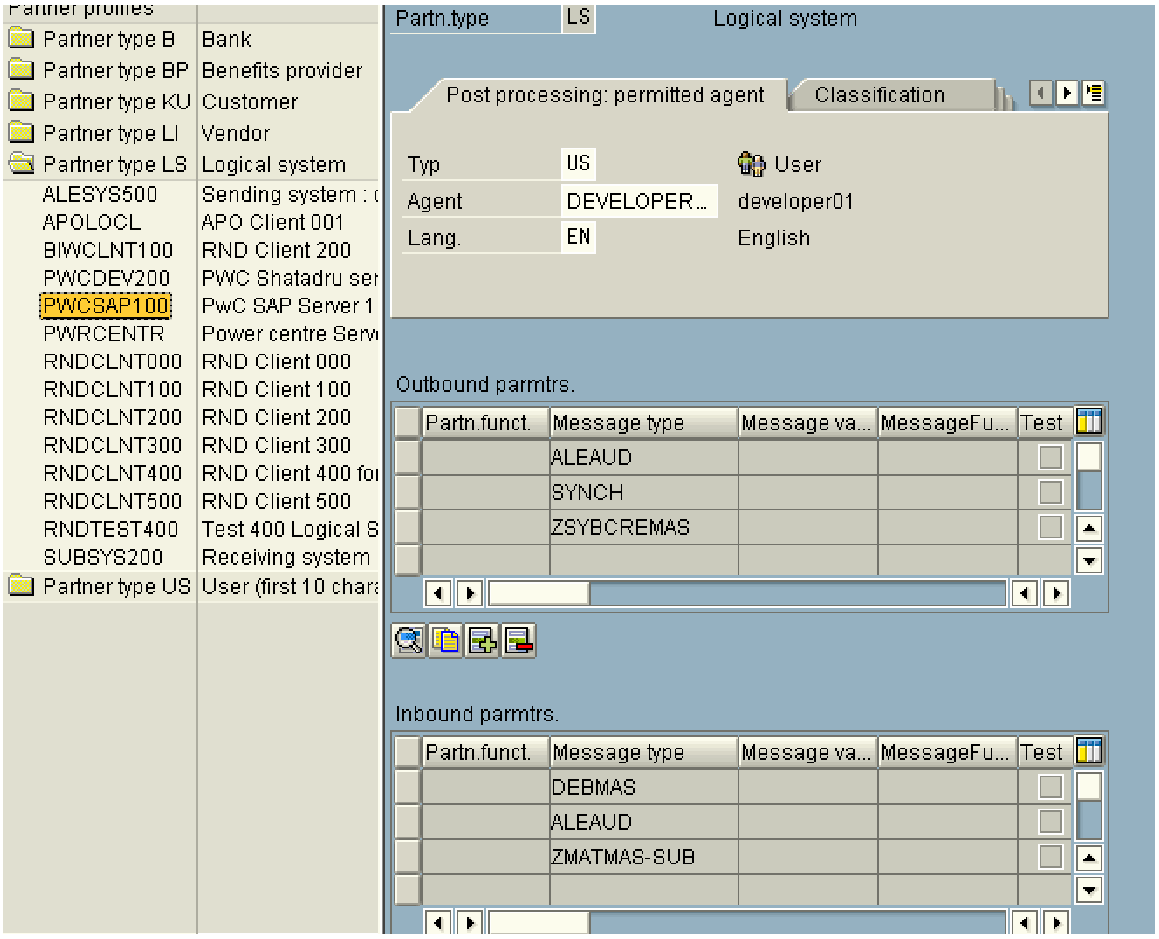
IDocs being processed in the receiving system PWCDEV200

Status	Number	Status text
51	2	Error: Application document not posted
53	1	Application document posted
	3	Number of IDocs in rec. system

5.3. Manually Process IDoc in Receiving System

In this context, for example, PwCSAP,100 sends Idocs for customer master to Shatadru,100. If no error occurs, the IDoc is automatically posted in Shatadru. Actually, automatic or manual processing depends on the partner profile in Shatadru for that message type when the sender is PwCSAP,100

This can be viewed by transaction WE20.



Partner profiles	
Partner type B	Bank
Partner type BP	Benefits provider
Partner type KU	Customer
Partner type LI	Vendor
Partner type LS	Logical system
ALESYS500	Sending system : (
APOLOCL	APO Client 001
BIWCLNT100	RND Client 200
PWCDEV200	PWC Shatadru ser
PWCSAP100	PwC SAP Server 1
PWRCENTR	Power centre Serv
RNDCLNT000	RND Client 000
RNDCLNT100	RND Client 100
RNDCLNT200	RND Client 200
RNDCLNT300	RND Client 300
RNDCLNT400	RND Client 400 for
RNDCLNT500	RND Client 500
RNDTEST400	Test 400 Logical S
SUBSYS200	Receiving system
Partner type US	User (first 10 chara

Partn.type LS Logical system

Post processing: permitted agent | Classification

Typ	US	User
Agent	DEVELOPER...	developer01
Lang.	EN	English

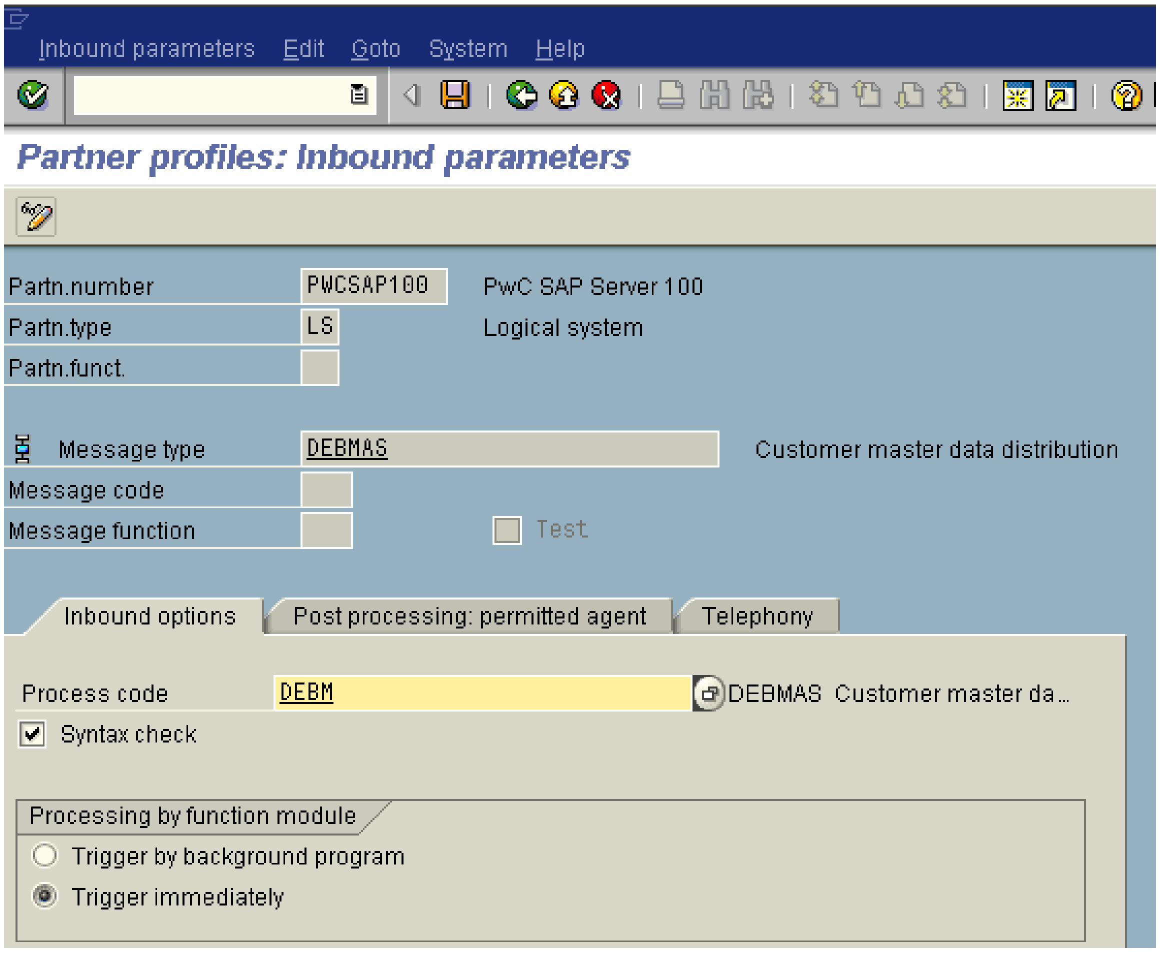
Outbound parmtrs.

Partn.funct.	Message type	Message va...	MessageFu...	Test
	ALEAUD			☐
	SYNCH			☐
	ZSYBCREMAS			☐

Inbound parmtrs.

Partn.funct.	Message type	Message va...	MessageFu...	Test
	DEBMAS			☐
	ALEAUD			☐
	ZMATMAS-SUB			☐

Look at the partner profile maintained for PWCSAP100 in Shatadru. You see DEBMAS as the message type in the table control for Inbound parameters. Double click on that line.



Partner profiles: Inbound parameters

Partn.number	PWCSAP100	PwC SAP Server 100
Partn.type	LS	Logical system
Partn.funct.		
Message type	DEBMAS	Customer master data distribution
Message code		
Message function		☐ Test

Inbound options | Post processing: permitted agent | Telephony

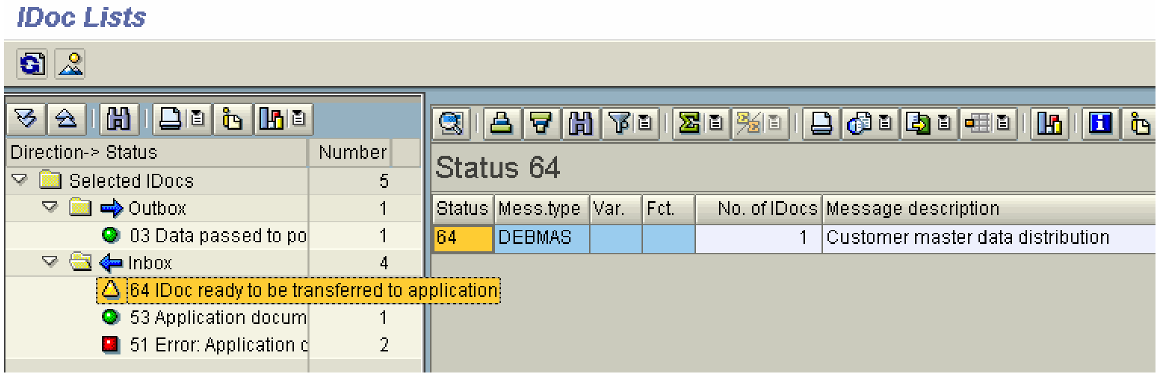
Process code DEBM DEBMAS Customer master da...

☑ Syntax check

Processing by function module
- ○ Trigger by background program
- ● Trigger immediately

You are navigated to the next screen. Notice that the radiobutton for triggering the inbound process automatically is checked. Now, uncheck it and check the radiobutton: Trigger by background program.

Now, in PwCSAP,100, create/change a customer and create the master IDoc by BD12. Now, login to Shatadru,200 and see the status of the IDoc by WE05


IDoc Lists

Direction-> Status	Number
Selected IDocs	5
Outbox	1
03 Data passed to po	1
Inbox	4
64 IDoc ready to be transferred to application	
53 Application docum	1
51 Error: Application c	2

Status 64

Status	Mess.type	Var.	Fct.	No. of IDocs	Message description
64	DEBMAS			1	Customer master data distribution

Now, note down the IDoc number with status=64(IDoc ready to be transferred to application).

Now, go to transaction : BD87 in Shatadru,200 and select the IDoc in the selection-screen and execute the program. Select the node for the status 64 and click Process Pushbutton from application toolbar. The IDoc will be manually processed.

Then go to transaction WE05. You will see the status of the IDoc to be 53(Application Document posted).

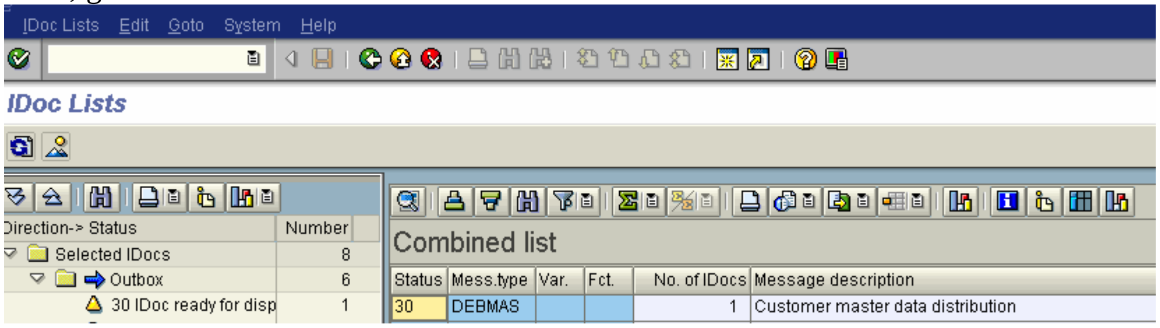
5.4. Collect IDoc and Transfer

While generating the partner profile by transaction BD82, one can set the mode:

- Send IDoc immediately
- Collect IDoc and transfer

Let us take the case of sending IDoc for message type DEBMAS(Customer master) from PwCSAP100 to Shatadru,200. Login into PwCDEV, 100 and go to transaction WE20. Select the line for DEBMAS in the table control for outbound parameters and double click on it. Then, in the Details screen, check the radiobutton: **Collect IDoc and transfer** for the message type and Save..

Then create/change customer master and create the IDoc by tcode : BD12 in PwCSAP,100.
Then, go to transaction WE05 and see the status of the IDoc created.



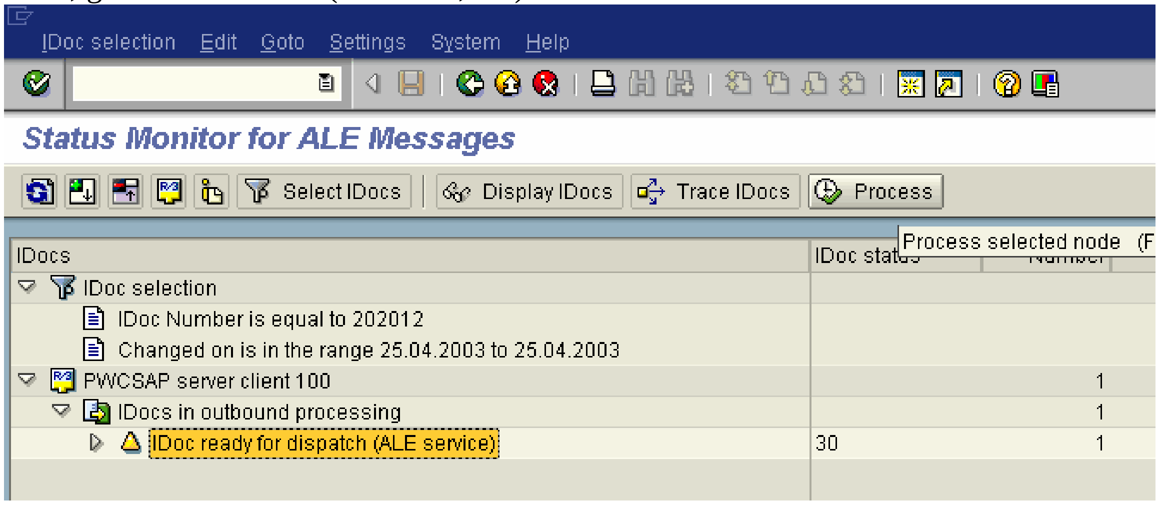
Direction-> Status	Number
Selected IDocs	8
Outbox	6
30 IDoc ready for disp	1

Combined list

Status	Mess.type	Var.	Fct.	No. of IDocs	Message description
30	DEBMAS			1	Customer master data distribution

So, now, the IDoc is ready for dispatch, but yet not dispatched.
Now, to dispatch the IDoc, access transaction WE14(program : RSEOUT00) in PwCSAP,100. Fill in the selection-screen and execute.It will show you appropriate message signifying that IDoc is dispatched.

Then, go to tcode: WE05(PwCSAP,100) and execute.


IDoc selection Edit Goto Settings System Help
Status Monitor for ALE Messages
Select IDocs
Display IDocs
Trace IDocs
Process
Process selected node (F

IDocs	IDoc status	Number
IDoc selection		
IDoc Number is equal to 202012		
Changed on is in the range 25.04.2003 to 25.04.2003		
PWCSAP server client 100		1
IDocs in outbound processing		1
IDoc ready for dispatch (ALE service)	30	1

It will show you that the IDoc is passed to port.

Note :

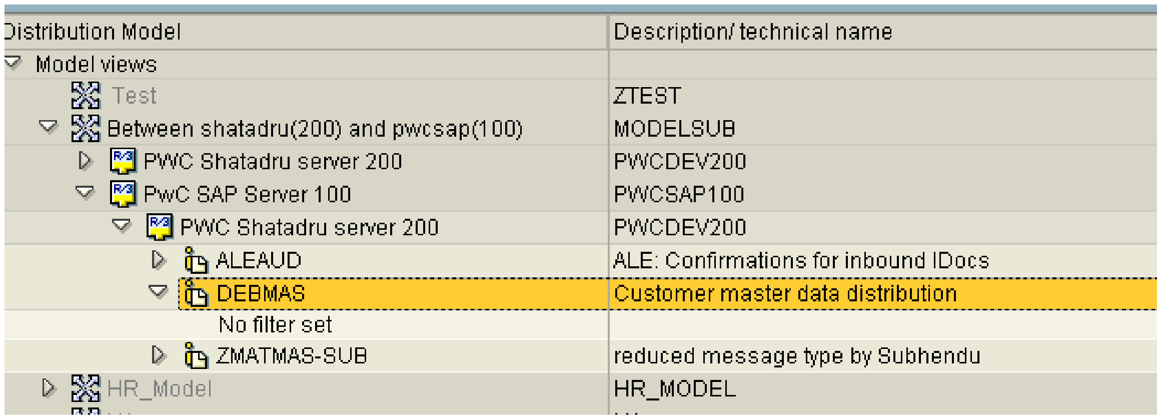
When the settings is done as : Send IDoc immediately, program RSEOUT00 is executed at once. Else, it has to be manually executed

5.5. Creating Filter Objects

Often you would prefer selective transfer of information in the form of IDoc from one system to another. Based on some specific values, you would prefer some specific recipient for the information.

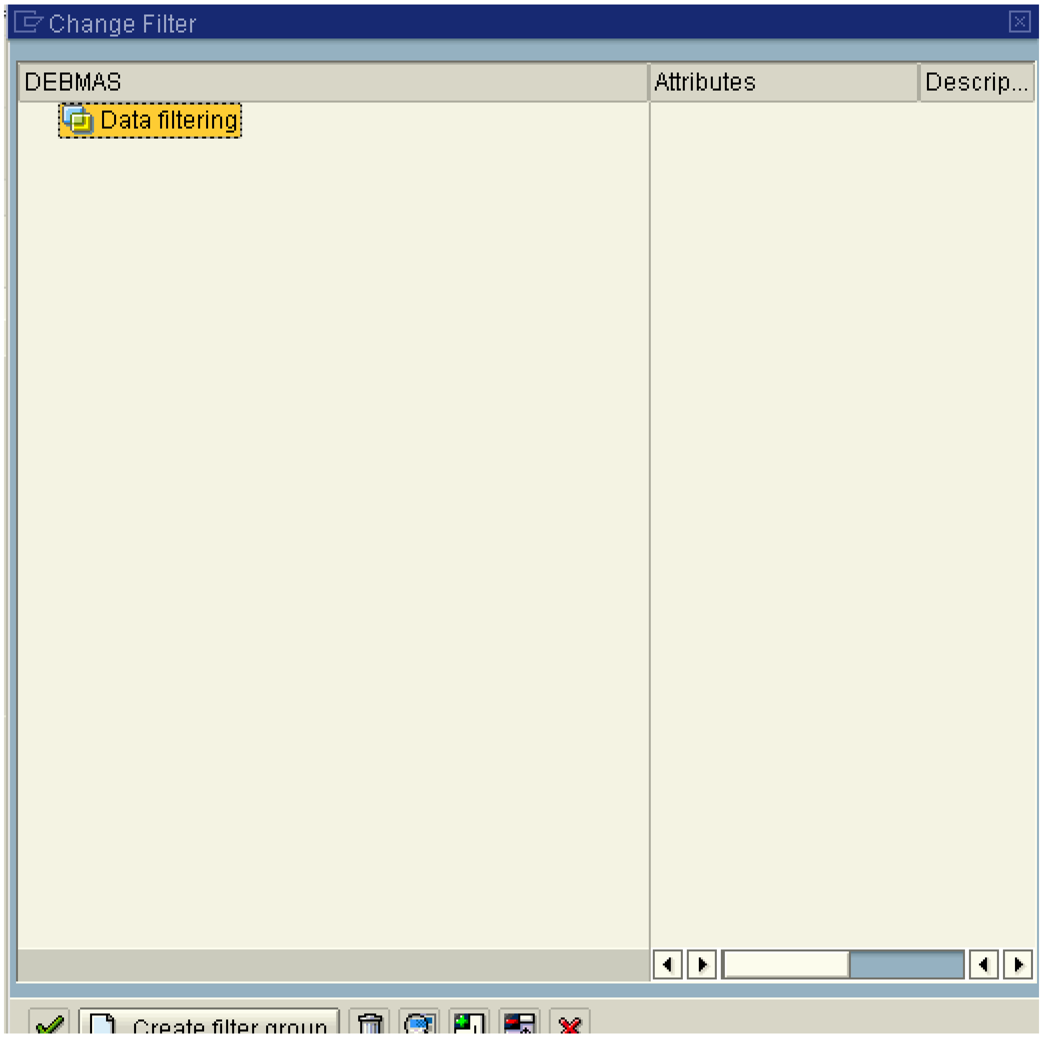
For example, system PwCSAP,100 creates customer master and sends it to system PWCDEV,200-Shatadru. We want that only data pertaining to company code = 0001 will come from PwCSAP,100 to Shatadru only. So, for that, we have to set a filter object for the message type DEBMAS in PwCDEV,200(because this was originally the system where CDM was created) using the object for country.

Go to tcode BD64 and dig down from the customer distribution model to ultimately select the line for message type DEBMAS exchanged between PwCSAP,100 and PwCDEV,200.



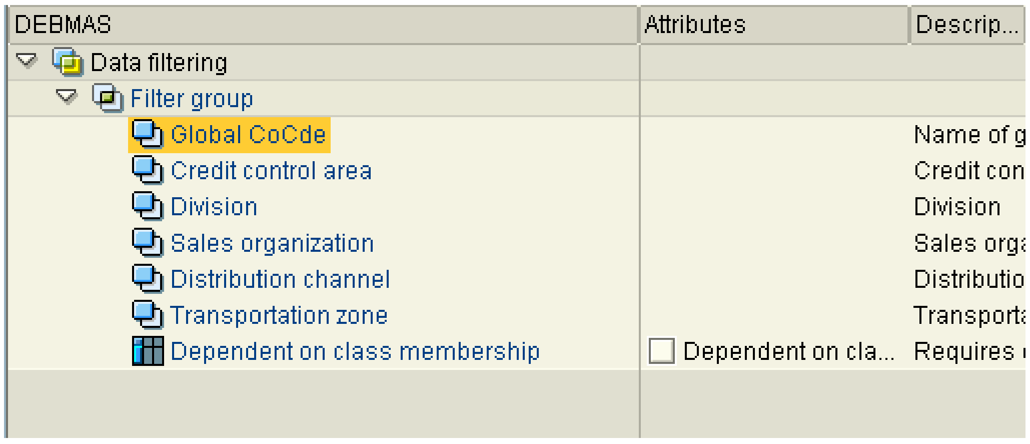
Distribution Model	Description/ technical name
Model views	
Test	ZTEST
Between shatadru(200) and pwcsap(100)	MODELSUB
PWC Shatadru server 200	PWCDEV200
PwC SAP Server 100	PWCSAP100
PWC Shatadru server 200	PWCDEV200
ALEAUD	ALE: Confirmations for inbound IDocs
DEBMAS	Customer master data distribution
No filter set	
ZMATMAS-SUB	reduced message type by Subhendu
HR_Model	HR_MODEL

Double click on it. A screen will appear:-



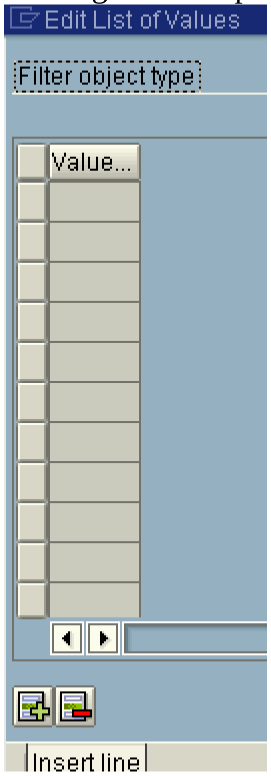
Select the node : Data Filtering and press **Create Filter Group** pushbutton from the bottom of the dialog window..

Then expend the node : Data filtering when U will view the following:-



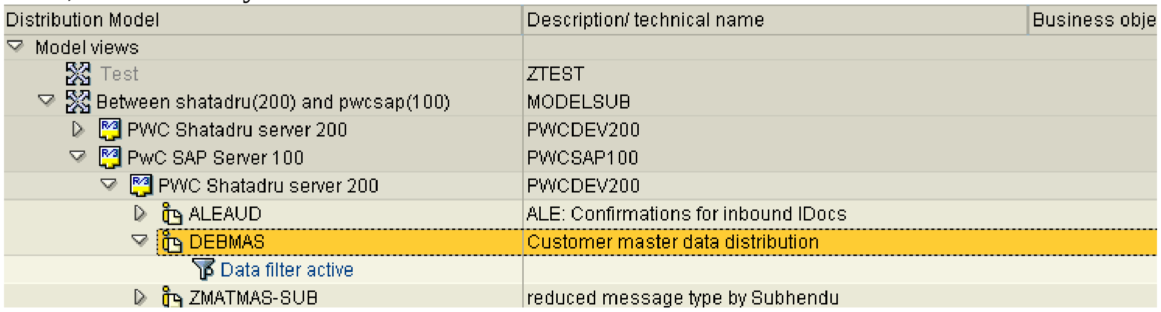
DEBMAS	Attributes	Descrip...
Data filtering		
Filter group		
Global CoCde		Name of g
Credit control area		Credit con
Division		Division
Sales organization		Sales orga
Distribution channel		Distributio
Transportation zone		Transporta
Dependent on class membership	☐ Dependent on cla...	Requires

This shows that you can create filter object on one/more of the above fields. Double click on the global company code node.



Press **Insert Line** pushbutton from bottom.Write 0001 in the value column and press Enter. Then Save the entry by pressing **Save** pushbutton from Application toolbar.

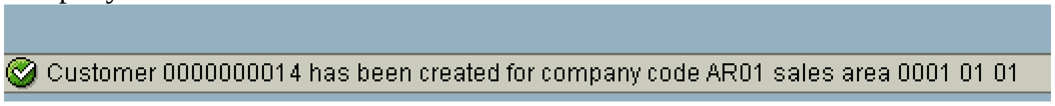
Now, it will show you that Data filter is active.



Distribution Model	Description/ technical name	Business obje
Model views		
Test	ZTEST	
Between shatadru(200) and pwcsap(100)	MODELSUB	
PWC Shatadru server 200	PWCDEV200	
PwC SAP Server 100	PWCSAP100	
PWC Shatadru server 200	PWCDEV200	
ALEAUD	ALE: Confirmations for inbound IDocs	
DEBMAS	Customer master data distribution	
Data filter active		
ZMATMAS-SUB	reduced message type by Subhendu	

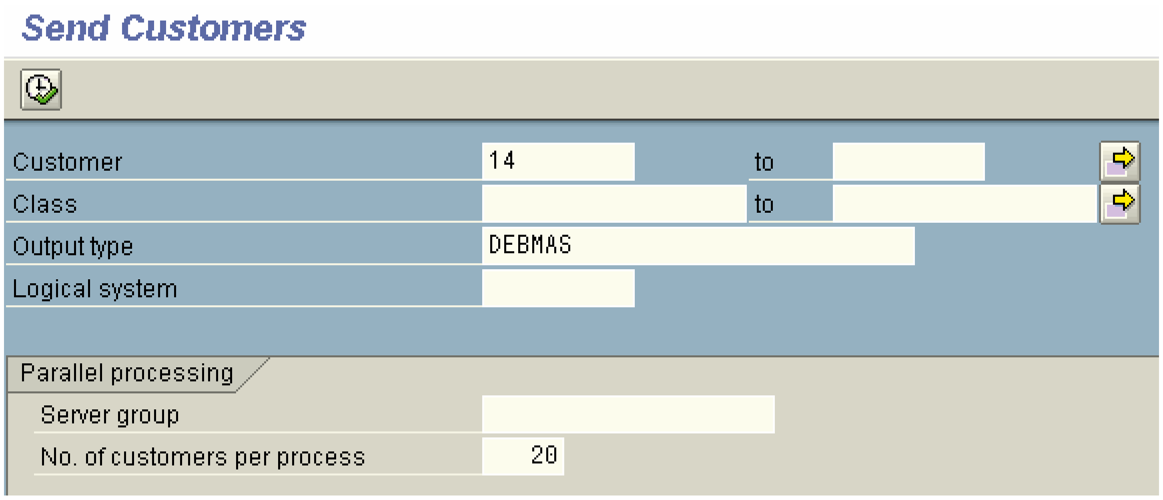
Then, distribute the customer distribution model so that it affects system PwCSAP,100. You will see the change affecting the view by tcode: BD64 in system : PWCSAP,100.

Now, logon to PwCSAP,100 and create a customer with company code having global company code other than 0001.



Customer 0000000014 has been created for company code AR01 sales area 0001 01 01

Then, try to create the IDoc by BD12. IT will create Master IDoc but no communication IDoc.



Send Customers

Customer	14	to	
Class		to	
Output type	DEBMAS		
Logical system			

Parallel processing

Server group	
No. of customers per process	20

So, Idocs with global company code = 0001 will be sent as IDoc to PwCDEV,200 only.

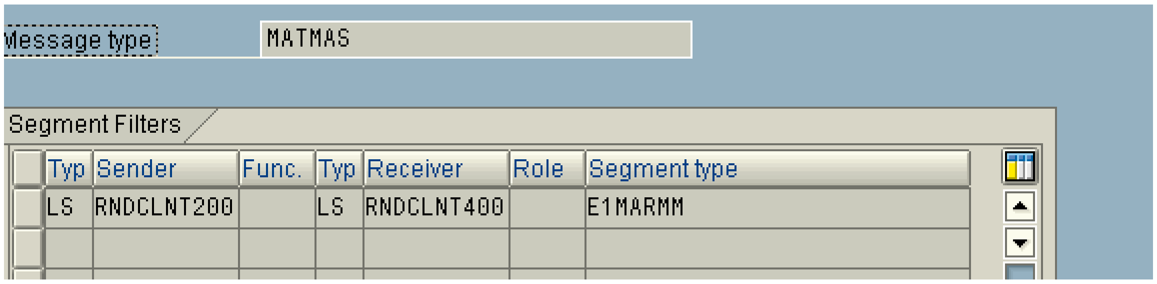
5.6. Segment Filtering

An IDoc consists of more than one segments and each segment consists of one/more than one fields. It may be possible that while sending an IDoc from one system to some specific system, you do not want to send the information on one segment to that system.

For example, you create material master in one SAP system and send that material master to another system. Material master valuation data (Valuation class, Valuation category etc) is maintained in the receiver system. So, there is no need to send the segment containing material master valuation data from the sender system. So, you need to specify that while sending information on material master message (MATMAS) from system x to system y, you do not need the material master valuation data segment(E1MBEWM) information.

For that, you need to access transaction BD56, specify the message type at the initial screen and maintain a table field entry, where you specify the :-

Partner type of sender
Sender
Partner type of receiver
Receiver
Segment type



Message type: MATMAS

Segment Filters

Typ	Sender	Func.	Typ	Receiver	Role	Segment type
LS	RNDCLNT200		LS	RNDCLNT400		E1MARMM

5.7. Creating Reduced message type

IDoc are instances of message types. It may be possible that you do not all the information of a message type to send to some specific receiver system. You want to omit one/more than one fields which are irrelevant to receiver system/ maintained by receiver system independently. In such a case, you create a reduced message type.

For example, let us assume that Shatadru,200 sends vendor master information to PwcSAP,100. But, you do not want to send the vendor's house address to receiver system. So, you create a custom message type(ZSYBCREMAS) copying vendor master message type(CREMAS) where you do not activate the field for vendor's address field. In brief, the steps will be as follows:-

1. Create reduced message type in sender system by tcode BD53
2. In the customer distribution model(BD64) between the two systems, add the message type : ZSYBCREMAS.
3. Generate the partner profile in Shatadru,200 by tcode : BD82
4. In Shatadru, from WE20, check the partner profile to see whether the appropriate reduced message type is specified.
5. From Shatadru, distribute the CDM(BD64)
6. Generate the Partner profile in PwCSAP,100 by BD82
7. In Shatadru, create vendor and distribute by BD14.
8. Observer IDoc status in PwCSAP,100 by WE05
9. Check vendor in PwCSAP,100 and see whether the desired unwanted information is successful or not

5.8. Change Pointers Technique

Change pointers technology helps to create IDoc when any field for which change pointer is set is changed.

1. Make the change pointer globally active by tcode : BD61
2. Activate change pointer for the message type(say, MATMAS) by BD50
3. Add fields for which change pointers are written , using transaction BD52(say, for MM master, object : MATERIAL, table : MARA, field : BRGEW)
4. Change the material by MM02
5. Check entry in BDCP table
6. Execute BD21 that will create IDoc from change pointers
7. Check the IDoc by WE05 and verify its existence in the receiver system.

5.9. Reprocessing IDocs not posted due to errors

IDocs may not leave the source system successfully or they may not be posted into database in the receiver system due to errors.

Such IDocs can be seen from transaction WE05 where the erroneous IDocs are shown in Red signal. The reason for unsuccessful posting or unsuccessful transmission to the destination system can be found from the error message.

In this case, the developer has to remove the reason for error. Then, he has to manually process those IDocs again.

- In source system, use transaction BD73 for reprocessing of Outbound IDocs (IDocs leaving the source system) after removing the reason for failure (of the IDOC posting initially).
- In destination system, for inbound posting, use transaction BD84 for reprocessing of Inbound IDocs after removing the reason for failure.

5.10. Processing IDocs waiting in the queue

Sometimes, when a huge number of IDocs are sent from one system to another, they remain in queue as not enough work processes are available. To process them, one can use transaction BD20.

6. Developing and Transmitting New IDoc

Shatadru, client 777 will convey to client 555 information on educational qualification. So, for this scenario, sender is : Shatadru 777 and receiver is Shatadru 555.The steps are outlined below:-

6.1. Prepare data container in both sender and receiver

Following tables should exist in both the sender and receiver:-

Table : ZEMP_MAST

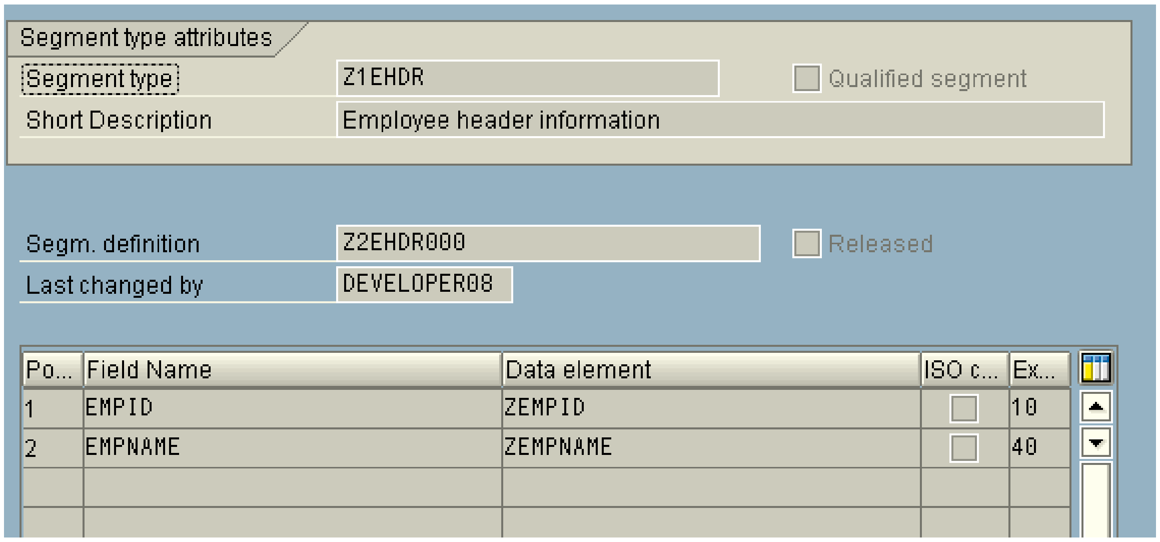
Field	**Data element**
Mandt	Mandt
Empid	Zempid
Empname	zempname

Table : ZEMP_QUAL

Field	**Data element**
mandt	mandt
empid	zempid
pyear	zyear
qual	zqual

6.2. Prepare Segments(WE31)

Segment Z1EHDR (with fields empid and empname) should be there in both the systems as follows:-

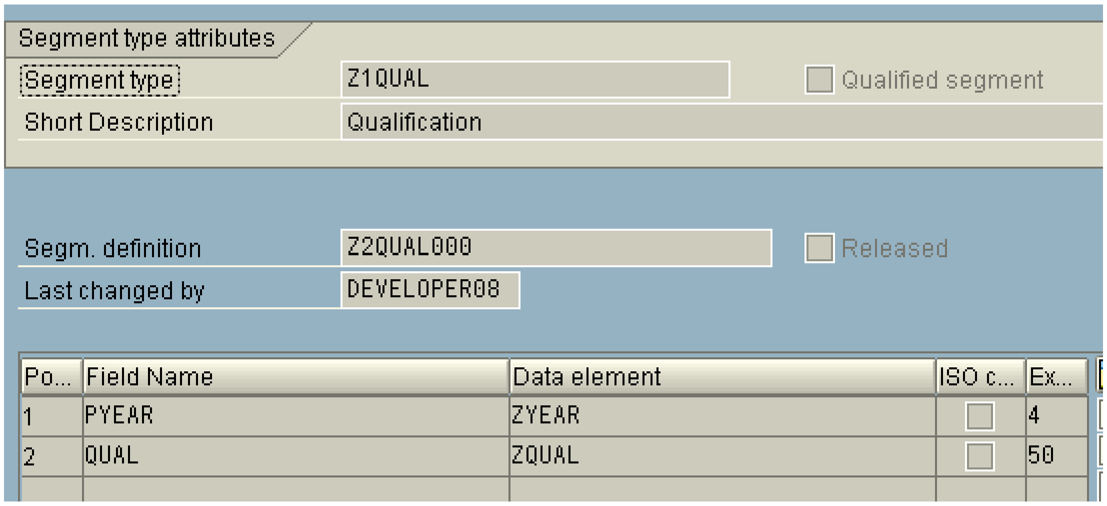


Segment type attributes

Segment type	Z1EHDR
Short Description	Employee header information
Qualified segment	☐
Segm. definition	Z2EHDR000
Released	☐
Last changed by	DEVELOPER08

Po...	Field Name	Data element	ISO c...	Ex...
1	EMPID	ZEMPID	☐	10
2	EMPNAME	ZEMPNAME	☐	40

Similarly, construct another segment :- Z1QUAL in both the systems as follows:-

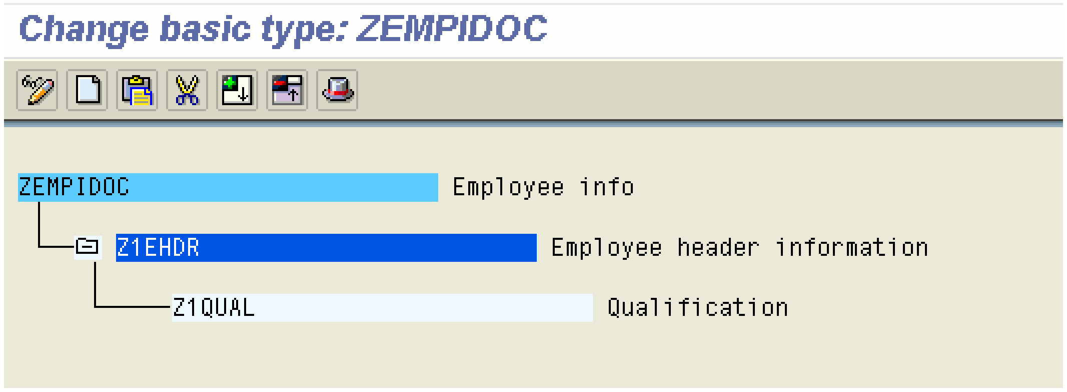


Segment type attributes

Segment type	Z1QUAL
Short Description	Qualification
Qualified segment	☐
Segm. definition	Z2QUAL000
Released	☐
Last changed by	DEVELOPER08

Po...	Field Name	Data element	ISO c...	Ex...
1	PYEAR	ZYEAR	☐	4
2	QUAL	ZQUAL	☐	50

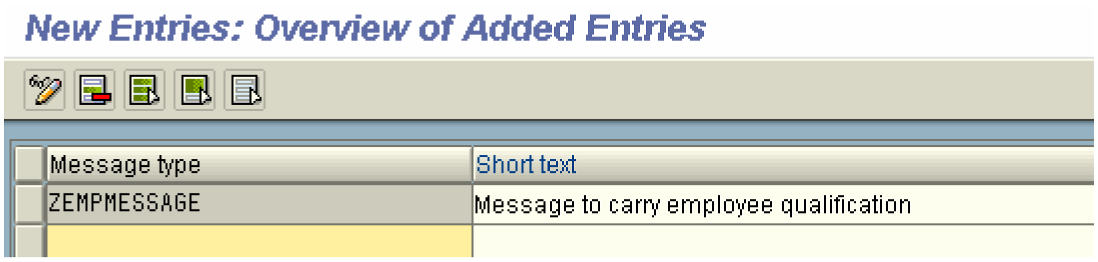
6.3. Prepare IDocs with the segments in both systems (WE30)



Change basic type: ZEMPIDOC

```
ZEMPIDOC        Employee info
  Z1EHDR        Employee header information
    Z1QUAL      Qualification
```

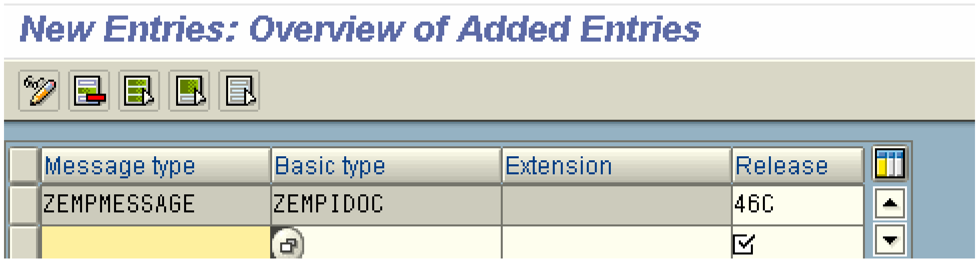
6.4. Create new message type in both the systems (WE81)



New Entries: Overview of Added Entries

Message type	Short text
ZEMPMESSAGE	Message to carry employee qualification

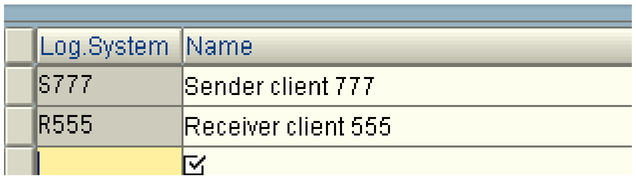
6.5. Link new message type with IDoc type in both systems (WE82)



New Entries: Overview of Added Entries

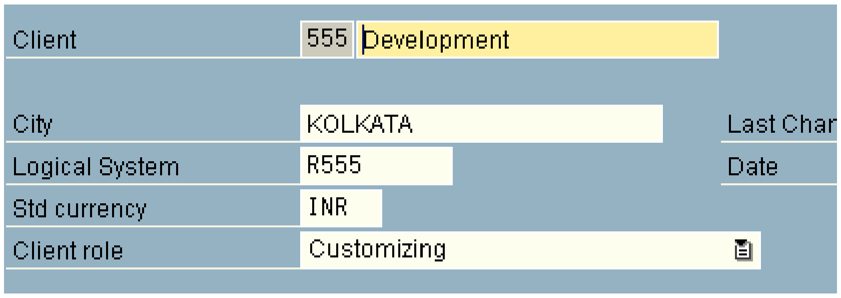
Message type	Basic type	Extension	Release
ZEMPMESSAGE	ZEMPIDOC		46C

6.6. Maintain two logical systems, one for sender-another for receiver in both the systems (BD54)

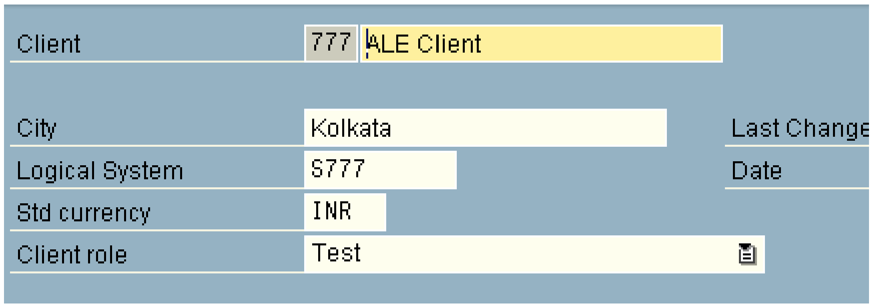


Log.System	Name
S777	Sender client 777
R555	Receiver client 555

6.7. Assign Logical System for Receiver to appropriate client in Receiver system and assign logical system for sender to appropriate client in sender system (SCC4)

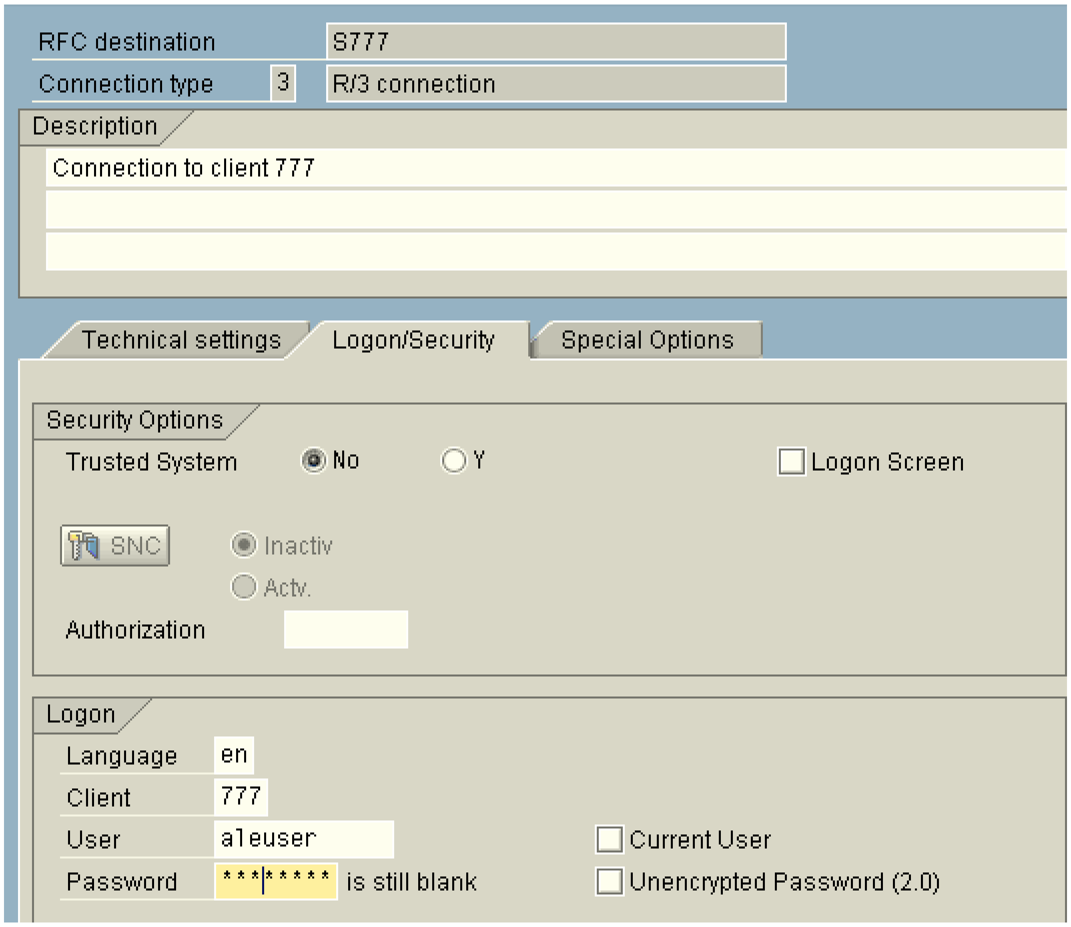


Client	555 Development	
City	KOLKATA	Last Char
Logical System	R555	Date
Std currency	INR	
Client role	Customizing	



Client	777 ALE Client	
City	Kolkata	Last Change
Logical System	S777	Date
Std currency	INR	
Client role	Test	

6.8. Create RFC Destination for Receiver in sender system and for sender in receiver system(SM59)



RFC destination	S777	
Connection type	3	R/3 connection

Description

Connection to client 777

Technical settings | Logon/Security | Special Options

Security Options

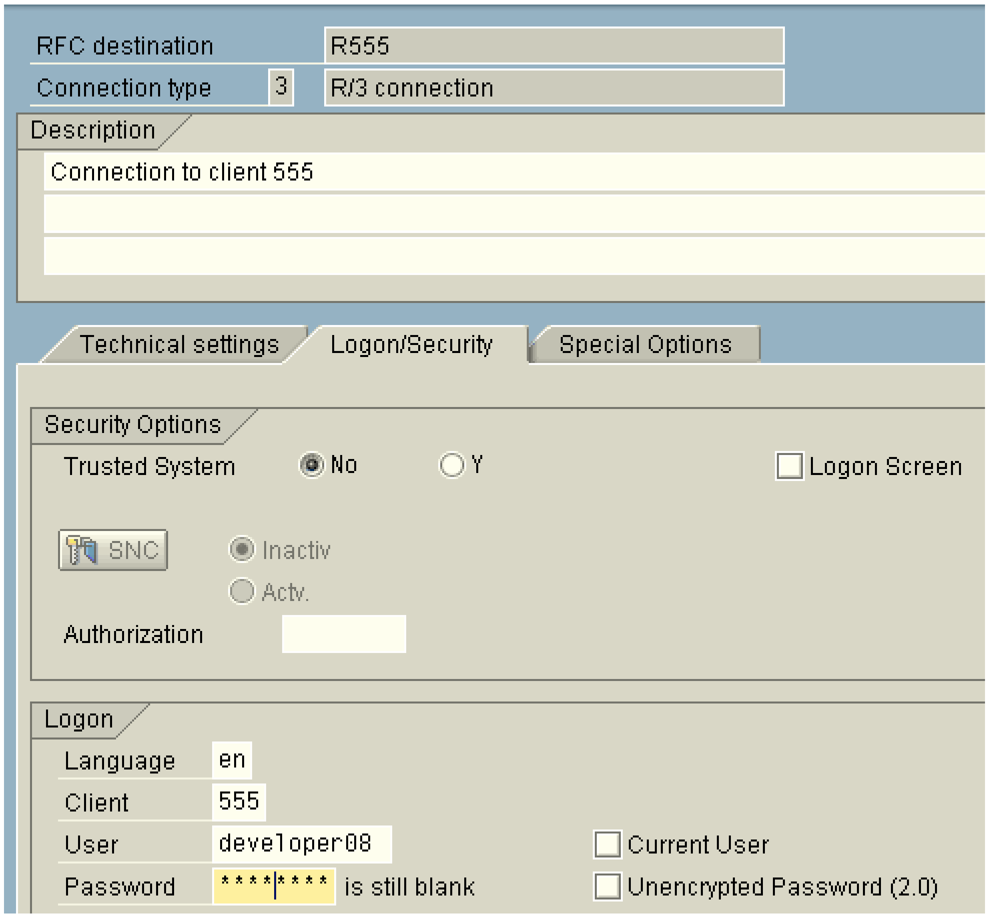
Trusted System [✓ No] Y — Logon Screen ☐

SNC — [✓ Inactiv] Actv.

Authorization

Logon

Language	en	
Client	777	
User	aleuser	☐ Current User
Password	******** is still blank	☐ Unencrypted Password (2.0)



RFC destination		R555
Connection type	3	R/3 connection

Description

Connection to client 555

Technical settings | Logon/Security | Special Options

Security Options

Trusted System [✓ No] Y ☐ Logon Screen

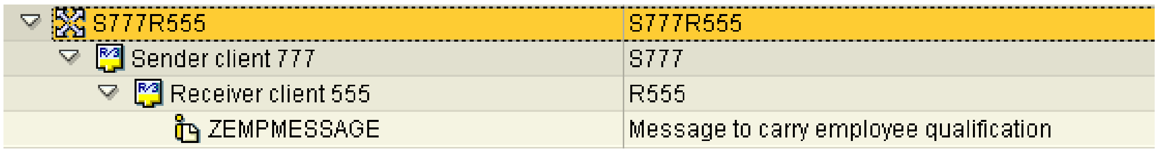
SNC [✓ Inactiv] Actv.

Authorization

Logon

Language	en	
Client	555	
User	developer08	☐ Current User
Password	******** is still blank	☐ Unencrypted Password (2.0)

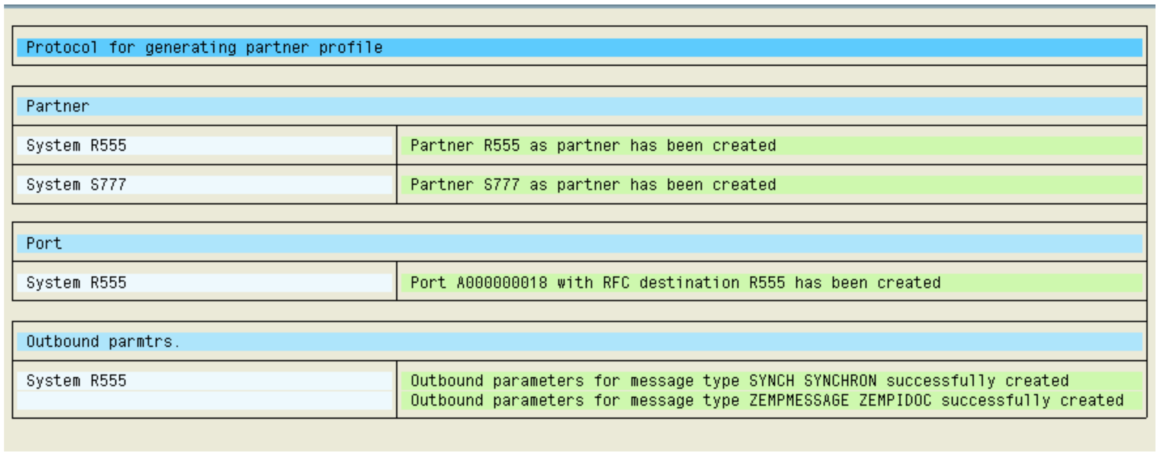
6.9. Create Customer Distribution Model for the Message type in sender systems(BD64)



S777R555	S777R555
Sender client 777	S777
Receiver client 555	R555
ZEMPMESSAGE	Message to carry employee qualification

6.10. Generate the partner profile in the sender system(BD82)

Execute the program using the name of the technical system in the CDM.
Partners for both senders and receivers,ports and partner profiles are automatically created and is informed to the user by a list output:-



```
Protocol for generating partner profile

Partner
System R555        Partner R555 as partner has been created
System S777        Partner S777 as partner has been created

Port
System R555        Port A000000018 with RFC destination R555 has been created

Outbound parmtrs.
System R555        Outbound parameters for message type SYNCH SYNCHRON successfully created
                   Outbound parameters for message type ZEMPMESSAGE ZEMPIDOC successfully created
```

6.11. Distribute the CDM from sender system to the reciver system(BD64)

From BD64, select the model and follow the menupath :- Edit→Model View→Distribute. Select the reciver system and press enter.

Now, this CDM will also be created in the receiver system too.

6.12. Create the outbound program in sender system to populate and distribute the Idocs(SE38)

```
*&---------------------------------------------------------------------*
*& Report  YSUBCLASS_DISTRIBUTE                                        *
*&                                                                     *
*&---------------------------------------------------------------------*
*&                                                                     *
*&                                                                     *
*&---------------------------------------------------------------------*

REPORT  ZEMP_OUTBOUND                   .

 tables : zemp_mast ,
          zemp_qual .

 data : x_control like edidc ,
        x_z1ehdr   like Z1EHDR ,
        x_z1qual   like Z1QUAL .

data : it_z1qual    like z1qual occurs 0 with header line ,
       it_edidd    like edidd occurs 0 with header line ,
       it_control  like edidc occurs 0 with header line .


selection-screen begin of block B0001 with frame.
   parameters : p_empid  like zemp_mast-empid  OBLIGATORY,
                p_dest    like tbdlst-logsys .
selection-screen end of block B0001.


START-OF-SELECTION.
 PERFORM SUB_FETCH_MASTER_RECORDS.
 PERFORM SUB_FETCH_EMPLOYEE_INFO.

END-OF-SELECTION.
 PERFORM SUB_POPULATE_CONTROL_RECORD.
 PERFORM SUB_POPULATE_HEADER_INFO.
 PERFORM SUB_POPULATE_STUDENT_INFO.
 PERFORM SUB_CALL_FM.




*&---------------------------------------------------------------------*
*&      Form  SUB_FETCH_MASTER_RECORDS
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
form SUB_FETCH_MASTER_RECORDS .

     SELECT SINGLE * FROM ZEMP_MAST
       WHERE EMPID = P_EMPID.

   IF sy-subrc ne 0.
    message i398(00) with 'No employee record found'.
    leave list-processing.
   ENDIF.

endform.                    " SUB_FETCH_MASTER_RECORDS
*&---------------------------------------------------------------------*
*&      Form  SUB_FETCH_STUDENT_INFO
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
form SUB_FETCH_EMPLOYEE_INFO .

      SELECT pyear qual FROM Zemp_qual
      into table it_z1qual
      WHERE EMPID = p_empid.

    IF sy-subrc ne 0.
    message i398(00) with 'No qualification record for the employee' p_empid 'found'.
    leave list-processing.
   ENDIF.
endform.                    " SUB_FETCH_EMPLOYEE_INFO
*&---------------------------------------------------------------------*
*&      Form  SUB_POPULATE_CONTROL_RECORD
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
form SUB_POPULATE_CONTROL_RECORD .

 X_CONTROL-MESTYP = 'ZEMPMESSAGE'.
 X_CONTROL-DOCTYP = 'ZEMPIDOC'.
 X_CONTROL-RCVPRT = 'LS'.
 X_CONTROL-RCVPRN = P_DEST.


endform.                    " SUB_POPULATE_CONTROL_RECORD
*&---------------------------------------------------------------------*
*&      Form  SUB_POPULATE_HEADER_INFO
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
form SUB_POPULATE_HEADER_INFO .

 X_Z1EHDR-EMPID  = ZEMP_MAST-EMPID.
 X_Z1EHDR-EMPNAME = ZEMP_MAST-EMPNAME.

 IT_EDIDD-SEGNAM = 'Z1EHDR'.
 IT_EDIDD-SDATA  = X_Z1EHDR.
 APPEND IT_EDIDD.
 CLEAR IT_EDIDD.

endform.                    " SUB_POPULATE_HEADER_INFO
*&---------------------------------------------------------------------*
*&      Form  SUB_POPULATE_STUDENT_INFO
*&---------------------------------------------------------------------*
*       text
```

```
*---------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
form SUB_POPULATE_STUDENT_INFO .

 LOOP AT it_z1qual.
  x_z1qual-pyear = it_z1qual-pyear.
  x_z1qual-qual   = it_z1qual-qual .

  it_edidd-segnam = 'Z1QUAL'.
  it_edidd-sdata = X_Z1QUAL.
  APPEND IT_EDIDD.
  CLEAR IT_EDIDD.
 ENDLOOP.



endform.                    " SUB_POPULATE_STUDENT_INFO
*&---------------------------------------------------------------------*
*&      Form  SUB_CALL_FM
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
form SUB_CALL_FM .

  CALL FUNCTION 'MASTER_IDOC_DISTRIBUTE'
    EXPORTING
      master_idoc_control                  = x_control
*     OBJ_TYPE                            = ''
*     CHNUM                               = ''
    tables
      communication_idoc_control           = it_control
      master_idoc_data                     = it_edidd
   EXCEPTIONS
     ERROR_IN_IDOC_CONTROL                = 1
     ERROR_WRITING_IDOC_STATUS            = 2
     ERROR_IN_IDOC_DATA                   = 3
     SENDING_LOGICAL_SYSTEM_UNKNOWN       = 4
     OTHERS                               = 5
            .
  IF sy-subrc <> 0.
     message i398(00) with 'Problem in ALE service Layer'.
     leave list-processing.
  ELSE.
    loop at it_control.
     write:/5 'IDoc generated : ' ,  it_control-docnum .
    endloop.
    commit work.
  ENDIF.

endform.                    " SUB_CALL_FM
```

6.13. Develop Inbound Function Module in the Receiver System(SE37)

```
FUNCTION Z_IDOC_INPUT_EMP.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  IMPORTING
*"     VALUE(INPUT_METHOD) LIKE  BDWFAP_PAR-INPUTMETHD
*"     VALUE(MASS_PROCESSING) LIKE  BDWFAP_PAR-MASS_PROC
*"  EXPORTING
*"     VALUE(WORKFLOW_RESULT) LIKE  BDWF_PARAM-RESULT
*"     VALUE(APPLICATION_VARIABLE) LIKE  BDWF_PARAM-APPL_VAR
*"     VALUE(IN_UPDATE_TASK) LIKE  BDWFAP_PAR-UPDATETASK
*"     VALUE(CALL_TRANSACTION_DONE) LIKE  BDWFAP_PAR-CALLTRANS
*"  TABLES
*"      IDOC_CONTRL STRUCTURE  EDIDC
*"      IDOC_DATA STRUCTURE  EDIDD
*"      IDOC_STATUS STRUCTURE  BDIDOCSTAT
*"      RETURN_VARIABLES STRUCTURE  BDWFRETVAR
*"      SERIALIZATION_INFO STRUCTURE  BDI_SER
*"  EXCEPTIONS
*"      WRONG_FUNCTION_CALLED
*"----------------------------------------------------------------------

*-----Data Declaration------------------------------------*


* Work area for class data
data : x_z1ehdr like z1ehdr ,
       x_z1qual like z1qual ,
       l_success type i .

data : it_empm   like zemp_mast   occurs 0 with header line ,
       it_qual like zemp_qual occurs 0 with header line .

*-----End of data declaration-------------------------------*

       workflow_result = 0.

 break-point.
  loop at idoc_contrl.

       clear : l_success.

        if idoc_contrl-mestyp ne 'ZEMPMESSAGE'.
         raise wrong_function_called.
         exit.
        endif.

        clear : x_z1ehdr ,
                x_z1qual ,
                it_empm ,
                it_qual .

         refresh : it_empm,
                   it_qual .

        loop at idoc_data where docnum eq idoc_contrl-docnum .

         case idoc_data-segnam .
          when 'z1ehdr'.
           x_z1ehdr = idoc_data-sdata.
```

```
   it_empm-empid = x_z1ehdr-empid.
   it_empm-empname = x_z1ehdr-empname .
   append it_empm.
   clear it_empm.

  when 'z1qual'.
   x_z1qual = idoc_data-sdata.
   it_qual-empid = x_z1ehdr-empid .
   it_qual-pyear  = x_z1qual-pyear .
   it_qual-qual   = x_z1qual-qual.
   append it_qual.
   clear it_qual.
 endcase.
endloop.


sort it_empm by   empid.
sort it_qual by empid pyear.
delete adjacent duplicates from it_empm comparing empid.
delete adjacent duplicates from it_qual comparing empid pyear.

loop at it_empm.

  SELECT SINGLE * FROM ZEMP_MAST
   WHERE EMPID  = it_empm-empid.


   IF sy-subrc ne 0.
    insert into zemp_mast values it_empm.
    l_success = l_success + 1.
   endif.

   loop at it_qual where empid =  it_empm-empid.



    select single * from zemp_qual
      where empid = it_qual-empid
        and pyear = it_qual-pyear.



      if sy-subrc ne 0.
       insert into zemp_qual values it_qual.
       l_success = l_success + 1.
      else.

       update zemp_qual from it_qual.
       l_success = l_success + 1.
      endif.
  endloop.
 endloop.

    if l_success gt 0.
      return_variables-wf_param = 'Processed_IDOCs'.
      return_variables-doc_number = IDOC_CONTRL-DOCNUM.
      return_variables-wf_param = 'Appl_Objects'.
      concatenate x_z1ehdr-empid
                  '/'
                  x_z1ehdr-empname
                  into return_variables-doc_number.
      append return_variables.
```

```
        idoc_status-docnum = idoc_contrl-docnum.
        idoc_status-status = '53'.
        idoc_status-msgty = 'I'.
        idoc_status-msgid = '00'.
        idoc_status-msgno = '398'.
        concatenate x_z1ehdr-empid
                    '/'
                    x_z1ehdr-empname
                    into idoc_status-msgv1.
        append idoc_status.
      else.
       workflow_result = '99999'.
         return_variables-wf_param = 'Error_IDOCs'.
        return_variables-doc_number = IDOC_CONTRL-DOCNUM.
        return_variables-wf_param = 'Appl_Objects'.

        append return_variables.

        idoc_status-docnum = idoc_contrl-docnum.
        idoc_status-status = '51'.
        idoc_status-msgty = 'E'.
        idoc_status-msgid = '00'.
        idoc_status-msgno = '398'.
        concatenate x_z1ehdr-empid
                    '/'
                    x_z1ehdr-empname
                    into idoc_status-msgv1.
        append idoc_status.
       endif.
      endloop.

endfunction.
```

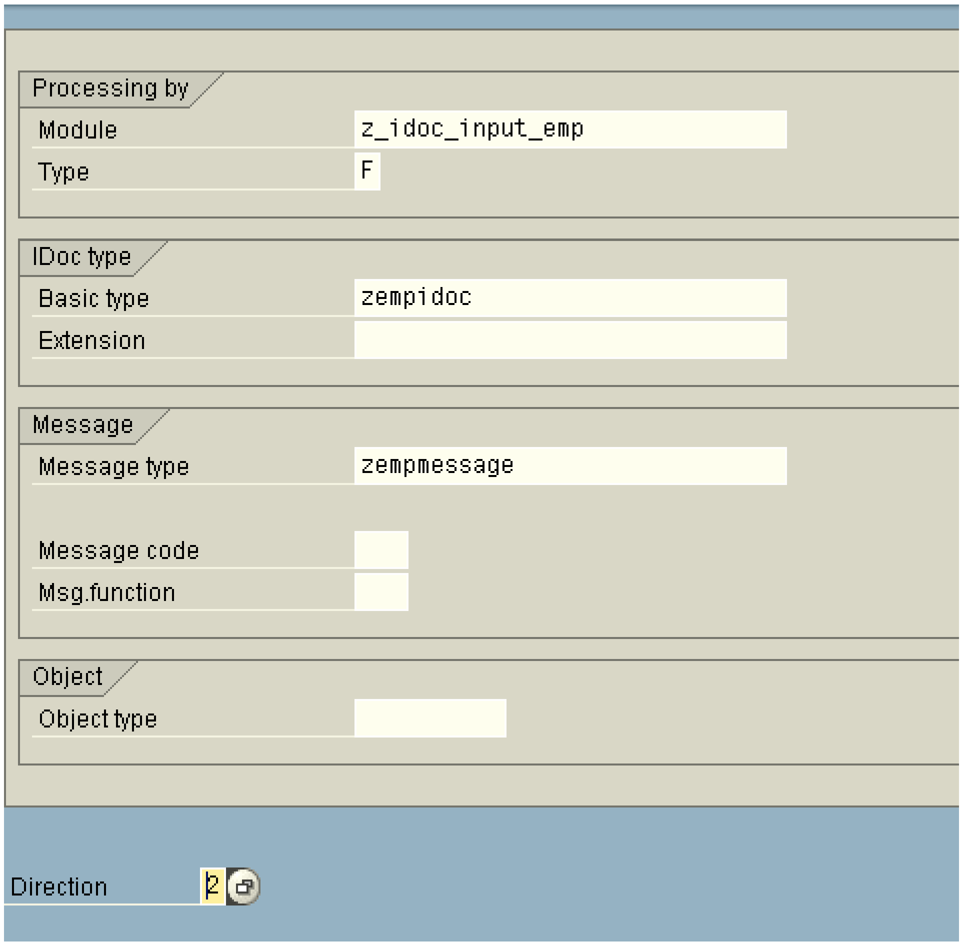
6.14. Create new Idoc Object in Business Object Repository(SWO1) in Receiver system

Idoc object zemp001 was developed. For details, see the book by A.Nagpal, page no. 660

6.15. Create a new task based on Application Idoc object(PFTC) in Receiver system

Done., the task is also for zemp001.

6.16. Allocate Function Module to the Message type(WE57) in Receiver system


Processing by

Module	z_idoc_input_emp
Type	F

IDoc type

Basic type	zempidoc
Extension	

Message

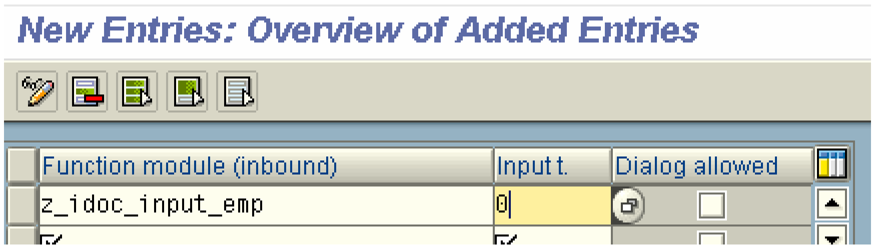
Message type	zempmessage
Message code	
Msg.function	

Object

Object type	

Direction 2

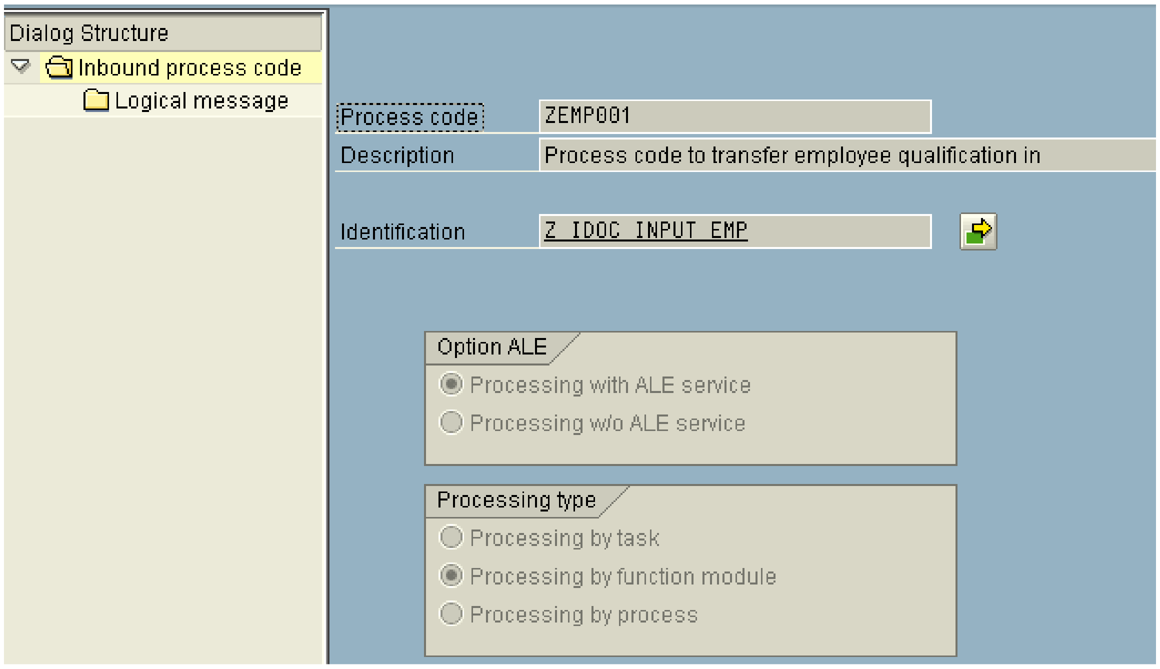
6.17. Define settings for Inbound FM in Receiver system(BD51)



New Entries: Overview of Added Entries

Function module (inbound)	Input t.	Dialog allowed
z_idoc_input_emp	0	☐

6.18. Create New Process code for the Inbound process(WE42) in Receiver system



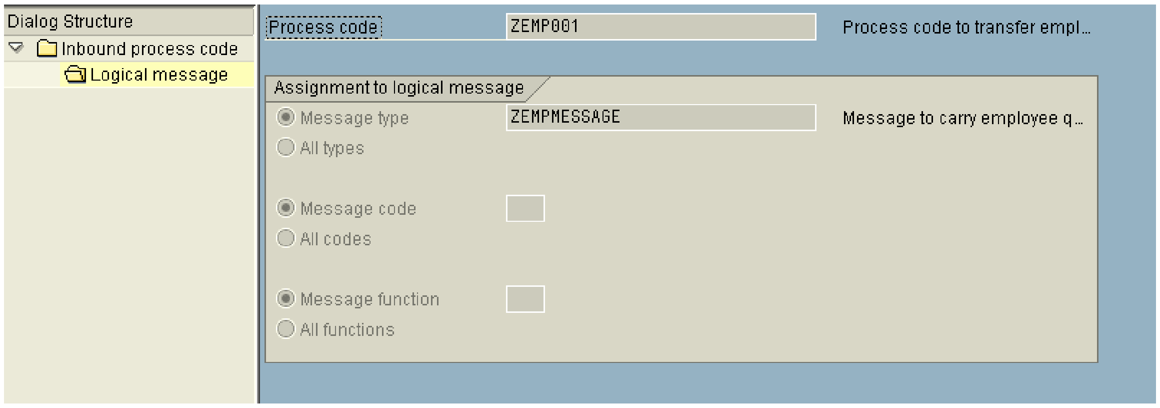
Dialog Structure
- Inbound process code
  - Logical message

Process code: ZEMP001
Description: Process code to transfer employee qualification in

Identification: Z_IDOC_INPUT_EMP

Option ALE
- [✓ Processing with ALE service]
- Processing w/o ALE service

Processing type
- Processing by task
- [✓ Processing by function module]
- Processing by process

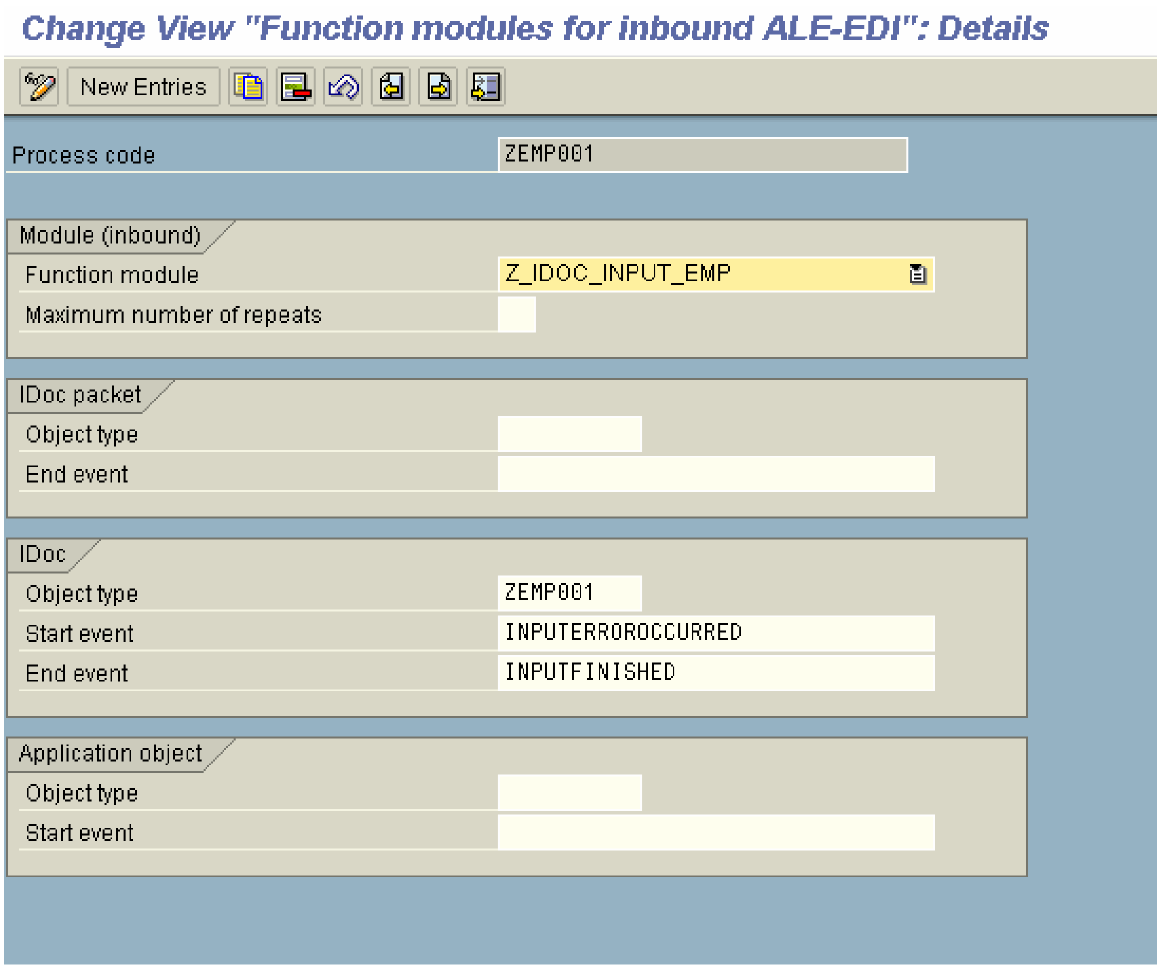


Dialog Structure
- Inbound process code
  - Logical message

Process code: ZEMP001 Process code to transfer empl...

Assignment to logical message
- [✓ Message type] ZEMPMESSAGE Message to carry employee q...
- All types
- [✓ Message code]
- All codes
- [✓ Message function]
- All functions

6.19. Assign Input Methods(BD67) in Receiver System


Change View "Function modules for inbound ALE-EDI": Details

Process code	ZEMP001

Module (inbound)

Function module	Z_IDOC_INPUT_EMP
Maximum number of repeats	

IDoc packet

Object type	
End event	

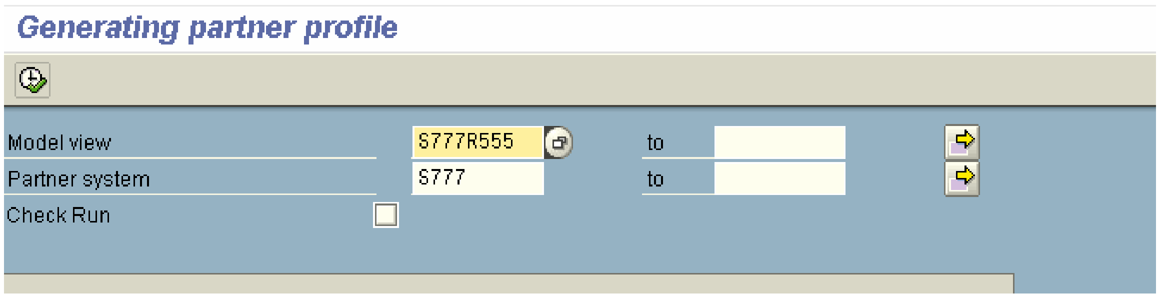
IDoc

Object type	ZEMP001
Start event	INPUTERROROCCURRED
End event	INPUTFINISHED

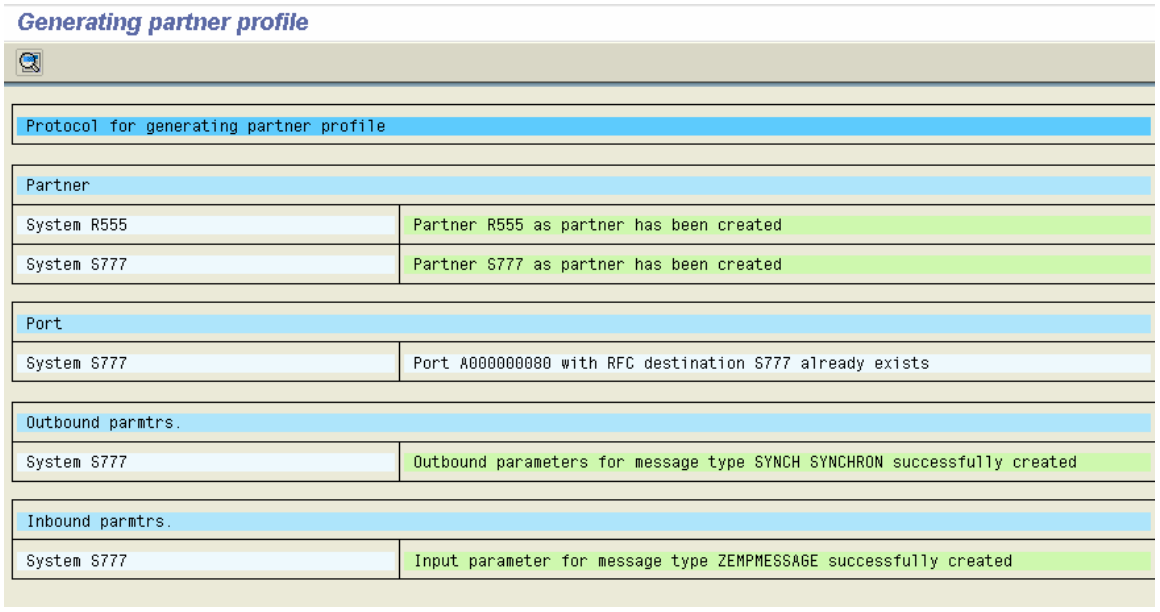
Application object

Object type	
Start event	

6.20. Generate Partner Profile in Receiver System(BD82)


Generating partner profile

Model view	S777R555	to	
Partner system	S777	to	
Check Run	☐		


Generating partner profile

```
Protocol for generating partner profile

Partner
System R555        Partner R555 as partner has been created
System S777        Partner S777 as partner has been created

Port
System S777        Port A000000080 with RFC destination S777 already exists

Outbound parmtrs.
System S777        Outbound parameters for message type SYNCH SYNCHRON successfully created

Inbound parmtrs.
System S777        Input parameter for message type ZEMPMESSAGE successfully created
```

Now, create an outbound record from the sender system using program ZEMP_OUTBOUND. An Idoc will be created and transferred to sender system , which will finally get assimilated into the database tables.

7. Standard SAP Idoc Extension

Introduction

Information on vendor is conveyed from one system to another using message type : CREMAS . But, it does not contain the following information on vendors:-

1. Reference of the vendor.
2. Rating of the vendor
3. Mobile number of the vendor.

This information is preserved in the sender system and is distributed to the receiver system(s) by extending the standard SAP Idoc.

Shatadru, client 777 (user: aleuser) is the sender system and client 555 (developer08) is the receiver system over here.

7.1. Steps to be followed

The steps to be followed to complete configuration and development in both the systems are outlined below in form of a table.

Srl. No	Description	(C)onfiguration/ (D)evelopment	Tcode	In (S)ender or (R)eceiver
1	Create an append structure ZVENDINFO to table LFA1 containing the following fields:- Perref (DE : ZREF) Ratings (DE : ZRATING) MOBILE(DE: ZMOBILE)	D	SE11	Both S and R
2	Adjust program by screen exit or build custom program to populate fields in the append structure for LFA1	D	SE38	S
3	Create custom segment ZVEND with the additional fields in step 1	C	WE31	S and R
4	Create extension CREMSUB of basic Idoc type CREMAS04 with segment ZVEND as child	C	WE30	S and R

Srl. No	Description	(C)onfiguration/ (D)evelopment	Tcode	In (S)ender or (R)eceiver
5.	Maintain the newly created extension linkage with message type and basic Idoc type	C	WE82	S
6.	Maintain the newly created extension in the partner profile for the receiving system	C	WE20	S
7.	In user exits, write code to populate the additional segment attached with basic Idoc type	D	CMOD,SMOD,SE 38	S
8	Test the outbound system		BD14,WE05	S
9	Maintain the linkage between message type, basic Idoc type, new extension	C	WE57	R
10	Find out suitable user-exit to update LFA1 from the additional info in the custom segment	D	SE38	R
11	Test the whole connection		BD14, WE05, SE11	R and S

7.2. Assumptions

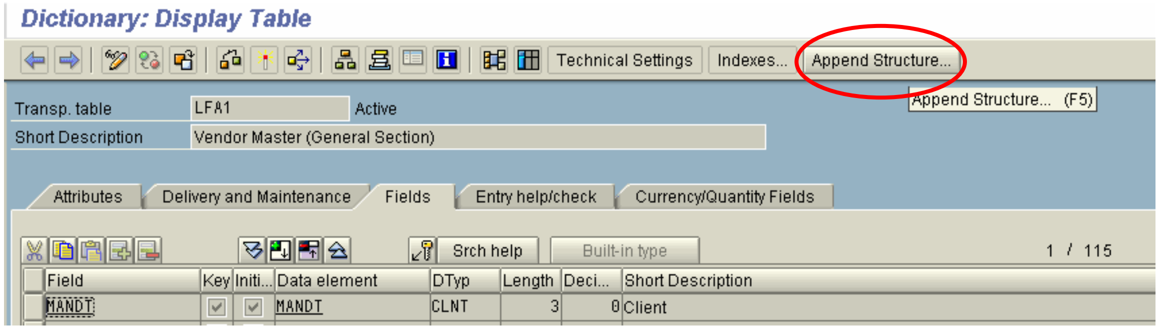
This documents assumes that the following basicconfigurations exists on sender and receiver systems to communicate vendor information:-

- Logical systems for sender and receiver on both systems.
- Assignment of logical systems to respective clients in respective systems.
- Remote connection for sender in receiver and vice versa.
- Customer distribution model and partner profile in both systems.

7.3. Steps in Detail

The steps outlined above will be documented in this section . Adequate screen-shots will be provided to explain the scenario.

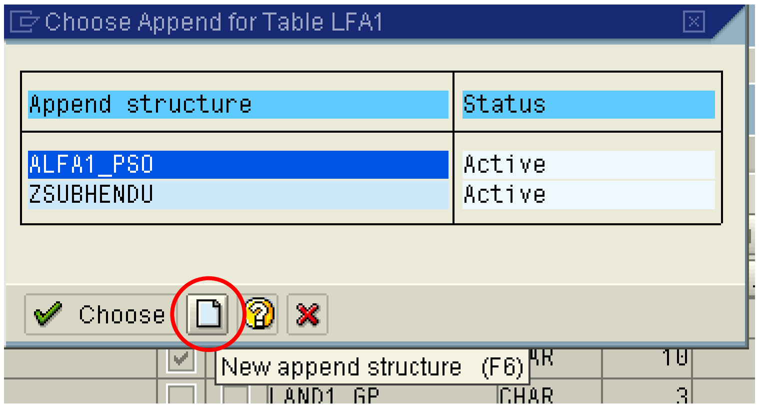
7.3.1. Step 1 – Build Append Structure ZVENDINFO on database table LFA1 in both systems


Dictionary: Display Table

Transp. table LFA1 Active
Short Description Vendor Master (General Section)

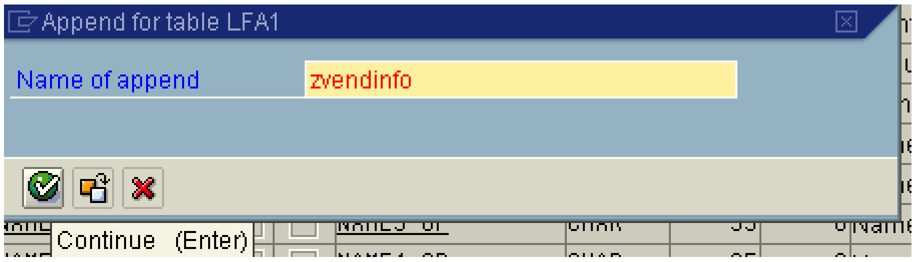
Field	Key	Initi...	Data element	DTyp	Length	Deci...	Short Description
MANDT	✓	✓	MANDT	CLNT	3	0	Client

Click the pushbutton as shown above to create append structure on LFA1 in display mode of tha table in SE11.


Choose Append for Table LFA1

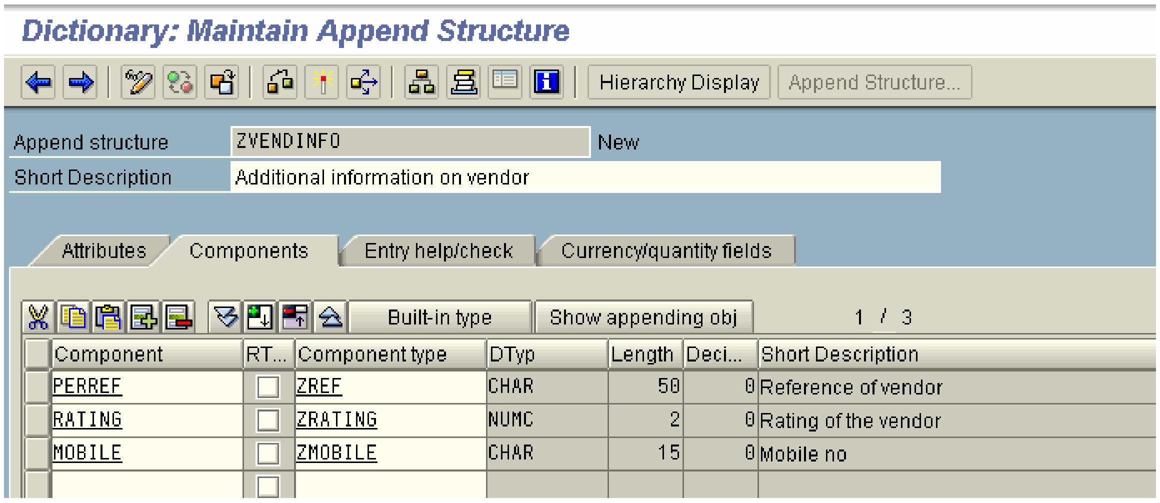
Append structure	Status
ALFA1_PSO	Active
ZSUBHENDU	Active

Press : New


Append for table LFA1

Name of append zvendinfo

Enter the name of the new append structure. Press : Enter.

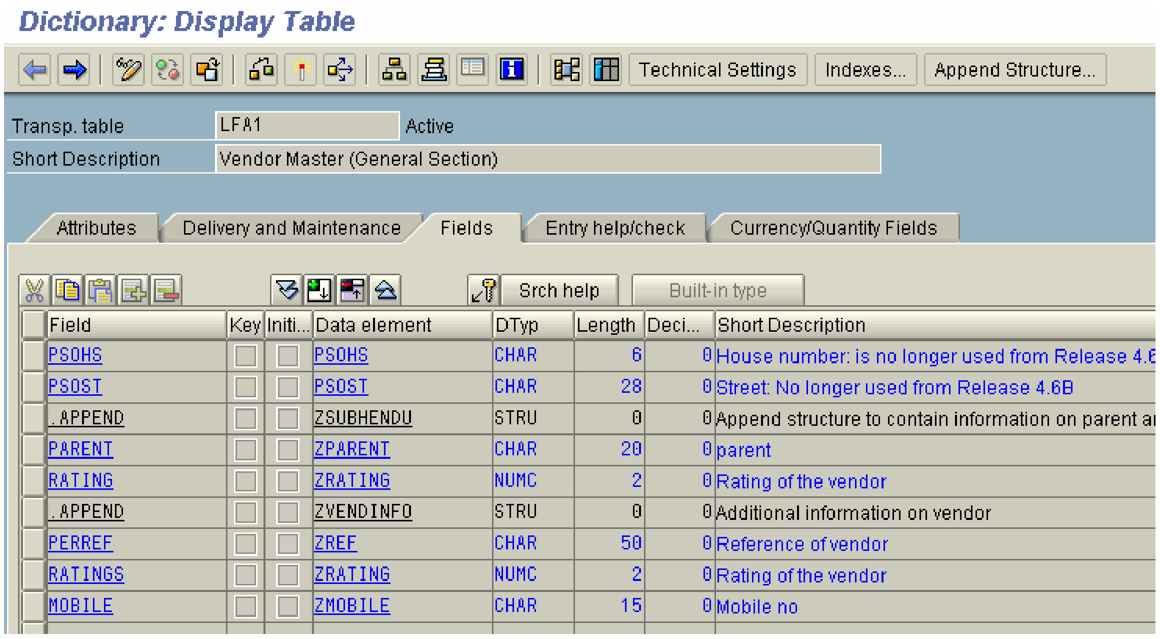


Dictionary: Maintain Append Structure

Append structure: ZVENDINFO New
Short Description: Additional information on vendor

Component	RT...	Component type	DTyp	Length	Deci...	Short Description
PERREF	☐	ZREF	CHAR	50	0	Reference of vendor
RATING	☐	ZRATING	NUMC	2	0	Rating of the vendor
MOBILE	☐	ZMOBILE	CHAR	15	0	Mobile no
	☐					

Create the components of the append structure. Create data elements and domains, if necessary. Then, save, activate and come out.



Dictionary: Display Table

Transp. table: LFA1 Active
Short Description: Vendor Master (General Section)

Field	Key	Initi...	Data element	DTyp	Length	Deci...	Short Description
PSOHS	☐	☐	PSOHS	CHAR	6	0	House number: is no longer used from Release 4.6
PSOST	☐	☐	PSOST	CHAR	28	0	Street: No longer used from Release 4.6B
.APPEND	☐	☐	ZSUBHENDU	STRU	0	0	Append structure to contain information on parent a
PARENT	☐	☐	ZPARENT	CHAR	20	0	parent
RATING	☐	☐	ZRATING	NUMC	2	0	Rating of the vendor
.APPEND	☐	☐	ZVENDINFO	STRU	0	0	Additional information on vendor
PERREF	☐	☐	ZREF	CHAR	50	0	Reference of vendor
RATINGS	☐	☐	ZRATING	NUMC	2	0	Rating of the vendor
MOBILE	☐	☐	ZMOBILE	CHAR	15	0	Mobile no

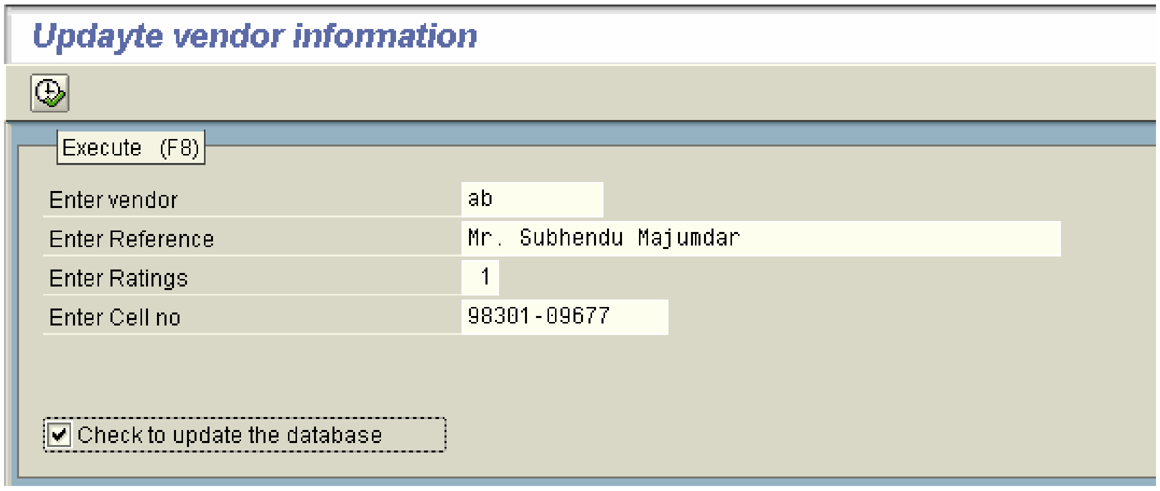
The append structure is now successfully added to database table LFA1

7.3.2. Step 2 - Adjust program by screen exit or build custom program to populate fields in the append structure for LFA1 in the sender system

```
*&---------------------------------------------------------------------*
*& Report  YVENDOR_MODIFY                                              *
*&                                                                     *
*&---------------------------------------------------------------------*
*&                                                                     *
*&                                                                     *
*&---------------------------------------------------------------------*

REPORT  YVENDOR_MODIFY                          .

TABLES : lfa1.

selection-screen begin of block b0001 with frame .
   parameters :
* Parameter for vendor number
              p_lifnr like lfa1-lifnr obligatory ,
* Parameter to enter reference name
              p_perref like lfa1-perref ,
* Parameter to enter rating of vendor
              p_rate like lfa1-ratings ,
* Parameter to enter mobile no
              p_mobile like lfa1-mobile .
 selection-screen skip 2.
* If the checkbox for update is checked, then only database table
* LFA1 will be updated with the user entered info in the selection-screen.
* Else, ponly information on reference, rating and mobile number of the vendor will be
* shown as a report.
   parameters :  p_update as checkbox .
selection-screen end of block b0001 .



  start-of-selection.

     select single * from lfa1
     where lifnr = p_lifnr .
     if sy-subrc eq 0 .
      if p_update = 'X'.
* If the vendor chosen by the user in the selection-screen exists in the
* database and if the user has checked the checkbox to update the vendor with newly
* added information in the selection-screen , then update the database.
        update lfa1
        set perref = p_perref
            ratings = p_rate
            mobile = p_mobile
            where lifnr = p_lifnr .
        if sy-subrc eq 0.
         message i398(00) with 'Updation successful'.
         commit work.
        ELSE .
         MESSAGE I398(00) WITH 'Updation not successful'.
        endif.
      else.
       message i398(00) with 'See report only'.
      endif.
      endif.
```

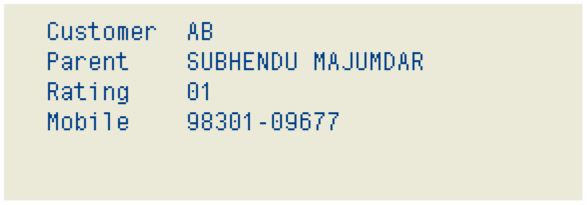
```
    end-of-selection.
* Fetch the updated information from the database for the vendor after updation
* and display
     select single * from lfa1 where lifnr = p_lifnr.
      if sy-subrc eq 0.
      write:/5 'Customer' , 15  lfa1-lifnr ,
           /5  'Parent'   , 15  lfa1-parent ,
           /5  'Rating'    , 15 lfa1-rating ,
           /5  'Mobile'   , 15 lfa1-mobile .
     else.
      write:/5 'No data found'.
     endif.
```

The selection-screen looks as follows:-



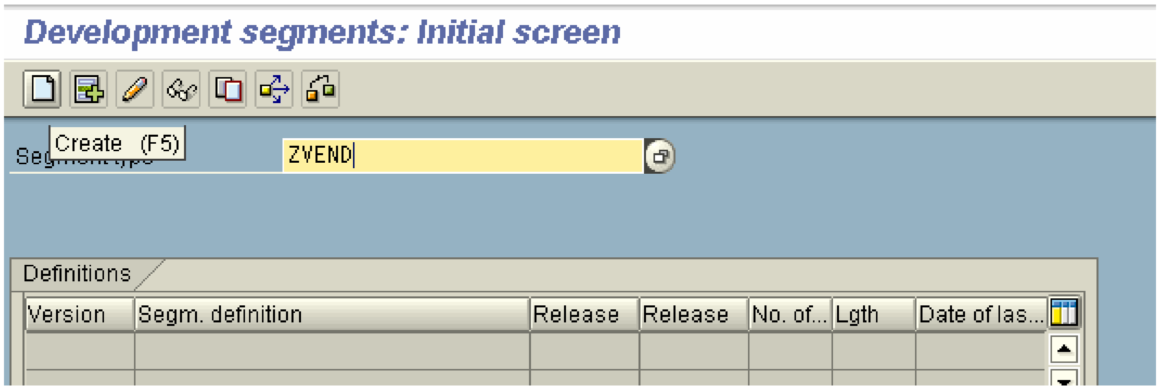
Enter vendor	ab
Enter Reference	Mr. Subhendu Majumdar
Enter Ratings	1
Enter Cell no	98301-09677

☑ Check to update the database



```
Customer  AB
Parent    SUBHENDU MAJUMDAR
Rating    01
Mobile    98301-09677
```

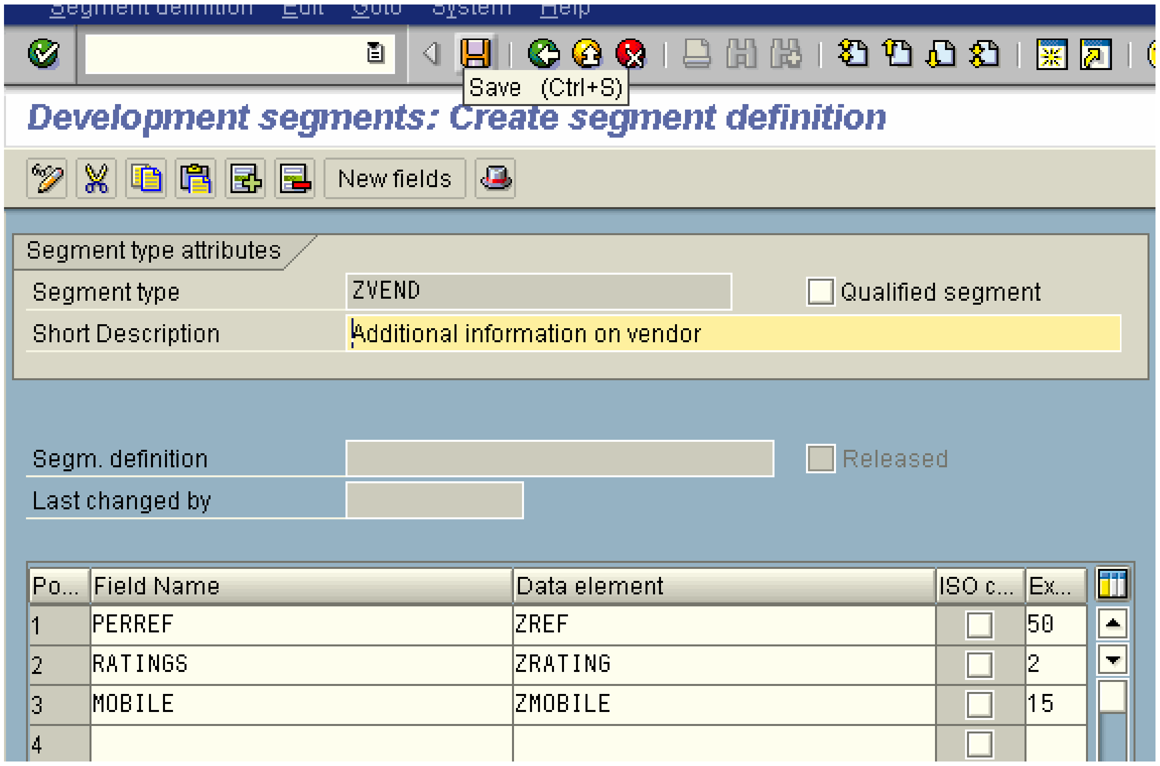
7.3.3. Step 3 - Create custom segment ZVEND containing the additional fields in step 1 in both systems


Development segments: Initial screen

Segment type: ZVEND

Version	Segm. definition	Release	Release	No. of...	Lgth	Date of las...

In transaction WE31, enter the name of the new segment and press : Create from application toolbar.

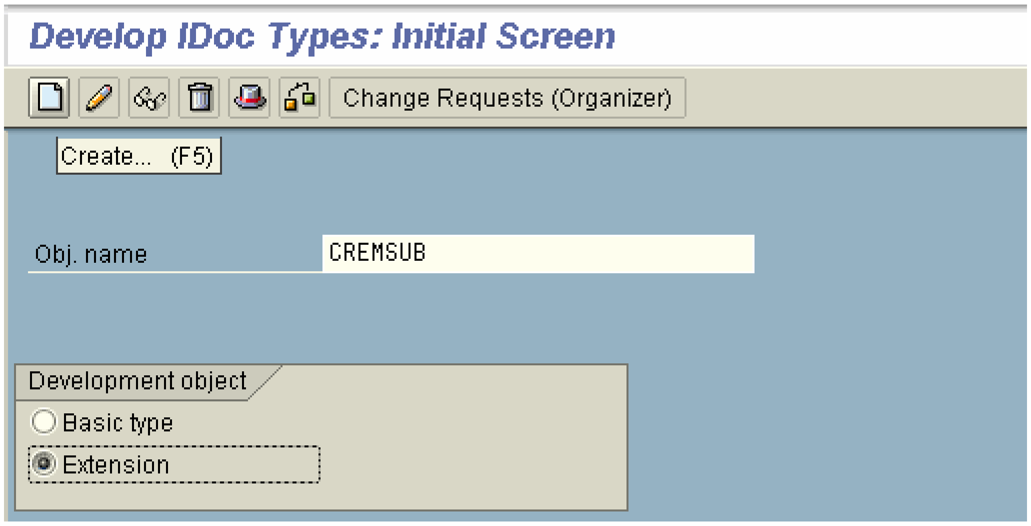

Development segments: Create segment definition

Segment type: ZVEND
Short Description: Additional information on vendor
Segm. definition:
Last changed by:

Po...	Field Name	Data element	ISO c...	Ex...
1	PERREF	ZREF	☐	50
2	RATINGS	ZRATING	☐	2
3	MOBILE	ZMOBILE	☐	15
4			☐	

Enter a short description. Then, specify the fields , their details and press : Save.

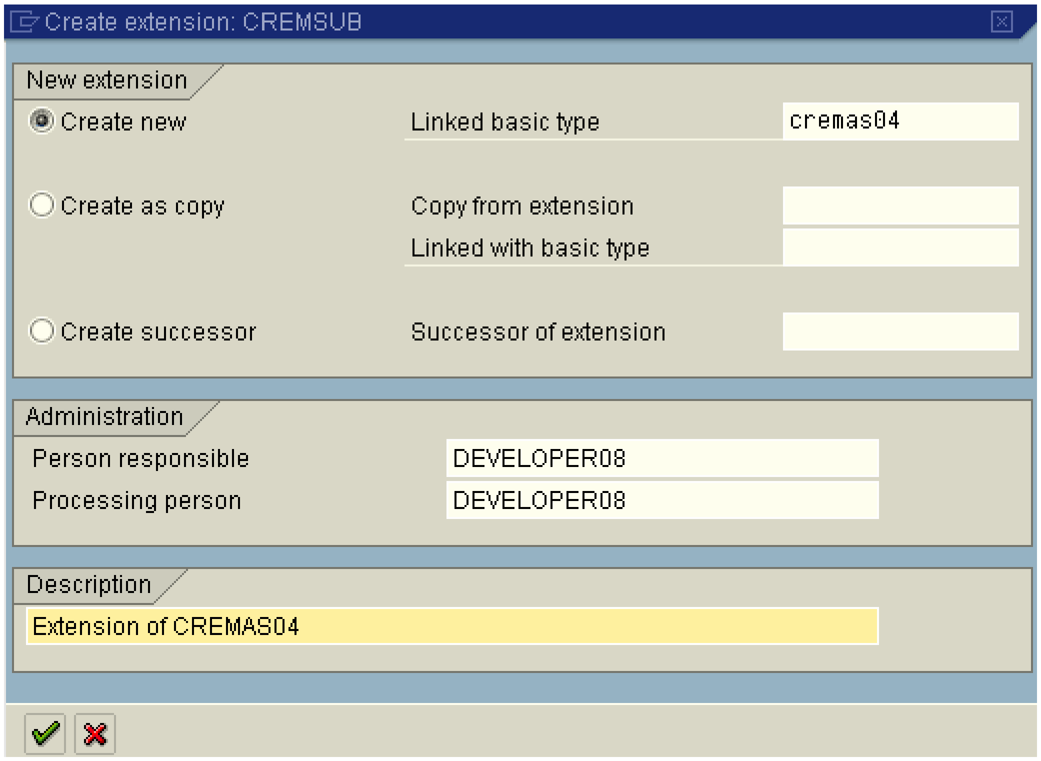
7.3.4. Step 4 : Create extension CREMSUB of basic Idoc type CREMAS04 with segment ZVEND as child in both sender and receiver


Develop IDoc Types: Initial Screen

Obj. name: CREMSUB

Development object
- ( ) Basic type
- (●) Extension

In the initial screen of transcation WE30, enter the name of the extension, choose : Extension and choose : Create from Application toolbar.

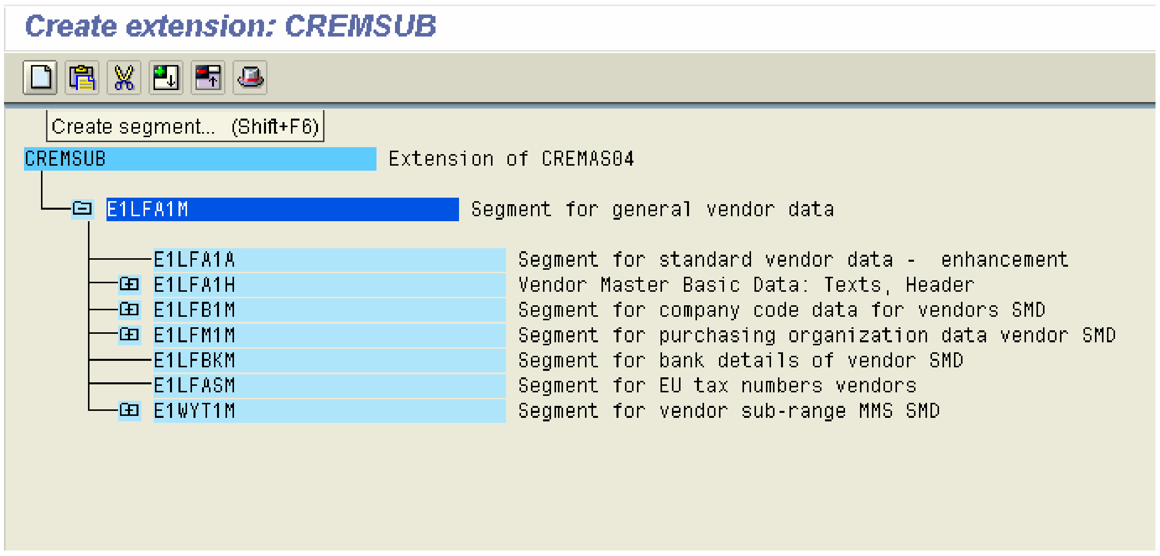

Create extension: CREMSUB

New extension
- (●) Create new — Linked basic type: cremas04
- ( ) Create as copy — Copy from extension: ; Linked with basic type:
- ( ) Create successor — Successor of extension:

Administration
- Person responsible: DEVELOPER08
- Processing person: DEVELOPER08

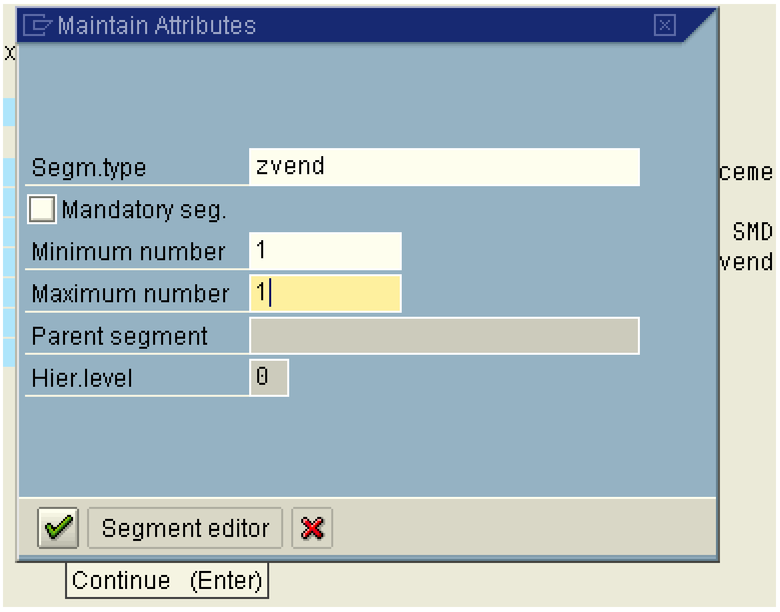
Description: Extension of CREMAS04

Enter as shown above. Press Enter.



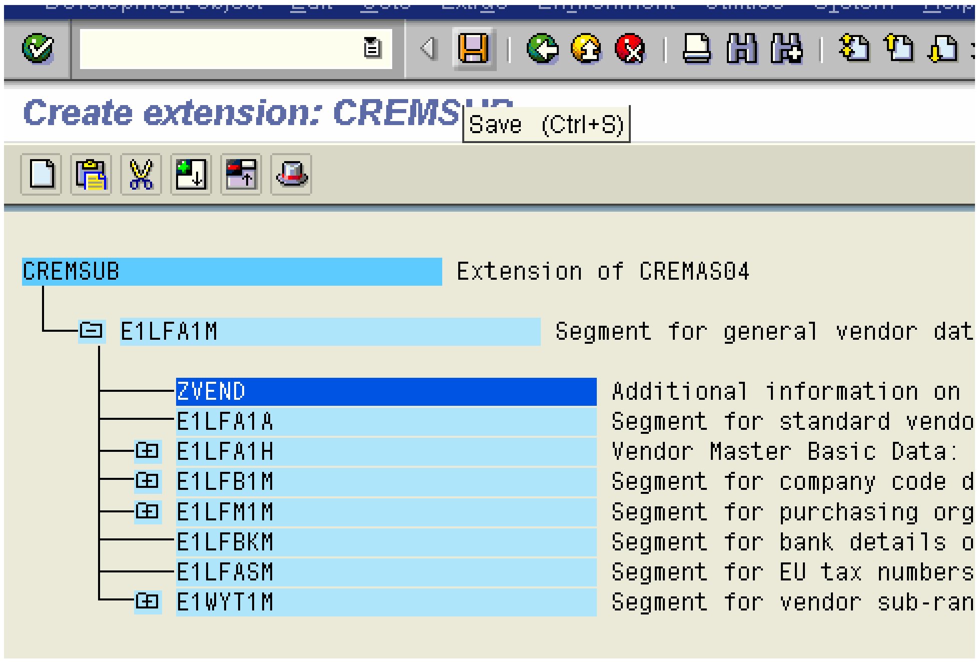
```
CREMSUB                    Extension of CREMAS04
  E1LFA1M                  Segment for general vendor data
    E1LFA1A                Segment for standard vendor data -  enhancement
    E1LFA1H                Vendor Master Basic Data: Texts, Header
    E1LFB1M                Segment for company code data for vendors SMD
    E1LFM1M                Segment for purchasing organization data vendor SMD
    E1LFBKM                Segment for bank details of vendor SMD
    E1LFASM                Segment for EU tax numbers vendors
    E1WYT1M                Segment for vendor sub-range MMS SMD
```

Choose the segment under which you want to create your child segment and choose : Create from Application Toolbar.



Segm.type	zvend
Mandatory seg.	☐
Minimum number	1
Maximum number	1
Parent segment	
Hier.level	0

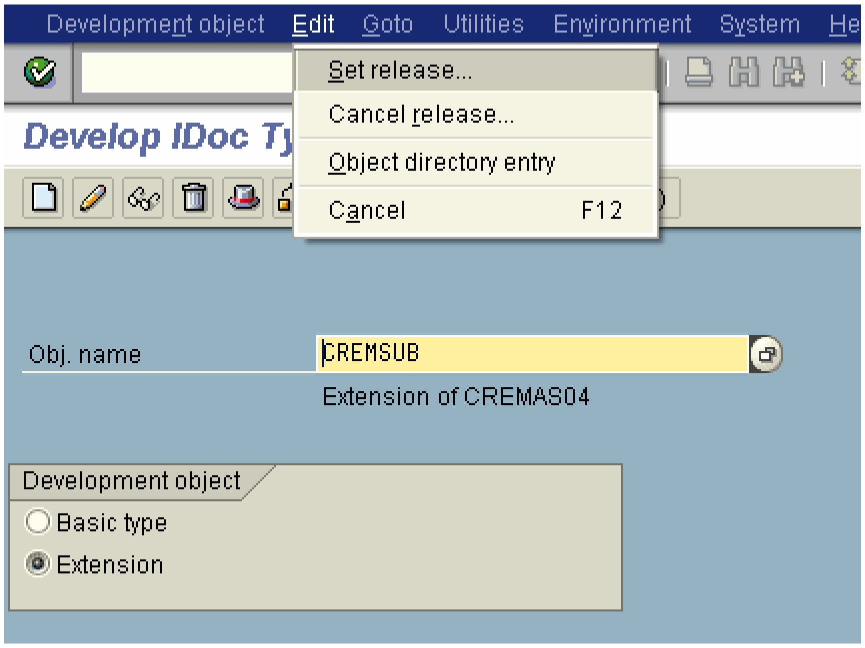
Enter the name of your child segment. Fill in the other details. Press Enter.



Create extension: CREMS

```
CREMSUB              Extension of CREMAS04
  E1LFA1M            Segment for general vendor dat
    ZVEND            Additional information on
    E1LFA1A          Segment for standard vendo
    E1LFA1H          Vendor Master Basic Data:
    E1LFB1M          Segment for company code d
    E1LFM1M          Segment for purchasing org
    E1LFBKM          Segment for bank details o
    E1LFASM          Segment for EU tax numbers
    E1WYT1M          Segment for vendor sub-ran
```

Your segment is added as a child segment under the chosen segment. Save.



Develop IDoc Ty

Obj. name: CREMSUB

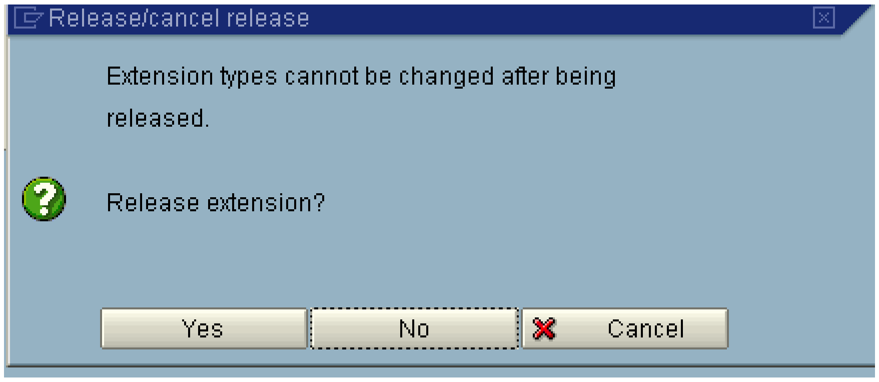
Extension of CREMAS04

Development object

( ) Basic type

(●) Extension

Then, transport the extension and the new segment from the initial screen of and WE30 following the appropriate menu paths.

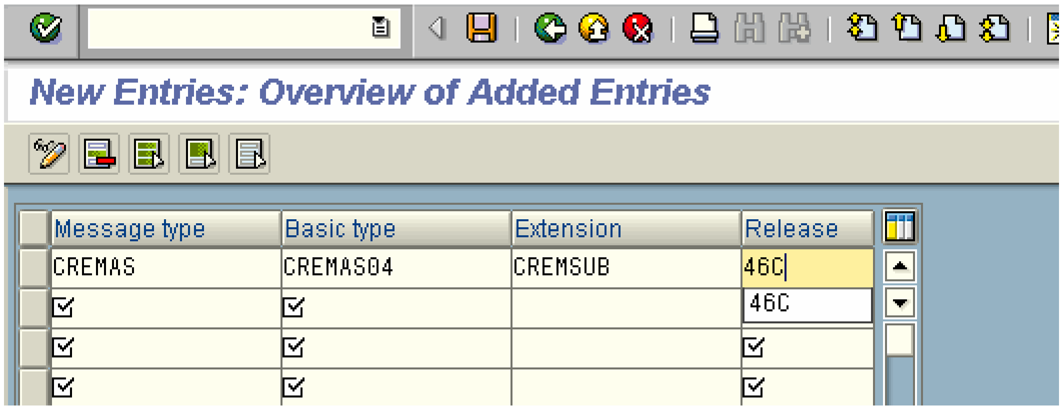

Release/cancel release

Extension types cannot be changed after being released.

Release extension?

Choose : Yes.

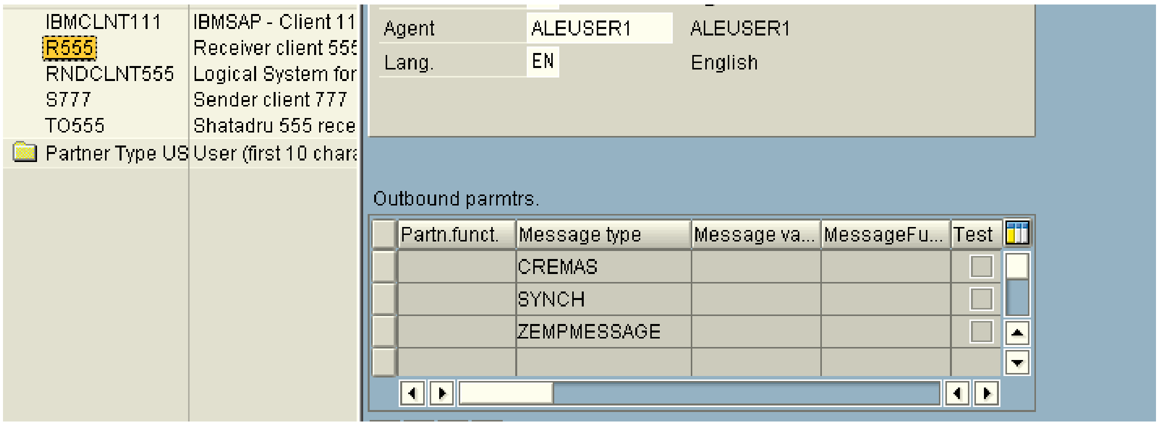
7.3.5. Step 5 : Maintain the newly created extension linkage with message type and basic Idoc type in Sender system


New Entries: Overview of Added Entries

Message type	Basic type	Extension	Release
CREMAS	CREMAS04	CREMSUB	46C
✓	✓		46C
✓	✓		✓
✓	✓		✓

Maintain the entry in WE82 of the sender system. Press Save.

7.3.6. Step 6 : Adjust the Partner Profile for CREMAS of the receiver system in the sender system using WE20



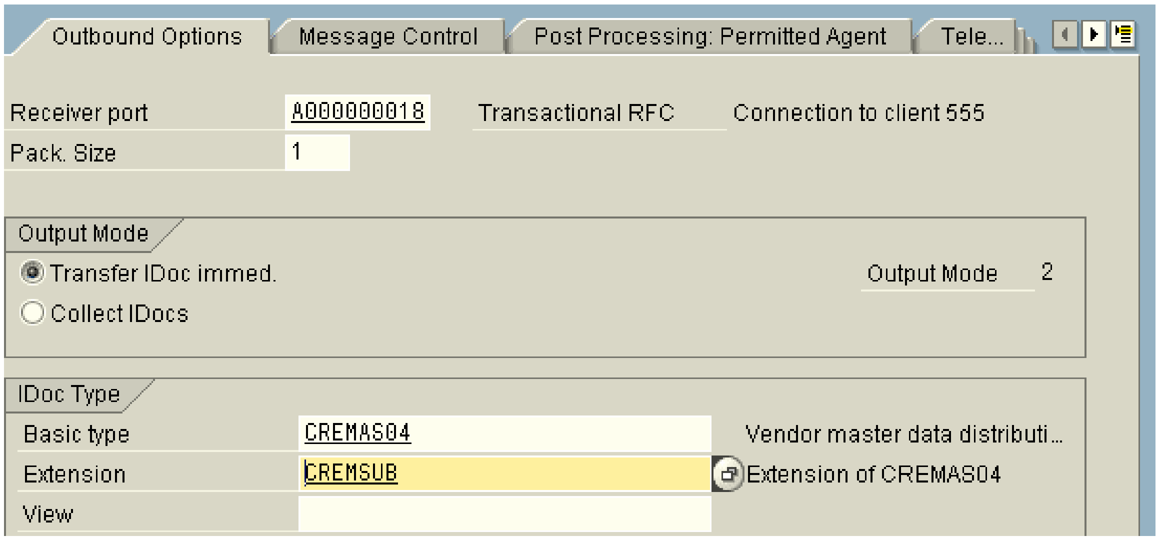
IBMCLNT111	IBMSAP - Client 11
R555	Receiver client 555
RNDCLNT555	Logical System for
S777	Sender client 777
TO555	Shatadru 555 rece
Partner Type US	User (first 10 chara

Agent ALEUSER1 ALEUSER1
Lang. EN English

Outbound parmtrs.

Partn.funct.	Message type	Message va...	MessageFu...	Test
	CREMAS			☐
	SYNCH			☐
	ZEMPMESSAGE			☐

Double-click on CREMAS.



Receiver port A000000018 Transactional RFC Connection to client 555
Pack. Size 1

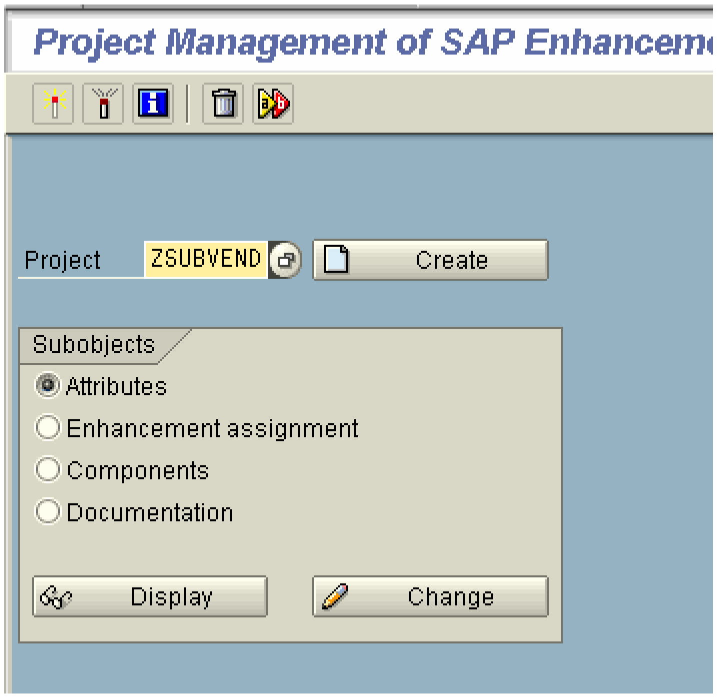
Output Mode
Transfer IDoc immed. [✓]
Collect IDocs
Output Mode 2

IDoc Type
Basic type CREMAS04 Vendor master data distributi...
Extension CREMSUB Extension of CREMAS04
View

Change the extension. Press : Save.

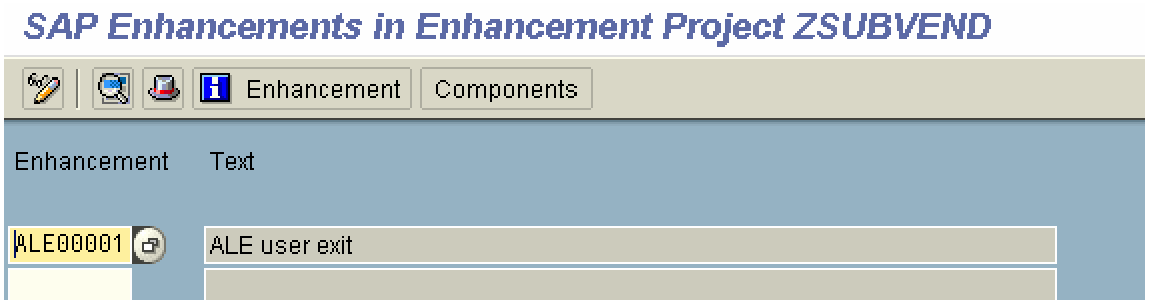
7.3.7. Step 7 : Write code in appropriate user-exit to populate the custom segment in outbound system

User-exit is available under the enhancement : ALE00001 . A project was developed containing the enhancement using tcode : CMOD and the custom include was populated with code for filling up custom segment : ZVEND in sender system.

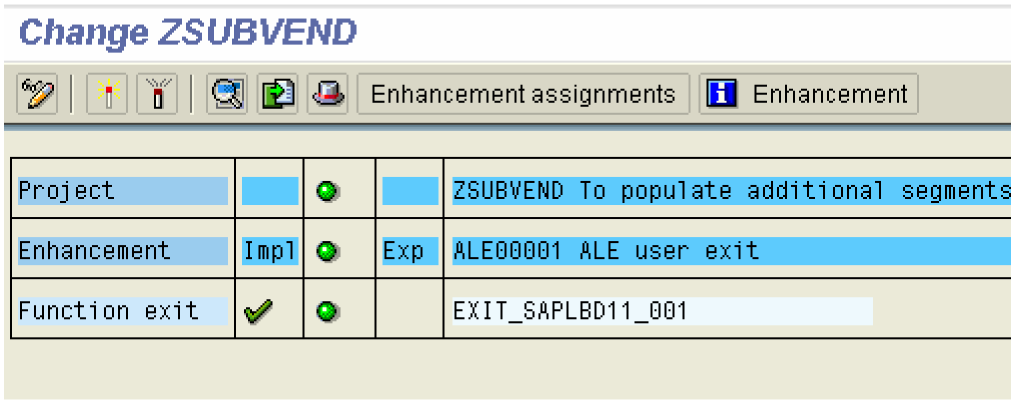

Project Management of SAP Enhanceme

Project: ZSUBVEND

Subobjects
- [✓ Attributes]
- Enhancement assignment
- Components
- Documentation

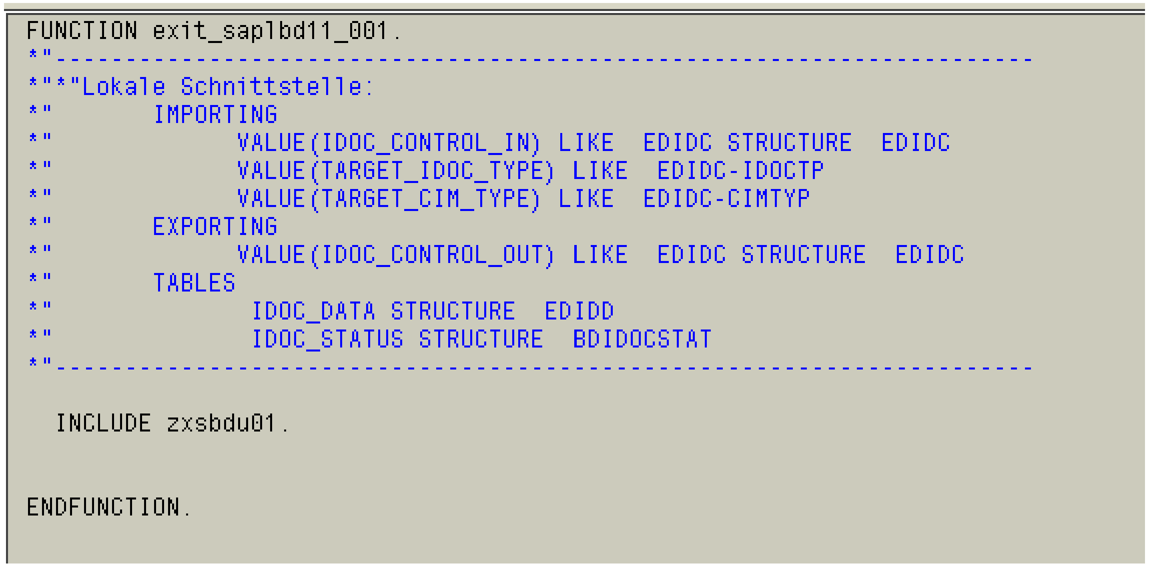

SAP Enhancements in Enhancement Project ZSUBVEND

Enhancement	Text
ALE00001	ALE user exit



Change ZSUBVEND

Project		●		ZSUBVEND To populate additional segments
Enhancement	Impl	●	Exp	ALE00001 ALE user exit
Function exit	✓	●		EXIT_SAPLBD11_001



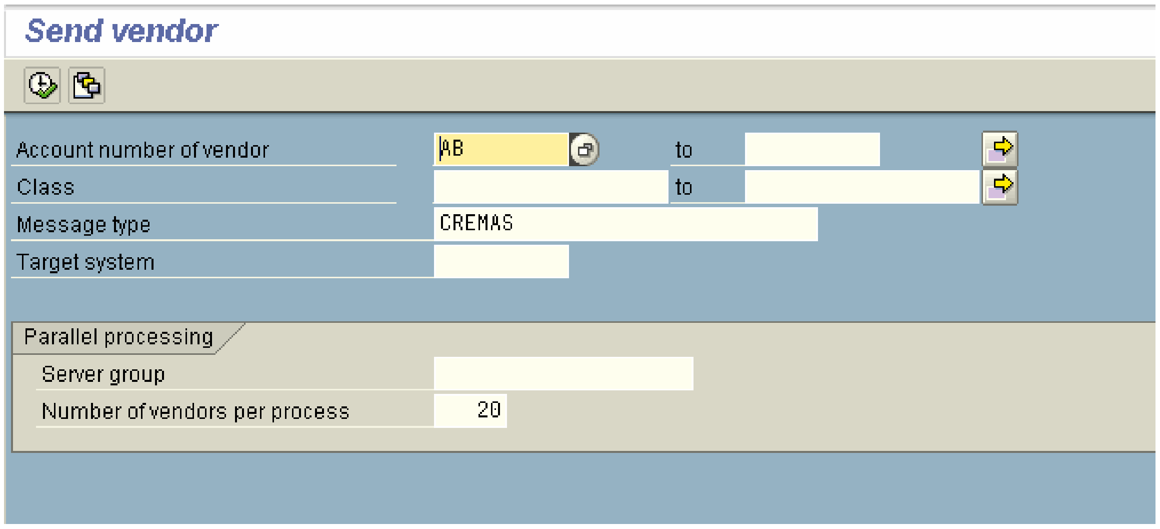
```
FUNCTION exit_saplbd11_001.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"       IMPORTING
*"             VALUE(IDOC_CONTROL_IN) LIKE  EDIDC STRUCTURE  EDIDC
*"             VALUE(TARGET_IDOC_TYPE) LIKE  EDIDC-IDOCTP
*"             VALUE(TARGET_CIM_TYPE) LIKE  EDIDC-CIMTYP
*"       EXPORTING
*"             VALUE(IDOC_CONTROL_OUT) LIKE  EDIDC STRUCTURE  EDIDC
*"       TABLES
*"              IDOC_DATA STRUCTURE  EDIDD
*"              IDOC_STATUS STRUCTURE  BDIDOCSTAT
*"----------------------------------------------------------------------

  INCLUDE zxsbdu01.


ENDFUNCTION.
```

The code inside the include is as follows:-

```
*&---------------------------------------------------------------------*
*&  Include           ZXSBDU01                                         *
*&---------------------------------------------------------------------*

data : lfa1m like E1LFA1M ,
       zvend like ZVEND .

data : perref like  lfa1-perref   ,
       ratings like lfa1-ratings ,
       mobile  like lfa1-mobile  ,
       ind type sy-tabix.

       data : x_val(20) type c .

 tables : ytodel.

case idoc_control_in-direct.
* Check for Outbound mode
when '1'.
* Check for Vendor information
 check idoc_control_in-mestyp = 'CREMAS'.
 loop at idoc_data .
  case idoc_data-segnam.
* From the main segment , get the vendor number and retrieve information
* on reference, rating and cell no to populate the new segment ZVEND
   when 'E1LFA1M'.
   ind = sy-tabix .
    move idoc_data-sdata to lfa1m.
    select single perref
                 ratings
                 mobile into
                 (perref,ratings,mobile )
           from lfa1
           where lifnr = lfa1m-lifnr .
           zvend-perref  = perref .
           zvend-ratings = ratings .
           zvend-mobile = mobile.

  endcase.
 endloop.
 ind = ind + 1.
* Insert the data for custom segment ZVEND after the segment E1LFA1M
 idoc_data-segnam = 'ZVEND'.
 move zvend to idoc_data-sdata .
 insert idoc_data index ind.
*check segment_name = 'E1LFA1M'



  endcase.
```

7.3.8. Step 8 : Test the Outbound System

Your configurations and developments for the development system is over. Now, it is time for testing.

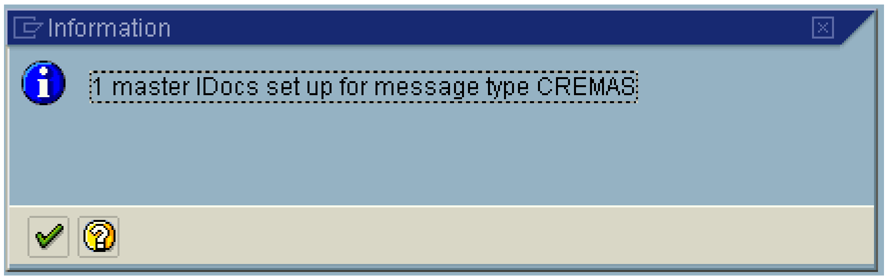

Send vendor

Account number of vendor	AB	to	
Class		to	
Message type	CREMAS		
Target system			

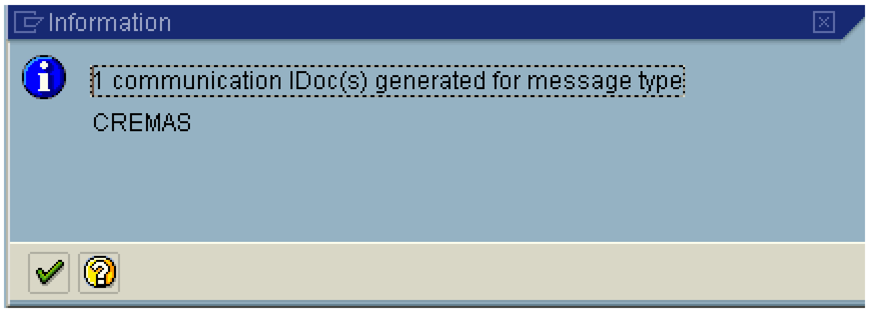
Parallel processing

Server group	
Number of vendors per process	20

Distribute a vendor from transaction BD14.

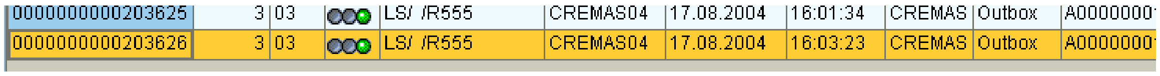

Information

1 master IDocs set up for message type CREMAS


Information

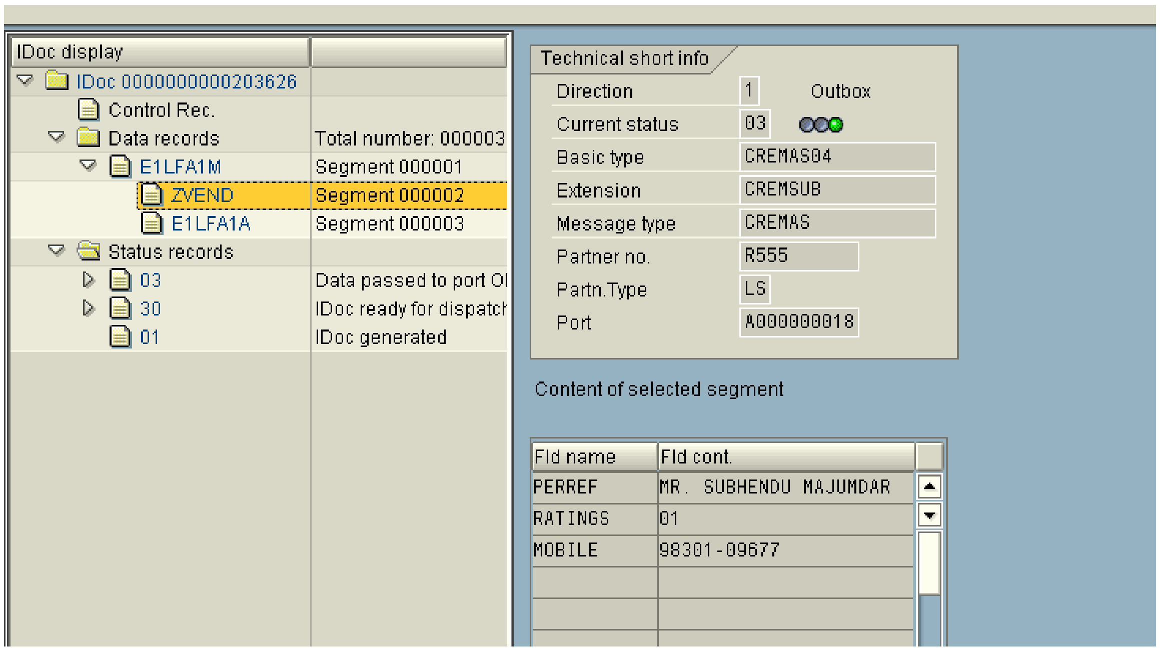
1 communication IDoc(s) generated for message type CREMAS

Go to WE05 and test the newly created Idoc.



0000000000203625	3	03	●●●	LS/ /R555	CREMAS04	17.08.2004	16:01:34	CREMAS	Outbox	A0000000
0000000000203626	3	03	●●●	LS/ /R555	CREMAS04	17.08.2004	16:03:23	CREMAS	Outbox	A0000000

Double-click on it



IDoc display	
IDoc 0000000000203626	
Control Rec.	
Data records	Total number: 000003
E1LFA1M	Segment 000001
ZVEND	Segment 000002
E1LFA1A	Segment 000003
Status records	
03	Data passed to port OI
30	IDoc ready for dispatch
01	IDoc generated

Technical short info

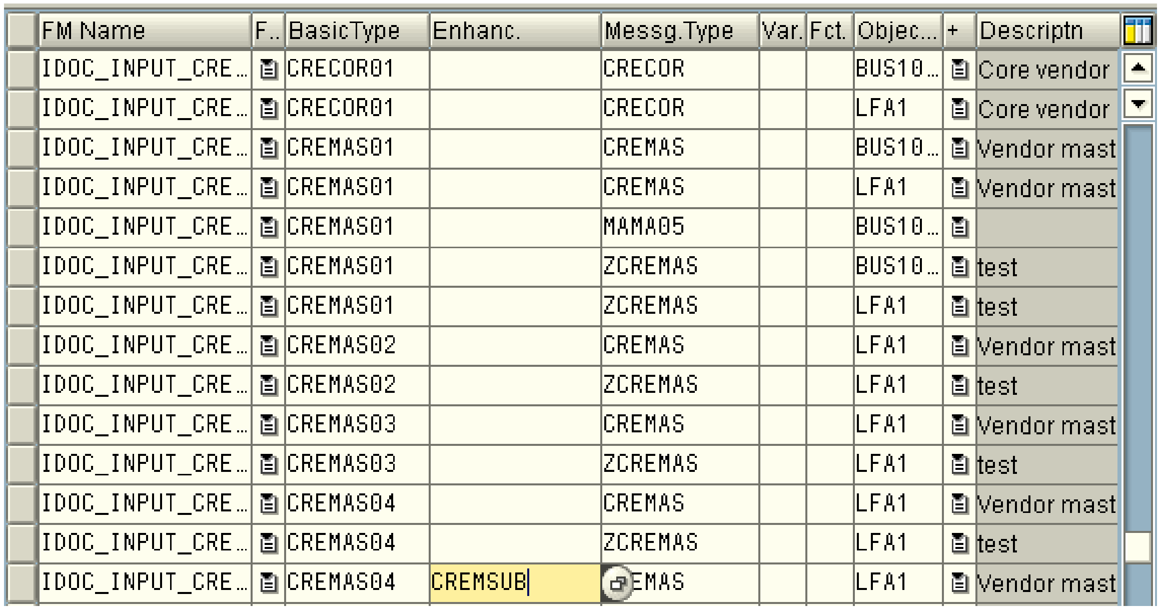
Direction	1 Outbox
Current status	03 ●●●
Basic type	CREMAS04
Extension	CREMSUB
Message type	CREMAS
Partner no.	R555
Partn.Type	LS
Port	A000000018

Content of selected segment

Fld name	Fld cont.
PERREF	MR. SUBHENDU MAJUMDAR
RATINGS	01
MOBILE	98301-09677

It shows that the custom segment is appropriately populated and the data is passed to port correctly.

7.3.9. Step 10 : Maintain the Linkage Between Message Type , Basic Idoc type and the New Extension in Receiver System using tcode : WE57



FM Name	F..	BasicType	Enhanc.	Messg.Type	Var.	Fct.	Objec...	+	Descriptn
IDOC_INPUT_CRE...	▤	CRECOR01		CRECOR			BUS10...	▤	Core vendor
IDOC_INPUT_CRE...	▤	CRECOR01		CRECOR			LFA1	▤	Core vendor
IDOC_INPUT_CRE...	▤	CREMAS01		CREMAS			BUS10...	▤	Vendor mast
IDOC_INPUT_CRE...	▤	CREMAS01		CREMAS			LFA1	▤	Vendor mast
IDOC_INPUT_CRE...	▤	CREMAS01		MAMA05			BUS10...	▤	
IDOC_INPUT_CRE...	▤	CREMAS01		ZCREMAS			BUS10...	▤	test
IDOC_INPUT_CRE...	▤	CREMAS01		ZCREMAS			LFA1	▤	test
IDOC_INPUT_CRE...	▤	CREMAS02		CREMAS			LFA1	▤	Vendor mast
IDOC_INPUT_CRE...	▤	CREMAS02		ZCREMAS			LFA1	▤	test
IDOC_INPUT_CRE...	▤	CREMAS03		CREMAS			LFA1	▤	Vendor mast
IDOC_INPUT_CRE...	▤	CREMAS03		ZCREMAS			LFA1	▤	test
IDOC_INPUT_CRE...	▤	CREMAS04		CREMAS			LFA1	▤	Vendor mast
IDOC_INPUT_CRE...	▤	CREMAS04		ZCREMAS			LFA1	▤	test
IDOC_INPUT_CRE...	▤	CREMAS04	CREMSUB	EMAS			LFA1	▤	Vendor mast

7.3.10. Step 11 : Write Code in Receiver side in user exits to populate database from additional info carried by custom segments

Enhancement VSV00001 contains call to FM EXIT_SAPLKD02_001 which contains a custom include where the code can be written.

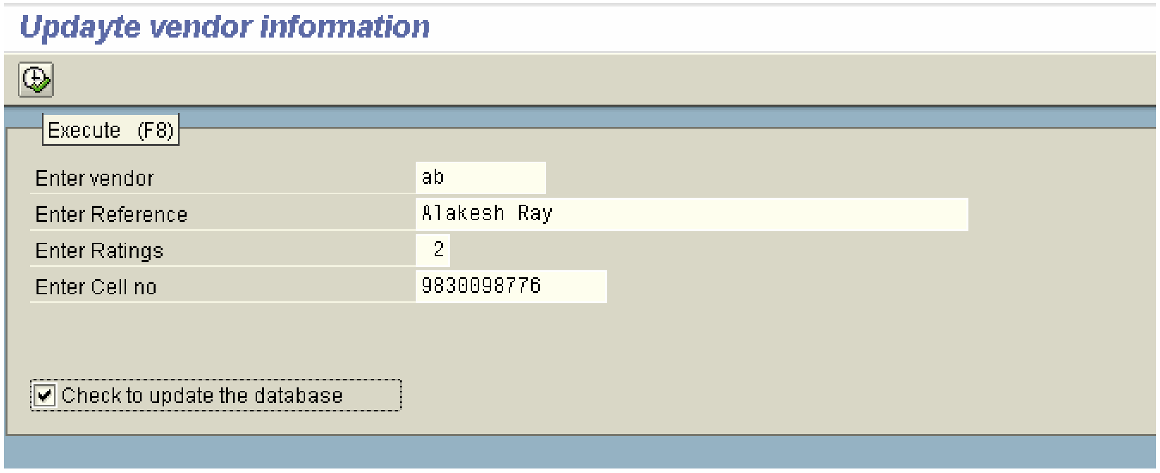
A project ZSUBINBD was developed containing the enhancement VSV00001 and the include code was written as follows:-

```
*&---------------------------------------------------------------------*
*&  Include           ZXVSVU04                                         *
*&---------------------------------------------------------------------*
data : lfa1m like E1LFA1M ,
       zlfa1 like ZVEND ,
       l_cnt type i   .


data : parent like lfa1-parent ,
       rating like lfa1-rating ,
       ind type sy-tabix.

       data : x_val(20) type c .

 tables : ytodel.
```

```
case idoc_control-direct.
* When Inbound
when '2'.
* For vendor master only
 check idoc_control-mestyp = 'CREMAS'.


 loop at idoc_data .
  case idoc_data-segnam.
   when 'E1LFA1M'.
    move idoc_data-sdata to lfa1m.
    clear l_cnt.
* From the parent segment, get the vendor number and check whether it
* exists in the database or not
    select count(*) into l_cnt
           from lfa1
           where lifnr = lfa1m-lifnr .
   when 'ZVEND'.
* For the child segment, if the vendor exists, update the reference, rating
* and mobile number
    move idoc_data-sdata to zlfa1.
     if l_cnt gt 0.
      UPDATE lfa1
      set perref = zlfa1-perref
          ratings = zlfa1-ratings
          mobile  = zlfa1-mobile
        where lifnr = lfa1m-lifnr.
        commit work.
      commit work.
     endif.

  endcase.
 endloop.
 endcase.
```

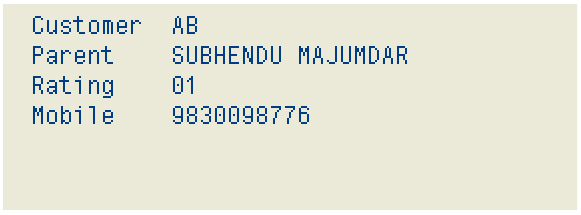
7.3.11. Step 12 : Test the whole Connection

7.3.11.1. Run program : YVENDOR_MODIFY in sender system from SE38 to update information for vendor AB.



Enter vendor	ab
Enter Reference	Alakesh Ray
Enter Ratings	2
Enter Cell no	9830098776

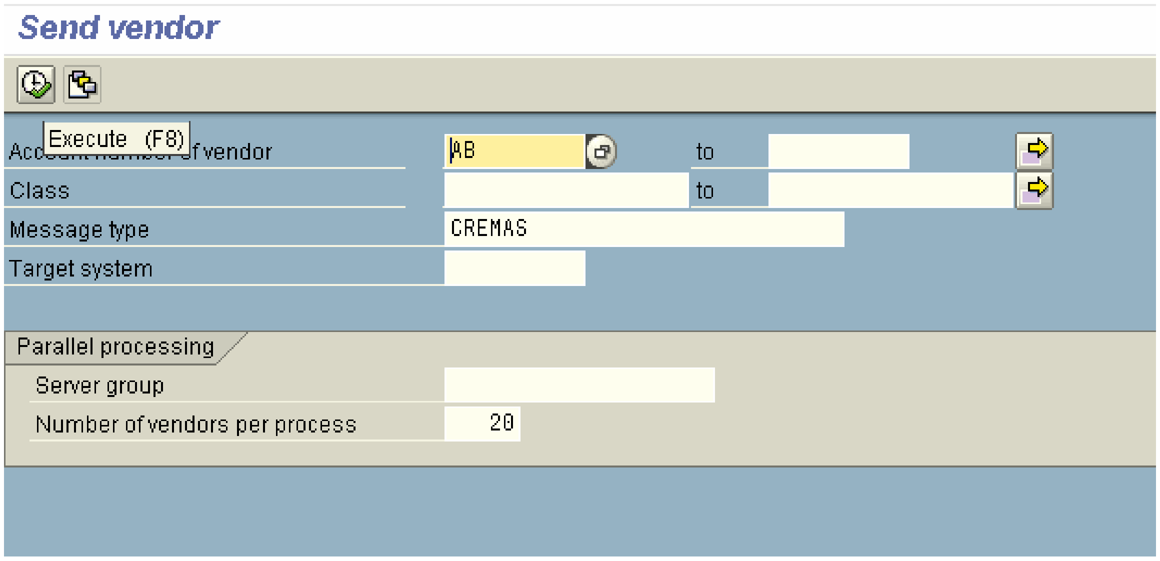
[✓] Check to update the database



```
Customer  AB
Parent    SUBHENDU MAJUMDAR
Rating    01
Mobile    9830098776
```

The report will ensure that updation is successful.

7.3.11.2. Distribute the vendor using BD14 in Sender system

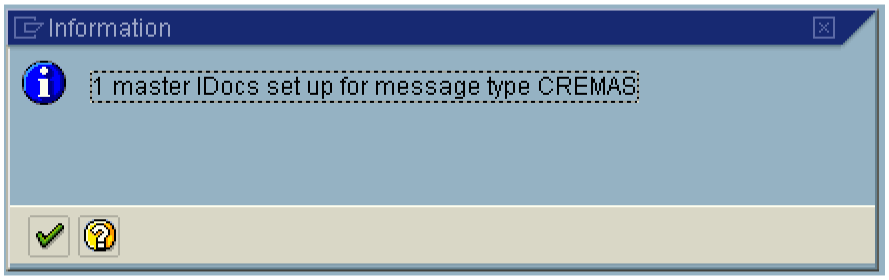

Send vendor

Execute (F8)

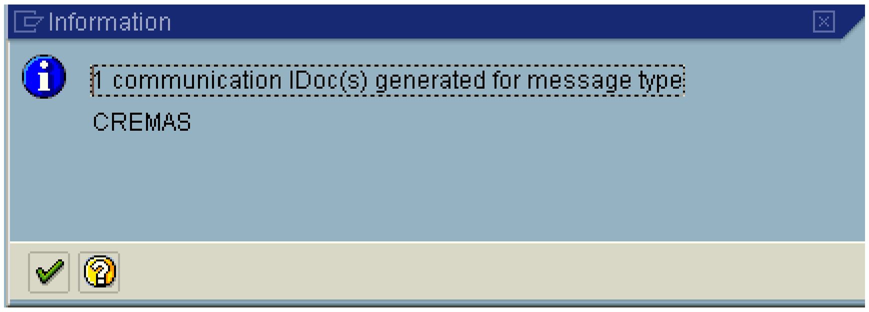
Account number of vendor	AB	to	
Class		to	
Message type	CREMAS		
Target system			

Parallel processing

Server group	
Number of vendors per process	20

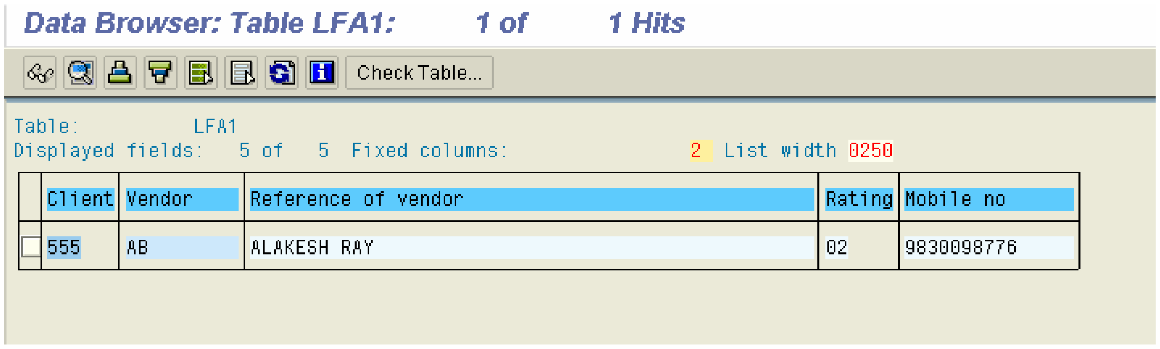

Information

1 master IDocs set up for message type CREMAS


Information

1 communication IDoc(s) generated for message type CREMAS

7.3.11.3. See the information for the vendor in receiver system



```
Table:           LFA1
Displayed fields:   5 of   5  Fixed columns:              2  List width 0250
```

Client	Vendor	Reference of vendor	Rating	Mobile no
555	AB	ALAKESH RAY	02	9830098776

So, your job is successful.

8. Configurations and Programmings to Maintain Change Documents for new information

Case

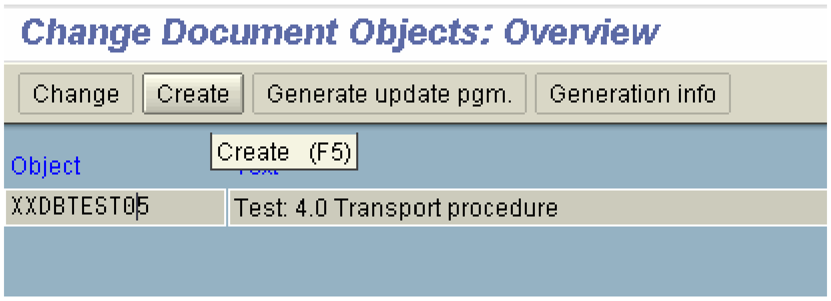
Table ZEMP_MAST in SAP contains information about employee id and name .
The requirement is that, any new update to this table will create change documents in SAP database, which can be used in future for audit trial or for using change pointer techniques.

8.1. Steps to be performed

To achieve the goal, following steps are to be performed:-

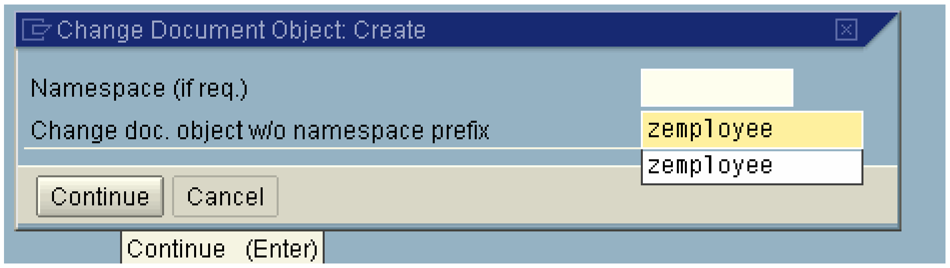
1. Create Change Document Object using transaction SCDO.
2. Generate Function module and includes for creating change document.
3. Write a Program using the program objects generated in the step above to create change document in the database.

8.1.1. Step 1 – Create Change Document Object using transaction SCDO.



XXDBTEST05 Test: 4.0 Transport procedure

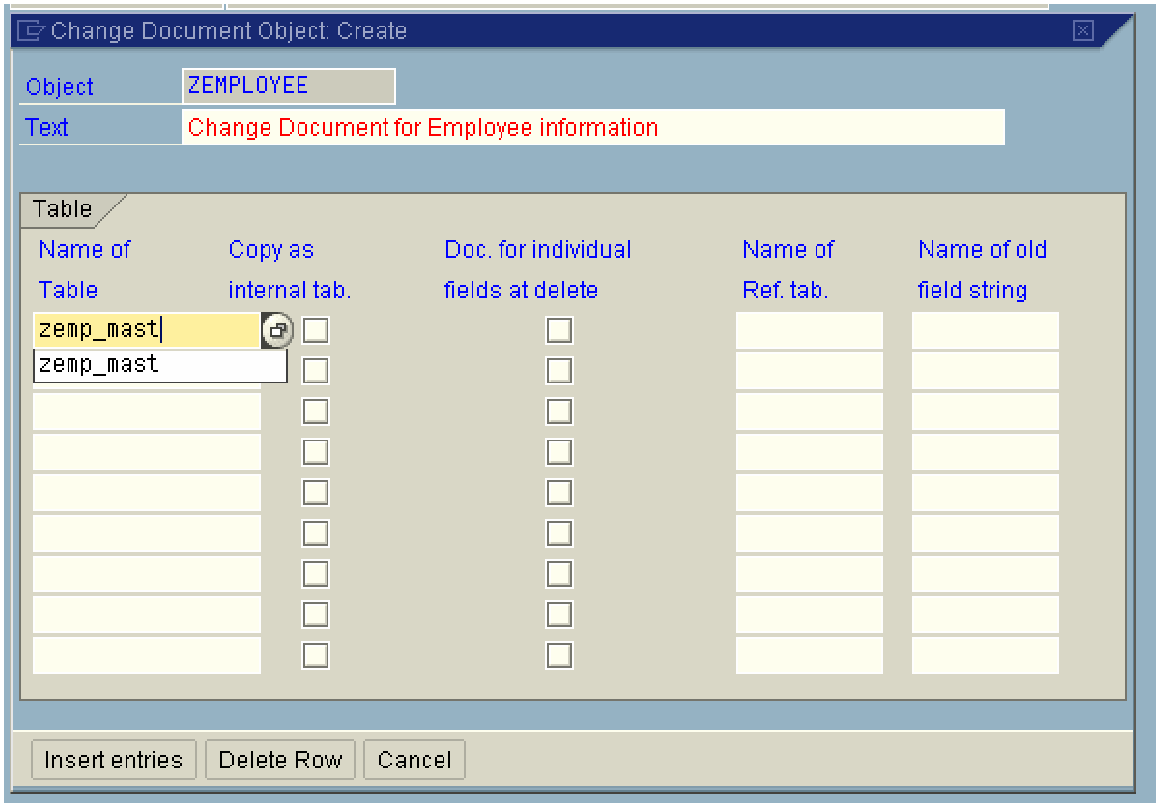
Go to transaction SCDO. Press **Create** from application toolbar.



Namespace (if req.)
Change doc. object w/o namespace prefix zemployee

zemployee

Enter a new name for the change document object. Choose : **Continue**.

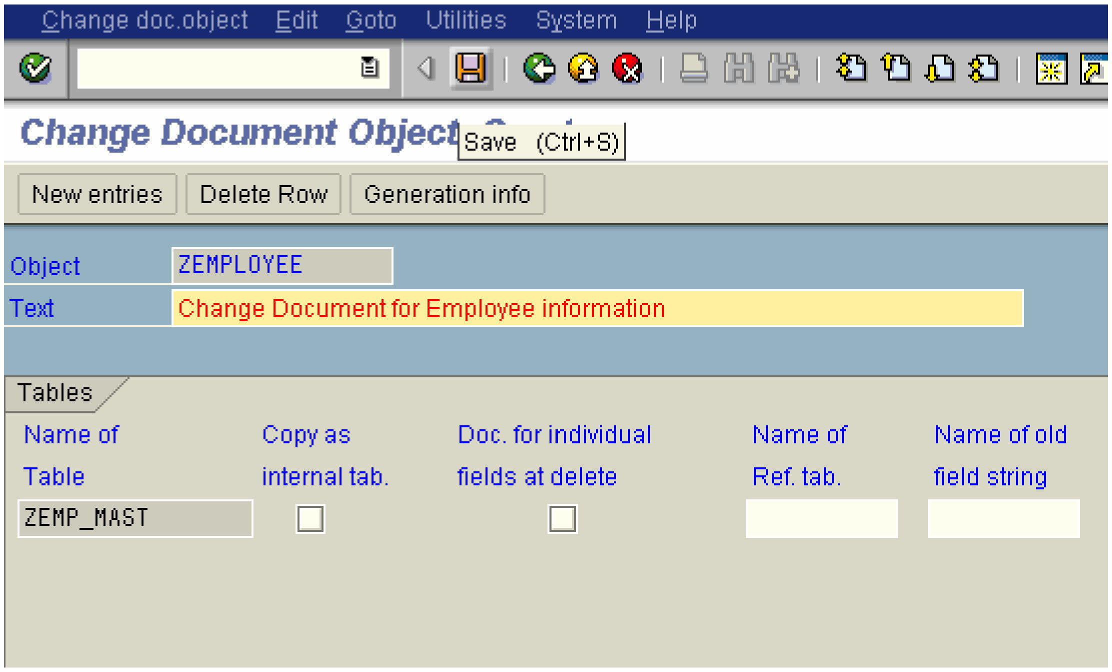


Object	ZEMPLOYEE
Text	Change Document for Employee information

Table

Name of Table	Copy as internal tab.	Doc. for individual fields at delete	Name of Ref. tab.	Name of old field string
zemp_mast	☐	☐		
zemp_mast	☐	☐		

Enter a descriptive text for change document object and list the tables which will lie under it. Then lick pushbutton : **Insert Entries**.



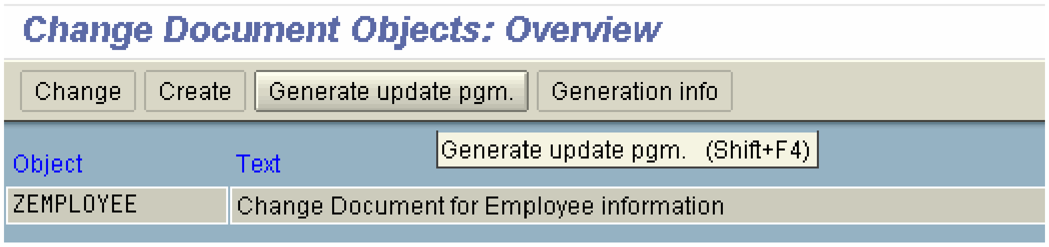
Object	ZEMPLOYEE
Text	Change Document for Employee information

Tables

Name of Table	Copy as internal tab.	Doc. for individual fields at delete	Name of Ref. tab.	Name of old field string
ZEMP_MAST	☐	☐		

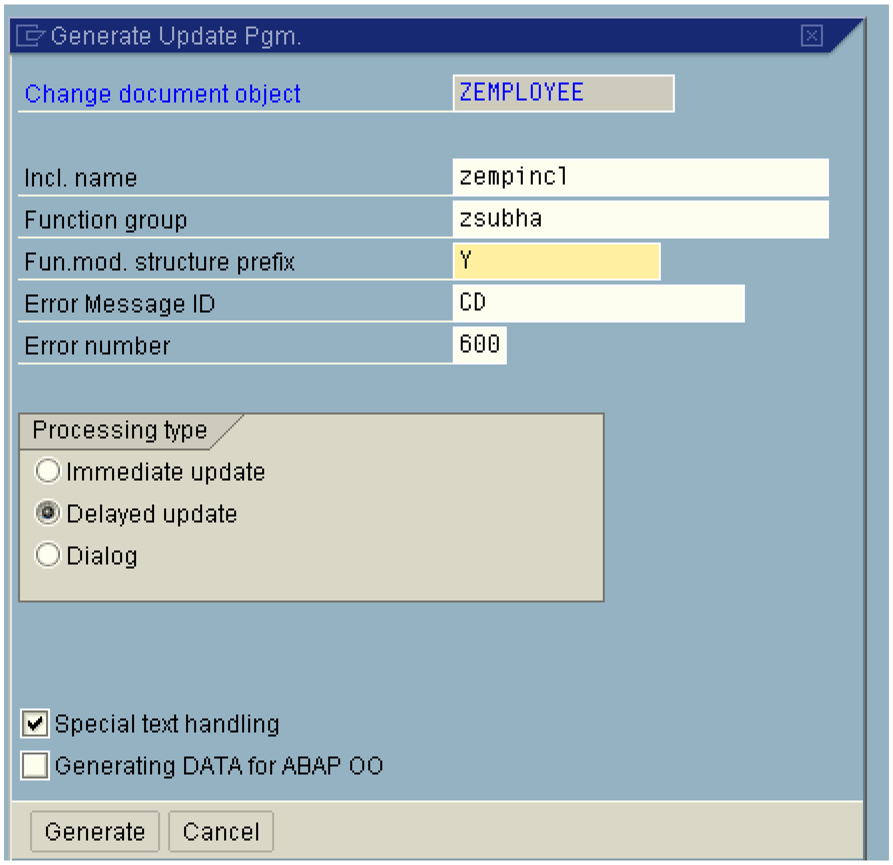
Save the entries.

8.1.2. Step 2 – Generate Programs and Includes


Change Document Objects: Overview

Object	Text
ZEMPLOYEE	Change Document for Employee information

Come to initial screen of SCDO. Place your cursor on the change object you have created and click the pushbutton shown above to generate programs.


Generate Update Pgm.

Change document object	ZEMPLOYEE
Incl. name	zempincl
Function group	zsubha
Fun.mod. structure prefix	Y
Error Message ID	CD
Error number	600

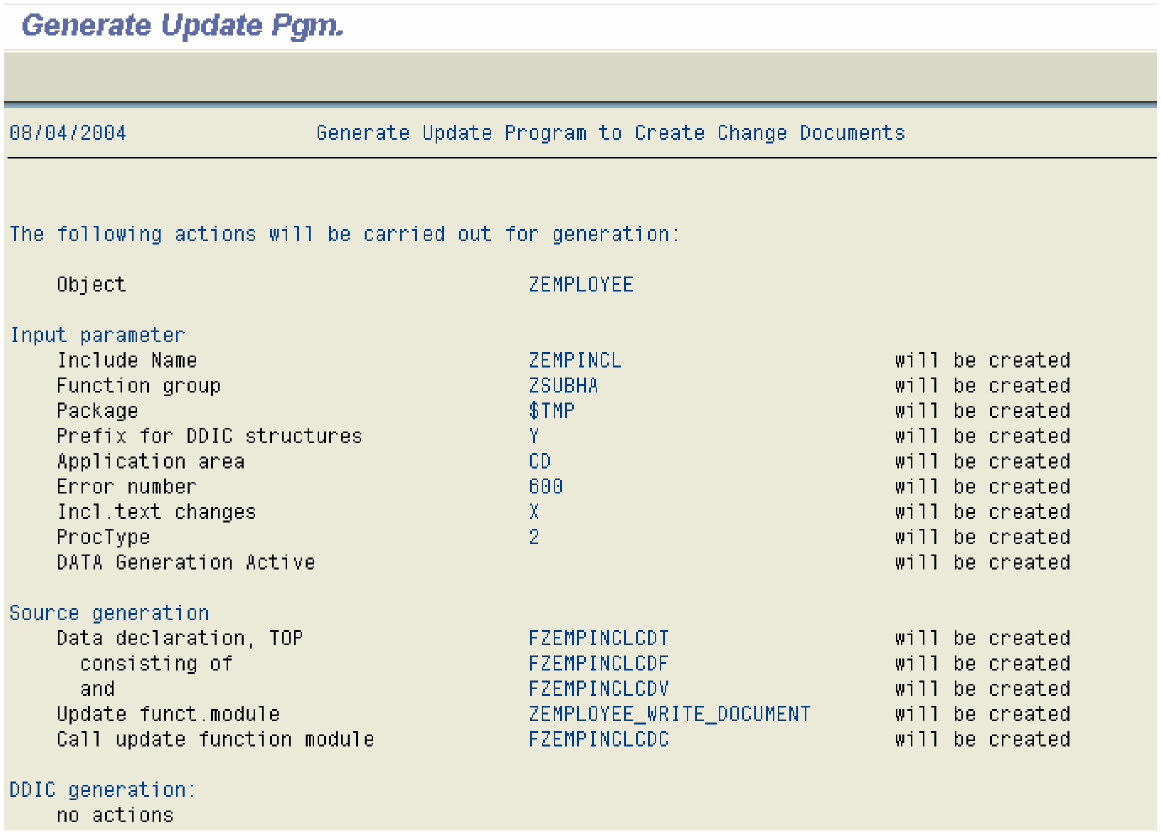
Processing type
- ○ Immediate update
- ● Delayed update
- ○ Dialog

- ☑ Special text handling
- ☐ Generating DATA for ABAP OO

Specify the following:-

- ❖ Prefixes for the includes to be generated.
- ❖ Function group which will contain the function module, which will create change pointer in the database.
- ❖ Prefixes for the structures which will be created in the database and used by the function module.
- ❖ Error message id and number to flash error message in case of an error.

Press Enter.



```
08/04/2004                 Generate Update Program to Create Change Documents

The following actions will be carried out for generation:

    Object                                ZEMPLOYEE

Input parameter
    Include Name                          ZEMPINCL                       will be created
    Function group                        ZSUBHA                         will be created
    Package                               $TMP                           will be created
    Prefix for DDIC structures            Y                              will be created
    Application area                      CD                             will be created
    Error number                          600                            will be created
    Incl.text changes                     X                              will be created
    ProcType                              2                              will be created
    DATA Generation Active                                               will be created

Source generation
    Data declaration, TOP                 FZEMPINCLCDT                   will be created
      consisting of                       FZEMPINCLCDF                   will be created
      and                                 FZEMPINCLCDV                   will be created
    Update funct.module                   ZEMPLOYEE_WRITE_DOCUMENT       will be created
    Call update function module           FZEMPINCLCDC                   will be created

DDIC generation:
    no actions
```

A pre-action report will be displayed . This shows the following:-

Function module ZEMPLOYEE_WRITE_DOCUMENT will be created under function group ZSUBHA.

This function module will create change document in the database for change document object ZEMPLOYEE.

An include program FZEMPINCLCDC will be created which will contain a call to the function module.

Includes FZEMPINCLCDF and FZEMPINCLCDV will contain data declaration for the variables which will be used as interface parameters to the function module.

Press Save. The objects will be created and the report will be modified , informing you that the objects are created.

8.1.3. Step 3 – Write a program / modify existing program to call FM to write change documents

Now, you have to modify the program which is used to update table ZEMP_MAST .

In the global section of the program, include program FZEMPINCLCDT and FZEMPINCLCDC. The first one(FZEMPINCLCDT) contains another two includes :-
FZEMPINCLCDF
FZEMPINCLCDV

Include FZEMPINCLCDC contains a call to the function module ZEMPLOYEE_WRITE_DOCUMENT.

Now, in the appropriate section of the code, after updating table ZEMP_MAST, you need to give a call to the function module by calling subroutine : CD_CALL_ZEMPLOYEE . This subroutine originally belongs to the include FZEMPINCLCDC and have a call to the function module.

Before calling this subroutine, you have to populate all the interface parameters of the function module.

The following program is a demo to this idea. It contains two parameters in the selection-screen , one for employee id and another for employee name .
New employee ids are inserted and existing ones have the employee names updated.

This program creates change documents in CDHDR and CDPOS table.

```
*&---------------------------------------------------------------------*
*& Report  YSUBDEL                                                     *
*&                                                                     *
*&---------------------------------------------------------------------*
*&                                                                     *
*&                                                                     *
*&---------------------------------------------------------------------*

REPORT  YSUBDEL123     .


include fzempinclcdt.
include fzempinclcdc.


data : x_mast like zemp_mast.

data : x_flag(1) type c ,
       x_name like zemp_mast-empname .

parameters : p_empid   like zemp_mast-empid obligatory,
             p_name like zemp_mast-empname .
```

```
 initialization.
   perform sub_clear_variables.



 at selection-screen.
  perform sub_flag_determine.


 start-of-selection.

   if x_flag = 'I'.
     perform sub_insertion_operation.
   else.
    perform sub_updation_operation.
   endif.
*&---------------------------------------------------------------------*
*&      Form  sub_clear_variables
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
form sub_clear_variables .
      clear : x_mast ,
            p_empid ,
            p_name ,
            x_flag ,
            x_name .
* Populate interface parameters for the function module.
   objectid = 'ZEMPLOYEE'.
   tcode = 'SE38'.
   utime = sy-uzeit.
   udate = sy-datum .
   username = sy-uname.
endform.                    " sub_clear_variables
*&---------------------------------------------------------------------*
*&      Form  sub_flag_determine
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
form sub_flag_determine .
    select single empname into x_name
   from zemp_mast
   where empid = p_empid.
   if sy-subrc ne 0.
    x_flag = 'I'.
   else.
    x_flag = 'U'.
    if p_name is initial.
     p_name = x_name.
    endif.
   endif.
endform.                    " sub_flag_determine
*&---------------------------------------------------------------------*
*&      Form  sub_insertion_operation
*&---------------------------------------------------------------------*
```

```
*      text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
form sub_insertion_operation .
  upd_zemp_mast = 'X'.
    x_mast-empid = p_empid.
    x_mast-empname = p_name.
    insert zemp_mast from x_mast.
    if sy-subrc eq 0.
     message i398(00) with 'Insertion successful'.
* Populate interface parameters for the function module.
     CDOC_UPD_OBJECT = 'I'.
     UPD_ZEMP_MAST = 'I'.
      zemp_mast-mandt = sy-mandt .
      zemp_mast-empid = p_empid .
      zemp_mast-empname = p_name .
* Call to the function module to create change pointers in the database.
     perform cd_call_zemployee.
     commit work.
    endif.
endform.                    " sub_insertion_operation
*&---------------------------------------------------------------------*
*&      Form  sub_updation_operation
*&---------------------------------------------------------------------*
*      text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
form sub_updation_operation .
* Populate interface parameters for the function module.
      upd_zemp_mast = 'X'.
    update zemp_mast
     set empname = p_name
     where empid = p_empid.

     if sy-subrc eq 0.
     message i398(00) with 'Updation successful'.
* Populate interface parameters for the function module.
     CDOC_UPD_OBJECT = 'U'.
     UPD_ZEMP_MAST = 'U'.
* *zemp_mast contains the old values and zemp_mast contains new values.
     *zemp_mast-mandt = sy-mandt .
     *zemp_mast-empid = p_empid .
     *zemp_mast-empname = x_name.


     zemp_mast-mandt = sy-mandt .
     zemp_mast-empid = p_empid .
     zemp_mast-empname = p_name .
* Call to the function module to create change pointers in the database.
      perform cd_call_zemployee.
    endif.
endform.                    " sub_updation_operation
```

Include FZEMPINCLCDT contains another two includes which have global data declarations.

```
INCLUDE FZEMPINCLCDF                          .
INCLUDE FZEMPINCLCDV                        .
```

The source code for FZEMPINCLCDF is as follows:-

```
DATA: OBJECTID              TYPE CDHDR-OBJECTID,
      TCODE                 TYPE CDHDR-TCODE,
      PLANNED_CHANGE_NUMBER   TYPE CDHDR-PLANCHNGNR,
      UTIME                 TYPE CDHDR-UTIME,
      UDATE                 TYPE CDHDR-UDATE,
      USERNAME              TYPE CDHDR-USERNAME,
      CDOC_PLANNED_OR_REAL    TYPE CDHDR-CHANGE_IND,
      CDOC_UPD_OBJECT         TYPE CDHDR-CHANGE_IND VALUE 'U',
      CDOC_NO_CHANGE_POINTERS TYPE CDHDR-CHANGE_IND.
```

The source code for FZEMPINCLCDV is as follows:-

```
* declaration for the long text
DATA: BEGIN OF ICDTXT_ZEMPLOYEE       OCCURS 20.
        INCLUDE STRUCTURE CDTXT.
DATA: END OF ICDTXT_ZEMPLOYEE      .

DATA: UPD_ICDTXT_ZEMPLOYEE       TYPE C.

TABLES: *ZEMP_MAST
       , ZEMP_MAST                  .
DATA: UPD_ZEMP_MAST                    TYPE C.
```

Source code for FZEMPINCLCDC contains call to the function module.

```
FORM CD_CALL_ZEMPLOYEE                    .
  IF   ( UPD_ZEMP_MAST                    NE SPACE )
   OR ( UPD_ICDTXT_ZEMPLOYEE       NE SPACE )
  .
   CALL FUNCTION 'ZEMPLOYEE_WRITE_DOCUMENT      ' IN UPDATE TASK
      EXPORTING
        OBJECTID                = OBJECTID
        TCODE                   = TCODE
        UTIME                   = UTIME
        UDATE                   = UDATE
        USERNAME                = USERNAME
        PLANNED_CHANGE_NUMBER   = PLANNED_CHANGE_NUMBER
        OBJECT_CHANGE_INDICATOR = CDOC_UPD_OBJECT
        PLANNED_OR_REAL_CHANGES = CDOC_PLANNED_OR_REAL
        NO_CHANGE_POINTERS      = CDOC_NO_CHANGE_POINTERS
        O_ZEMP_MAST
                   = *ZEMP_MAST
        N_ZEMP_MAST
                   = ZEMP_MAST
        UPD_ZEMP_MAST
                   = UPD_ZEMP_MAST
        UPD_ICDTXT_ZEMPLOYEE
                   = UPD_ICDTXT_ZEMPLOYEE
      TABLES
        ICDTXT_ZEMPLOYEE
                   = ICDTXT_ZEMPLOYEE
   .
  ENDIF.
  CLEAR PLANNED_CHANGE_NUMBER.
ENDFORM.
```

The source code for the function module is as follows:-

```
FUNCTION ZEMPLOYEE_WRITE_DOCUMENT     .

  CALL FUNCTION 'CHANGEDOCUMENT_OPEN'
    EXPORTING
      OBJECTCLASS             = 'ZEMPLOYEE      '
      OBJECTID                = OBJECTID
      PLANNED_CHANGE_NUMBER   = PLANNED_CHANGE_NUMBER
      PLANNED_OR_REAL_CHANGES = PLANNED_OR_REAL_CHANGES
    EXCEPTIONS
      SEQUENCE_INVALID        = 1
      OTHERS                  = 2.

  CASE SY-SUBRC.
    WHEN 0.                                   "ok.
    WHEN 1. MESSAGE A600 WITH 'SEQUENCE INVALID'.
    WHEN 2. MESSAGE A600 WITH 'OPEN ERROR'.
  ENDCASE.


IF UPD_ZEMP_MAST                     NE SPACE.
   CALL FUNCTION 'CHANGEDOCUMENT_SINGLE_CASE'
     EXPORTING
       TABLENAME              = 'ZEMP_MAST                     '
       WORKAREA_OLD           = O_ZEMP_MAST
       WORKAREA_NEW           = N_ZEMP_MAST
        CHANGE_INDICATOR       = UPD_ZEMP_MAST
        DOCU_DELETE            = ' '
     EXCEPTIONS
       NAMETAB_ERROR          = 1
       OPEN_MISSING           = 2
       POSITION_INSERT_FAILED = 3
       OTHERS                 = 4.

    CASE SY-SUBRC.
      WHEN 0.                                "ok.
      WHEN 1. MESSAGE A600 WITH 'NAMETAB-ERROR'.
      WHEN 2. MESSAGE A600 WITH 'OPEN MISSING'.
      WHEN 3. MESSAGE A600 WITH 'INSERT ERROR'.
      WHEN 4. MESSAGE A600 WITH 'SINGLE ERROR'.
    ENDCASE.
  ENDIF.

  IF UPD_ICDTXT_ZEMPLOYEE       NE SPACE.
     CALL FUNCTION 'CHANGEDOCUMENT_TEXT_CASE'
       TABLES
         TEXTTABLE              = ICDTXT_ZEMPLOYEE
       EXCEPTIONS
         OPEN_MISSING           = 1
         POSITION_INSERT_FAILED = 2
         OTHERS                 = 3.

    CASE SY-SUBRC.
      WHEN 0.                                "ok.
      WHEN 1. MESSAGE A600 WITH 'OPEN MISSING'.
      WHEN 2. MESSAGE A600 WITH 'INSERT ERROR'.
      WHEN 3. MESSAGE A600 WITH 'TEXT ERROR'.
    ENDCASE.
  ENDIF.
  CALL FUNCTION 'CHANGEDOCUMENT_CLOSE'
    EXPORTING
      OBJECTCLASS             = 'ZEMPLOYEE      '
      OBJECTID                = OBJECTID
      DATE_OF_CHANGE          = UDATE
      TIME_OF_CHANGE          = UTIME
      TCODE                   = TCODE
      USERNAME                = USERNAME
      OBJECT_CHANGE_INDICATOR = OBJECT_CHANGE_INDICATOR
      NO_CHANGE_POINTERS      = NO_CHANGE_POINTERS
    EXCEPTIONS
      HEADER_INSERT_FAILED    = 1
      OBJECT_INVALID          = 2
      OPEN_MISSING            = 3
      NO_POSITION_INSERTED    = 4
      OTHERS                  = 5.

  CASE SY-SUBRC.
    WHEN 0.                                   "ok.
    WHEN 1. MESSAGE A600 WITH 'INSERT HEADER FAILED'.
    WHEN 2. MESSAGE A600 WITH 'OBJECT INVALID'.
    WHEN 3. MESSAGE A600 WITH 'OPEN MISSING'.
*    WHEN 4. MESSAGE A600 WITH 'NO_POSITION_INSERTED'.
* do not abort, if positions are not inserted!!!
    WHEN 5. MESSAGE A600 WITH 'CLOSE ERROR'.
  ENDCASE.

ENDFUNCTION.
```

9. Configuring and Developing for Change Pointers for a custom message type

<u>Mission</u>

Table ZEMP_MAST and ZEMP_QUAL is maintained by Shatadru, client 777 and this information is transferred to receiver system , Shatadru,555 by IDoc using Ale service. Adequate configurations and settings exist for that.

Now, the demand is that, any new entry/updation to table ZEMP_MAST using transaction ZEMPMR (report program to update ZEMP_MAST table, creates change documents in database) is done, and then change pointer technique will send IDoc to receiver system (client 555, Shatadru).

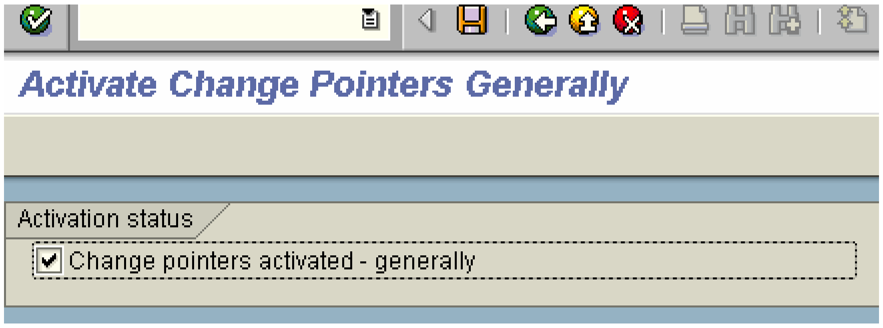
9.1. Assumptions

1. Adequate settings already exist for normal Idoc flow between two systems.
2. Change documents are created in database by ZEMPMR transaction.

9.2. Things to do

1. Activate Change Pointers globally in sender system.
2. Enable change pointers for the message type, ZEMPMESSAGE in sender system.
3. Specify fields for which the change pointers are to be written in sender system for the change document object.
4. Develop a function module that will read change pointers and then create master Idoc and distribute it in the ALE layer. In this way, IDoc will be transferred from sender to receiver system.

9.2.1. Activate Change Pointers Globally in Sender System(BD61)

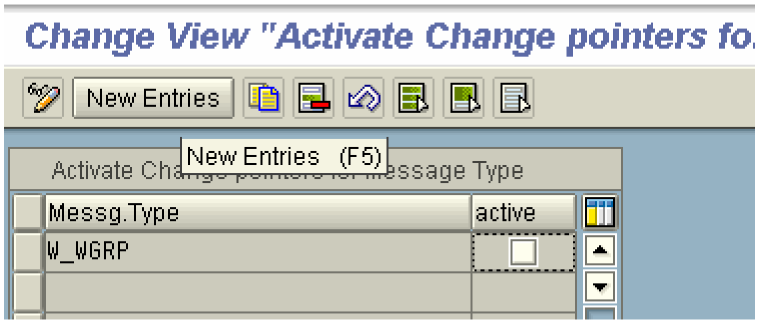

Activate Change Pointers Generally

Activation status

☑ Change pointers activated - generally

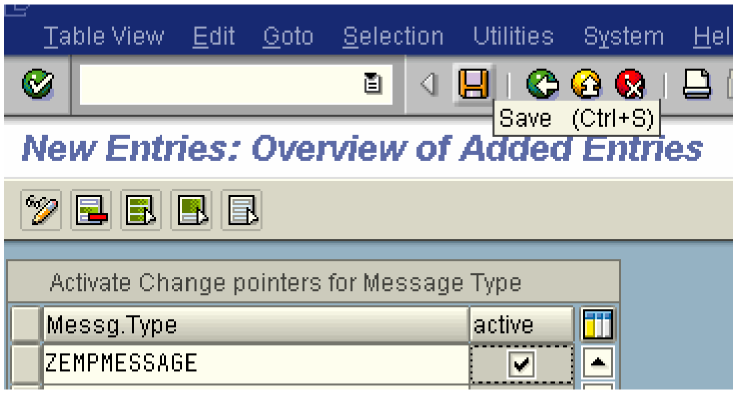
Record is maintained in table TBDA1.

9.2.2. Enable change pointers for a message type in sender system(BD50)


Change View "Activate Change pointers fo

Activate Change pointers for Message Type

Messg.Type	active
W_WGRP	☐



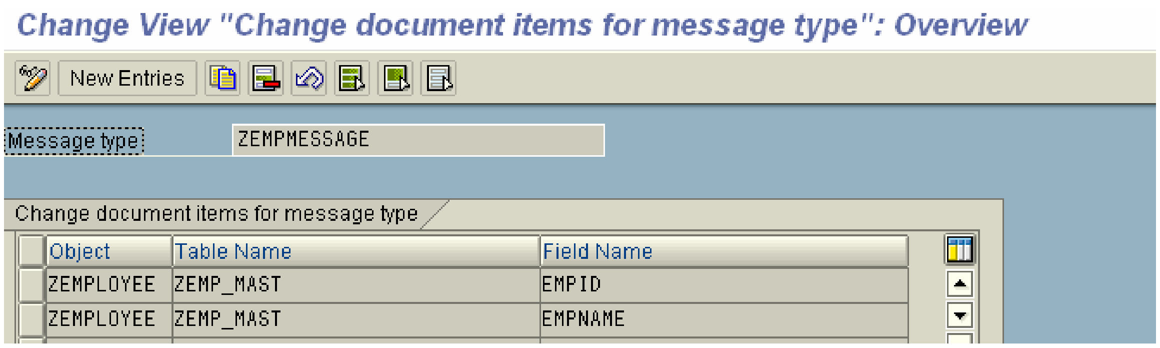
New Entries: Overview of Added Entries

Activate Change pointers for Message Type

Messg.Type	active
ZEMPMESSAGE	☑

Record is maintained in table **TBDA2.**

9.2.3. Specify Fields for which Change Pointers are to be written (BD52) in Sender System


Change View "Change document items for message type": Overview

Message type: ZEMPMESSAGE

Change document items for message type

Object	Table Name	Field Name
ZEMPLOYEE	ZEMP_MAST	EMPID
ZEMPLOYEE	ZEMP_MAST	EMPNAME

9.2.4. Develop a Function module for Sending Idocs(SE37)

You need to develop a function module in the sender system , which will read information on change documents and accordingly create master IDOCs and distribute it in ALE layer to transfer the IDoc to receiver system.

```
FUNCTION ZMASTERIDOC_CREATE_SMD_ZEMP.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  IMPORTING
*"     REFERENCE(MESSAGE_TYPE) LIKE  TBDME-MESTYP
*"----------------------------------------------------------------------


DATA : empid                like zemp_mast-empid ,
       created_c_idocs      like sy-tabix ,
       created_m_idocs      like sy-tabix ,
       created_comm_idocs   like sy-tabix ,
       done_since_commit    like sy-tabix ,
       c_mark(1)            type c value 'X' ,
       c_idocs_before_commit like sy-tabix value 50 .



 data : T_CHGPTRS LIKE STANDARD TABLE OF BDCP INITIAL SIZE 0 WITH HEADER LINE ,

       BEGIN OF T_CHGPTRS_EMP OCCURS 0 ,
        empid LIKE zemp_mast-empid ,
        cpident like bdcp-cpident ,
       END OF t_chgptrs_emp ,

       BEGIN OF T_CPIDENT OCCURS 0 ,
        cpident LIKE bdcp-cpident ,
       END OF T_CPIDENT .

* Step 1 : Scan database to find any change pointer information for the message type
       CALL FUNCTION 'CHANGE_POINTERS_READ'
```

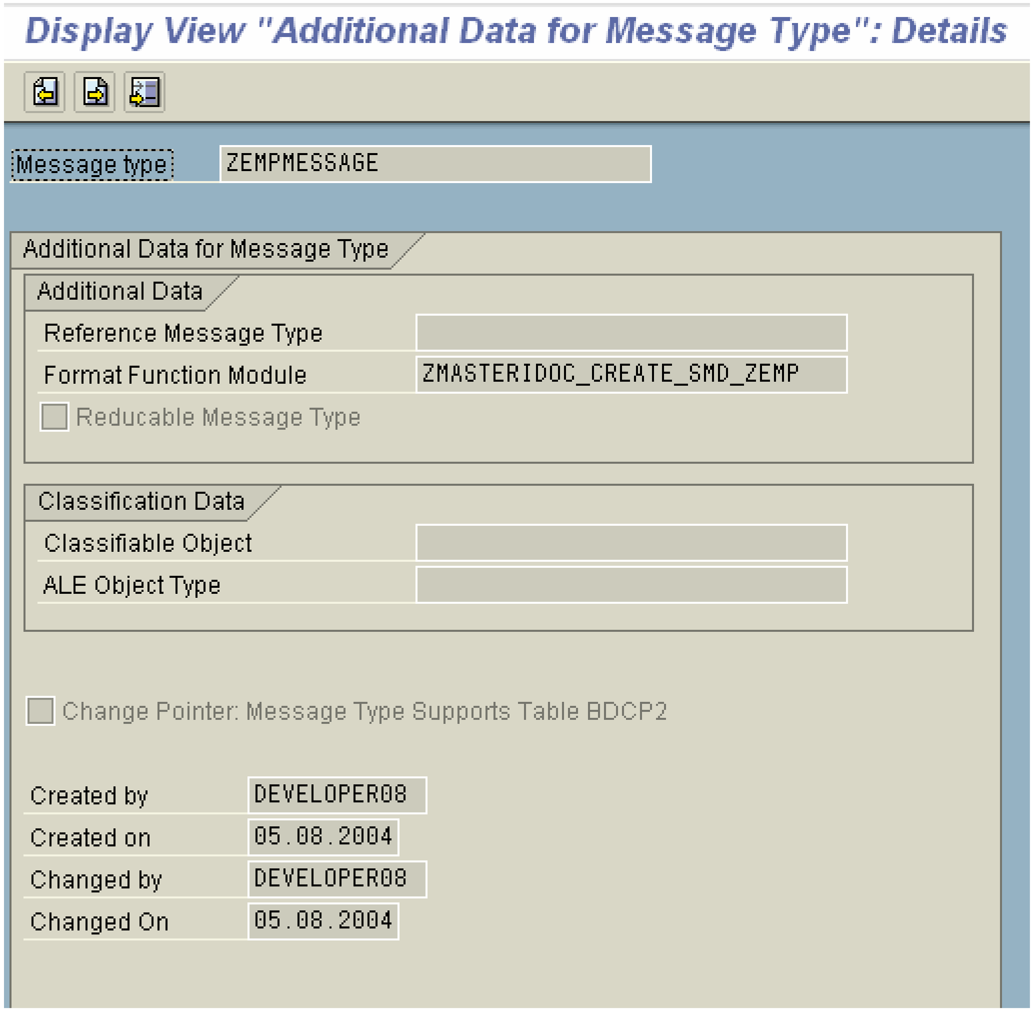
```
        EXPORTING
          message_type                    = message_type
          READ_NOT_PROCESSED_POINTERS       = 'X'
        tables
          change_pointers                 = t_chgptrs
*         MESSAGE_TYPES                     =
       EXCEPTIONS
         ERROR_IN_DATE_INTERVAL           = 1
         ERROR_IN_TIME_INTERVAL           = 2
         OTHERS                           = 3
                .
      IF sy-subrc <> 0.
        MESSAGE i398(00) with 'Error in reading change pointers'.
        EXIT.
      ENDIF.

     if t_chgptrs[] is initial.
      message i398(00) with 'No change documents detected for ' message_type.
      exit.
     endif.

     clear : created_c_idocs ,
             created_m_idocs ,
             done_since_commit .
* Prepare internal table t_chgptrs_emp with employee and change pointer no info
     LOOP AT t_chgptrs.
      shift t_chgptrs-tabkey left deleting leading space.
      t_chgptrs_emp-empid = t_chgptrs-tabkey+3.
      t_chgptrs_emp-cpident = t_chgptrs-cpident.
      append t_chgptrs_emp.
     ENDLOOP.

     sort t_chgptrs_emp by empid.

     clear empid.

     LOOP AT t_chgptrs_emp .
* Duplicate information on the same employee will not create any more IDoc
       if t_chgptrs_emp-empid eq empid .
        t_cpident-cpident = t_chgptrs_emp-cpident.
        append t_cpident.
        continue.
       endif.

       empid = t_chgptrs_emp-empid.
* Create IDoc and distribute
       CALL FUNCTION 'ZMASTERIDOC_CREATE_ZEMP'
         EXPORTING
           empid                  = empid
           message_type           = message_type
        IMPORTING
           CREATED_COMM_IDOCS       = created_comm_idocs
                 .

       created_m_idocs = created_m_idocs + 1.
       created_c_idocs = created_c_idocs + created_comm_idocs.
       done_since_commit = done_since_commit + 1.
       t_cpident-cpident = t_chgptrs_emp-cpident .
       append t_cpident.

       if done_since_commit ge 50.
        done_since_commit = 0.
```

```
* Change the status of the change pointers , once they are processed
         CALL FUNCTION 'CHANGE_POINTERS_STATUS_WRITE'
           EXPORTING
             message_type                = message_type
           tables
             change_pointers_idents      = t_cpident
                   .
           refresh : t_cpident.
           commit work.
           CALL FUNCTION 'DEQUEUE_ALL'
*             EXPORTING
*               _SYNCHRON       = ' '
                     .
       endif.
      endloop.

       if done_since_commit gt 0.
         CALL FUNCTION 'CHANGE_POINTERS_STATUS_WRITE'
           EXPORTING
             message_type                = message_type
           tables
             change_pointers_idents      = t_cpident
                   .
           refresh : t_cpident.
           commit work.
           CALL FUNCTION 'DEQUEUE_ALL'
*             EXPORTING
*               _SYNCHRON       = ' '
                     .
       endif.

       message i398(00) with 'For ' message_type 'Master IDoc created = ' created_m_idocs.
       message i398(00) with 'For ' message_type 'Communication IDoc created = ' created_c_idocs.


ENDFUNCTION.
```

The function module , ZMASTERIDOC_CREATE_ZEMP, used o create IDoc and distribute in the Ale layer(so that IDoc can be transferred from sender to receiver) is coded as follows:-

```
FUNCTION ZMASTERIDOC_CREATE_ZEMP.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  IMPORTING
*"     VALUE(EMPID) LIKE  ZEMP_MAST-EMPID
*"     VALUE(MESSAGE_TYPE) LIKE  TBDME-MESTYP
*"  EXPORTING
*"     VALUE(CREATED_COMM_IDOCS) LIKE  SY-TABIX
*"----------------------------------------------------------------------

DATA : control_record_out like edidc ,
       x_hdr  like Z1EHDR ,
       x_qual like Z1QUAL .

data : x_mast  like zemp_mast.
data : x_empqual like zemp_qual.

 data : it_qual       like standard table of zemp_qual initial size 0 with header line ,
        it_edidd      like standard table of edidd     initial size 0 with header line ,
        it_comm_idocs like standard table of edidc     initial size 0 with header line .


       SELECT SINGLE * FROM zemp_mast into x_mast
              WHERE empid = empid .
       IF sy-subrc ne 0.
        MESSAGE I398(00) WITH 'Information on employee' empid 'not found'.
        EXIT.
       ENDIF.


       SELECT * FROM ZEMP_QUAL INTO TABLE it_qual
               WHERE empid = empid.

       control_record_out-mestyp = message_type.
       control_record_out-doctyp = 'ZEMPIDOC'.

       x_hdr-empid = x_mast-empid.
       x_hdr-empname = x_mast-empname.

       it_edidd-segnam = 'Z1EHDR'.
       it_edidd-sdata = x_hdr .
       append it_edidd.

       if not it_qual[] is initial.
       LOOP AT it_qual.
        x_qual-pyear = it_qual-pyear.
        x_qual-qual = it_qual-qual.
        it_edidd-segnam = 'Z1QUAL'.
        it_edidd-sdata = x_qual.
        append it_edidd.
       ENDLOOP.
      endif.

       CALL FUNCTION 'MASTER_IDOC_DISTRIBUTE'
         EXPORTING
           master_idoc_control                = control_record_out
*          OBJ_TYPE                         = ''
*          CHNUM                          = ''
         tables
           communication_idoc_control          = it_comm_idocs
           master_idoc_data                   = it_edidd
        EXCEPTIONS
          ERROR_IN_IDOC_CONTROL              = 1
          ERROR_WRITING_IDOC_STATUS          = 2
          ERROR_IN_IDOC_DATA                 = 3
          SENDING_LOGICAL_SYSTEM_UNKNOWN      = 4
          OTHERS                             = 5
                 .
       IF sy-subrc <> 0.
 MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
        WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
       ENDIF.

  loop at it_comm_idocs.
   message i398(00) with 'IDoc' it_comm_idocs-docnum ' created in the database'.
 endloop.

  describe table it_comm_idocs lines created_comm_idocs.
  describe table it_comm_idocs lines created_comm_idocs.


ENDFUNCTION.
```

9.2.5. Link Message Type to Function Module in Sender System (BD60)



Display View "Additional Data for Message Type": Details

Message type	ZEMPMESSAGE

Additional Data for Message Type

Additional Data

Reference Message Type	
Format Function Module	ZMASTERIDOC_CREATE_SMD_ZEMP

☐ Reducable Message Type

Classification Data

Classifiable Object	
ALE Object Type	

☐ Change Pointer: Message Type Supports Table BDCP2

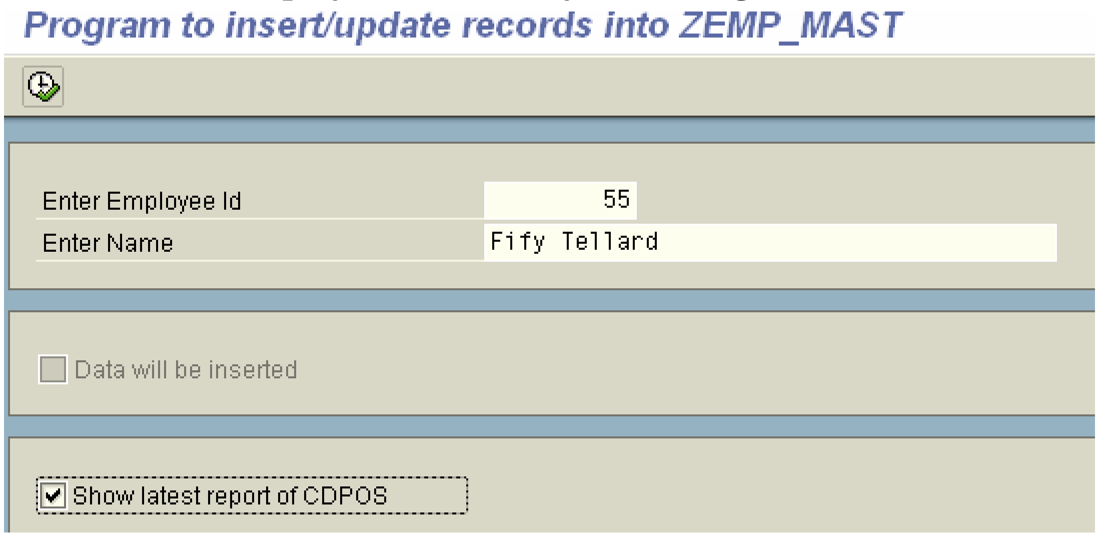
Created by	DEVELOPER08
Created on	05.08.2004
Changed by	DEVELOPER08
Changed On	05.08.2004

9.2.6. Testing

Now, all the configurations and developments are over. We need to test one scenario to ensure the satisfaction of the requirements.

Sender	Shatadru, client 777
Receiver	Shatadru, client 555
Message shared	zempmessage
IDoc	ZEMPIDOC contains information on employee master and qualifications
Change pointer recorded on	ZEMP_MAST (on fields empid and empname)
Transaction to update ZEMP_MAST	ZEMPMR

9.2.6.1. Step 1 – Create a new employee/ update an existing employee in sender system using transaction ZEMPMR.

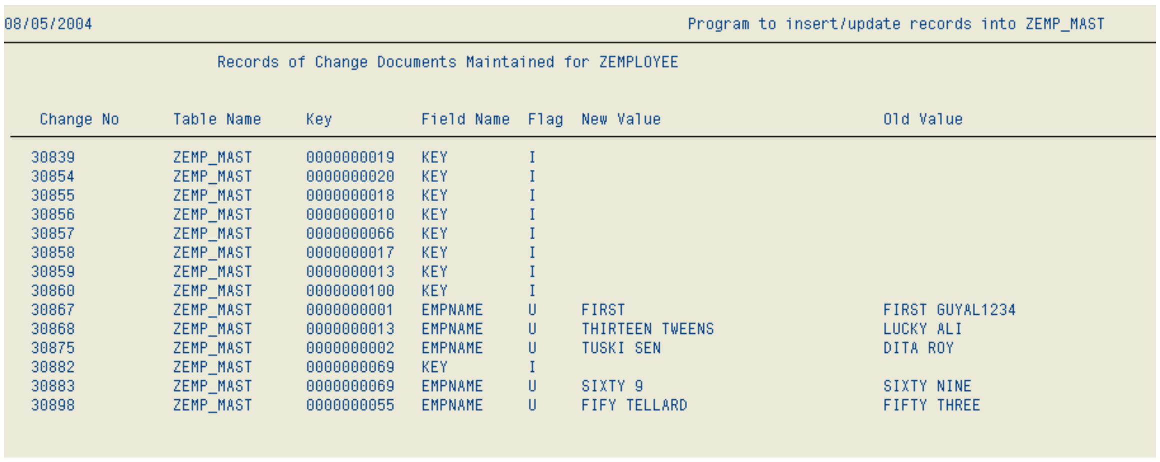

Program to insert/update records into ZEMP_MAST

Enter Employee Id 55
Enter Name Fify Tellard

☐ Data will be inserted

☑ Show latest report of CDPOS

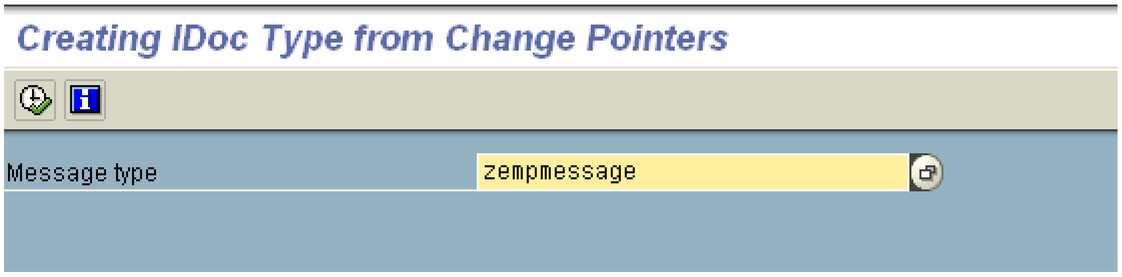
Enter an existing employee code .Change the name . Tick the checkbox to display the values in database table CDPOS after insertion . Execute the program.



```
08/05/2004                                                         Program to insert/update records into ZEMP_MAST

                         Records of Change Documents Maintained for ZEMPLOYEE

   Change No       Table Name       Key            Field Name  Flag  New Value                           Old Value

   30839           ZEMP_MAST        0000000019     KEY         I
   30854           ZEMP_MAST        0000000020     KEY         I
   30855           ZEMP_MAST        0000000018     KEY         I
   30856           ZEMP_MAST        0000000010     KEY         I
   30857           ZEMP_MAST        0000000066     KEY         I
   30858           ZEMP_MAST        0000000017     KEY         I
   30859           ZEMP_MAST        0000000013     KEY         I
   30860           ZEMP_MAST        0000000100     KEY         I
   30867           ZEMP_MAST        0000000001     EMPNAME     U     FIRST                               FIRST GUYAL1234
   30868           ZEMP_MAST        0000000013     EMPNAME     U     THIRTEEN TWEENS                     LUCKY ALI
   30875           ZEMP_MAST        0000000002     EMPNAME     U     TUSKI SEN                           DITA ROY
   30882           ZEMP_MAST        0000000069     KEY         I
   30883           ZEMP_MAST        0000000069     EMPNAME     U     SIXTY 9                             SIXTY NINE
   30898           ZEMP_MAST        0000000055     EMPNAME     U     FIFY TELLARD                        FIFTY THREE
```

An output is published , which shows that change documents have been maintained for new employee : 55

9.2.6.2. Step 2 – Execute program RBDMIDOC from SE38 in sender system

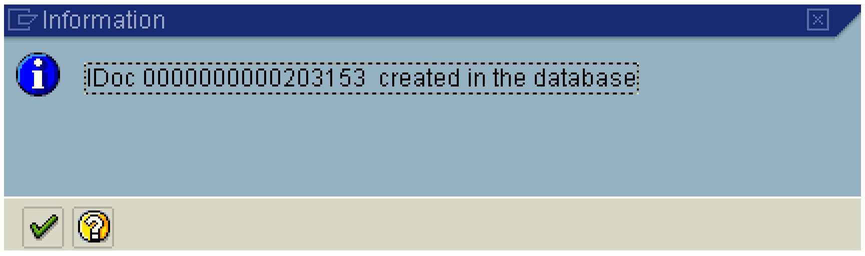


Creating IDoc Type from Change Pointers

Message type: zempmessage

Enter the message type. Execute.

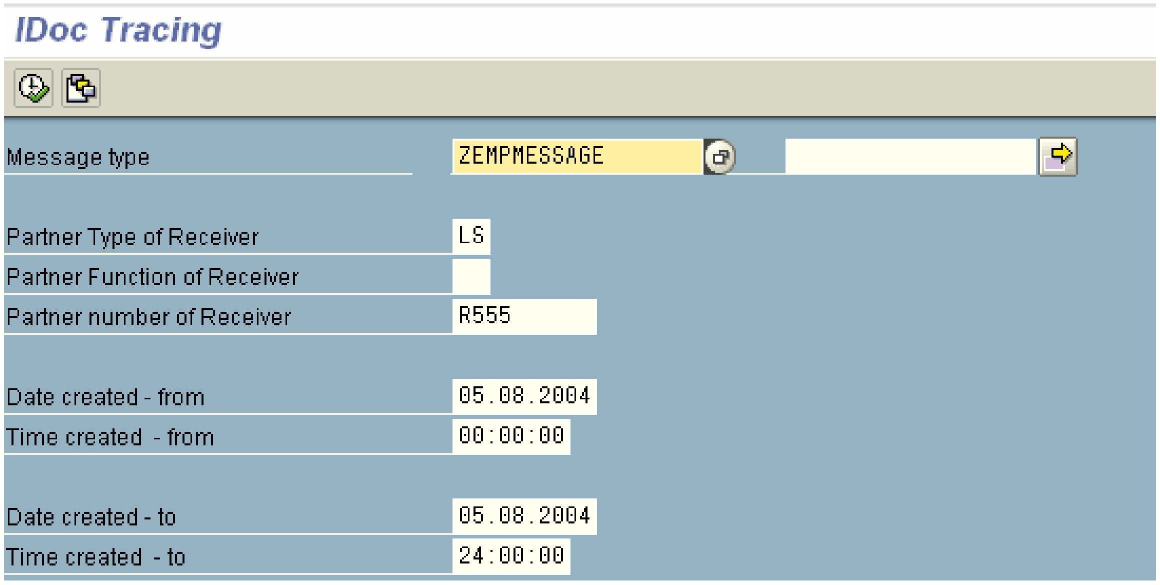
This program will call the function module you have created to read the change pointer information from database and finally will create and distribute Master IDoc.



IDoc 0000000000203153 created in the database

A message will inform you about the outbound IDoc number.

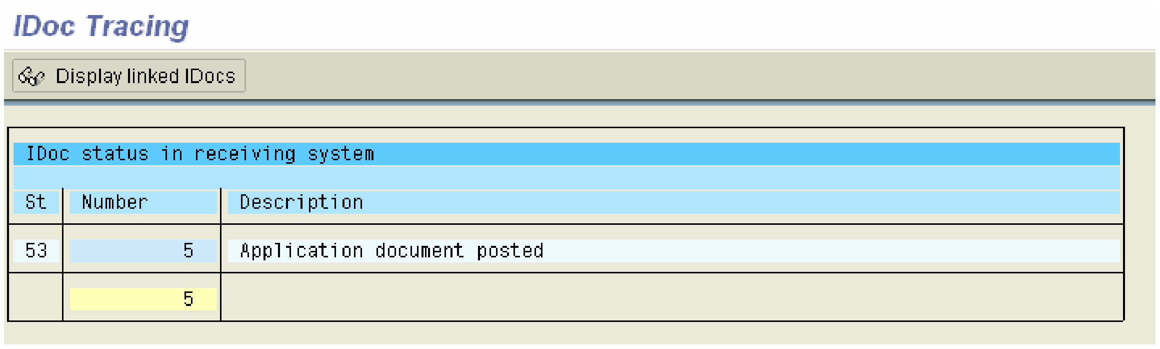
9.2.6.3. Step 3 – Verify the Status of Outbound and Inbound Idocs of both systems from tcode BDM2 in sender system



IDoc Tracing

Message type	ZEMPMESSAGE
Partner Type of Receiver	LS
Partner Function of Receiver	
Partner number of Receiver	R555
Date created - from	05.08.2004
Time created - from	00:00:00
Date created - to	05.08.2004
Time created - to	24:00:00

Go to BDM2. Enter the name of the message type and the name of the receiver system. Execute.

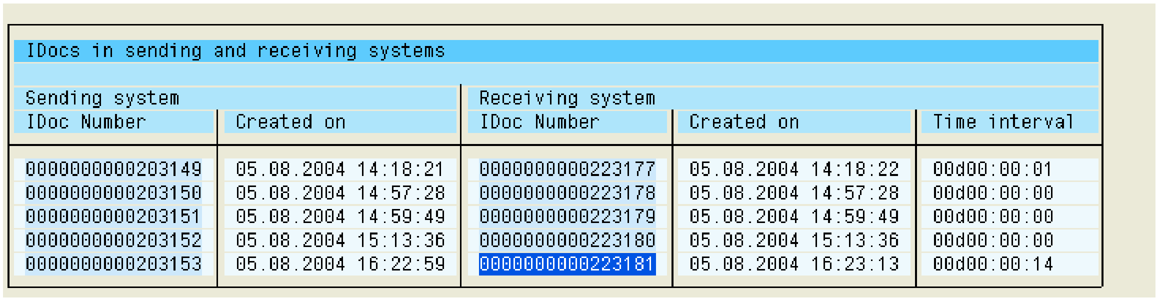


IDoc Tracing

IDoc status in receiving system

St	Number	Description
53	5	Application document posted
	5	

A report will show you the number of Idocs transferred between two systems.
Double click on the total line(marked in yellow).



IDocs in sending and receiving systems

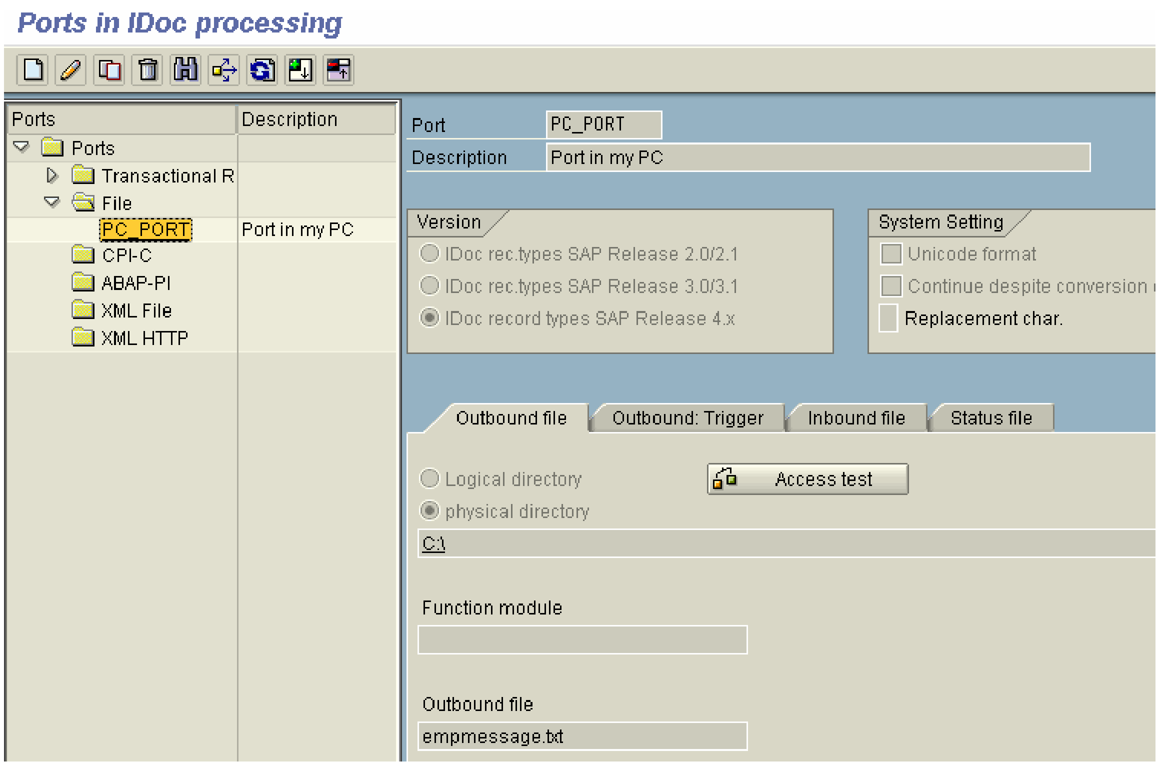
Sending system		Receiving system		
IDoc Number	Created on	IDoc Number	Created on	Time interval
0000000000203149	05.08.2004 14:18:21	0000000000223177	05.08.2004 14:18:22	00d00:00:01
0000000000203150	05.08.2004 14:57:28	0000000000223178	05.08.2004 14:57:28	00d00:00:00
0000000000203151	05.08.2004 14:59:49	0000000000223179	05.08.2004 14:59:49	00d00:00:00
0000000000203152	05.08.2004 15:13:36	0000000000223180	05.08.2004 15:13:36	00d00:00:00
0000000000203153	05.08.2004 16:22:59	0000000000223181	05.08.2004 16:23:13	00d00:00:14

Look at the list published. Your IDoc has created an inbound IDoc 223181 in the receiver system(Shatadru, 555). Double-click on the IDoc number in each systems to view their status in respective systems.

10. Downloading IDoc into Application server

10.1. Create the file port

To download the IDoc as a file in the application server, one need to create a file port in sender system using tcode: WE21.



PC_PORT Port in my PC

Port: PC_PORT
Description: Port in my PC

Version
- IDoc rec.types SAP Release 2.0/2.1
- IDoc rec.types SAP Release 3.0/3.1
- [✓ IDoc record types SAP Release 4.x]

System Setting
- Unicode format
- Continue despite conversion
- Replacement char.

Outbound file
- Logical directory
- [✓ physical directory]

C:\

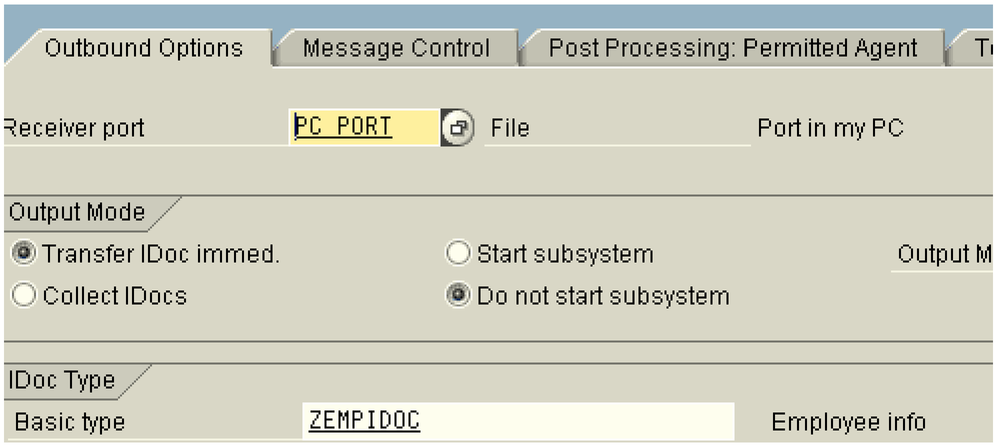
Function module

Outbound file
empmessage.txt

In the diagram shown above, a file port is created so that the IDoc file is maintained under the 'C:\' directory in application server as a text file, called empmessage.txt.

10.2. Change Outbound partner profile

Now, mention this file port in the outbound partner profile for the communication.



Receiver port PC_PORT File Port in my PC

Output Mode

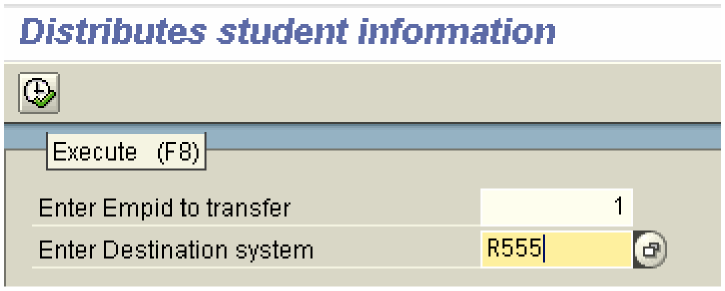
Transfer IDoc immed. [✓] | Start subsystem | Output M
Collect IDocs | Do not start subsystem [✓]

IDoc Type

Basic type ZEMPIDOC Employee info

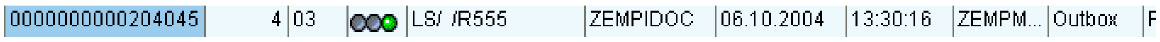
10.3. Trigger the outbound process

After that, trigger the outbound process and check for successful transfer of the Idoc. In this case, it is execution of program ZEMP_OUTBOUND.


Distributes student information

Enter Empid to transfer 1
Enter Destination system R555

10.4. Check the status of the IDoc from WE02



```
0000000000204045     4 |03  |●●● |LS/ /R555    |ZEMPIDOC |06.10.2004 |13:30:16 |ZEMPM... |Outbox |F
```

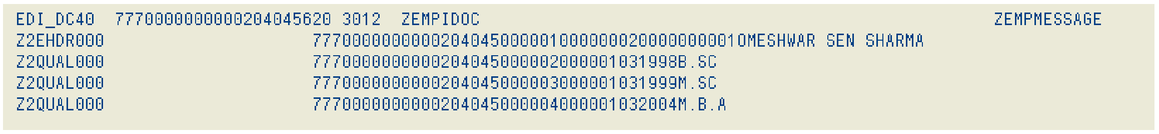
10.5. Check the physical text file for the Idoc

Then, login into the application server to view the Idoc. If not possible, write a separate program to view whether the Idoc is successfully created as a file in the application server or not.

In the following context, an ABAP program is written as follows for verification:-

```
REPORT YSUBOOPS7                          .

data : begin of itab occurs 0 ,
      line type string ,
      end of itab,
      xtab like line of itab .

data : subrc like sy-subrc.
open dataset 'C:\empmessage.txt' for input in text mode.
if sy-subrc eq 0.
 while subrc eq 0.
  read dataset 'C:\empmessage.txt' into itab-line.
   subrc = sy-subrc.
   append   itab.
   clear itab.
 endwhile.
close dataset 'C:\empmessage.txt'.
endif.

loop at itab.
 write:/5 itab-line.
endloop.
```

On execution of the same for verification, it shows the same:-



```
EDI_DC40  7770000000000204045620 3012  ZEMPIDOC                                        ZEMPMESSAGE
Z2EHDR000                     77700000000002040450000010000000200000000010MESHWAR SEN SHARMA
Z2QUAL000                     7770000000000204045000002000001031998B.SC
Z2QUAL000                     7770000000000204045000003000001031999M.SC
Z2QUAL000                     7770000000000204045000004000001032004M.B.A
```