**Transferring the Monthly Employee Report data through IDoc’s**

**From System A (SDI-100) To System B(SDD-400).**

**Scope :**

* **Building everything from scratch including Employee Data.**
* **Creation of Outbound IDoc (Configuration for New Ouboud and Stand alone program)**
* **Creation of Inboud IDoc (Configuration of Inboud IDoc and Inbound Function Module).**

**We will be creating a monthly Report similar to the below screen shot.**



**We will be creating the Data Elements for the following fields.**

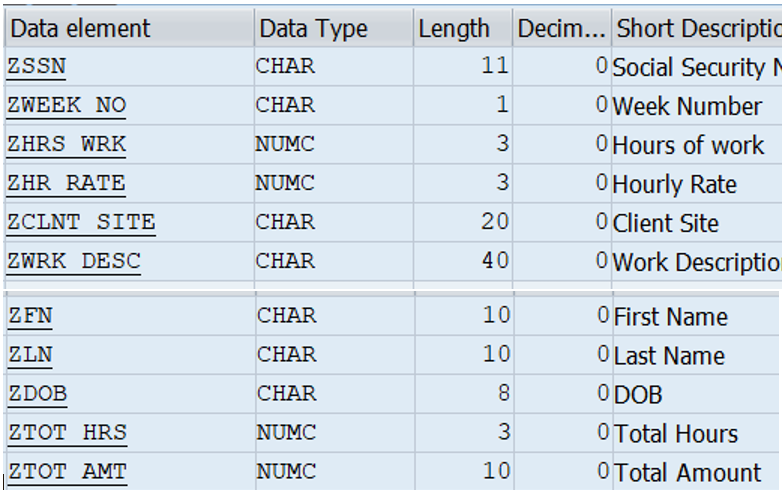


**The Basic IDoc Type will be the created as per the below screen shot.**

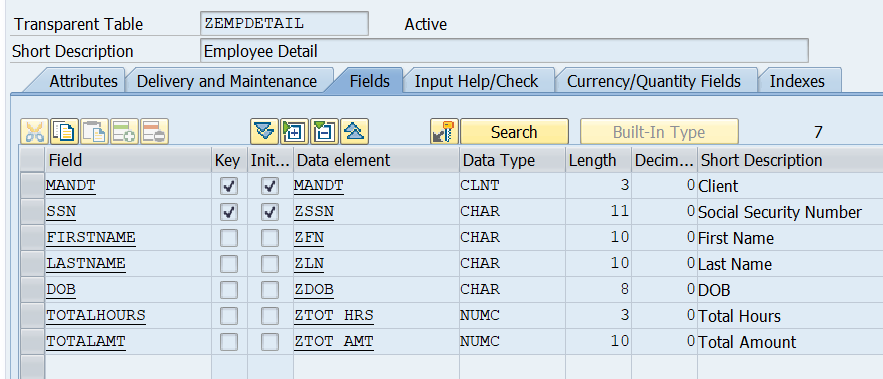
**Step 1:** Create Data Elements required for creating Tables in both the systems.

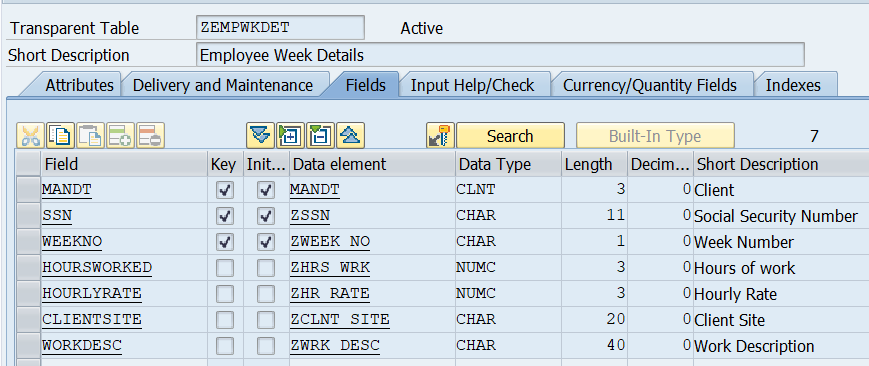
The step should be done in both the systems (SDI-100 ) and (SDD-400)

**Create Data Elements:**



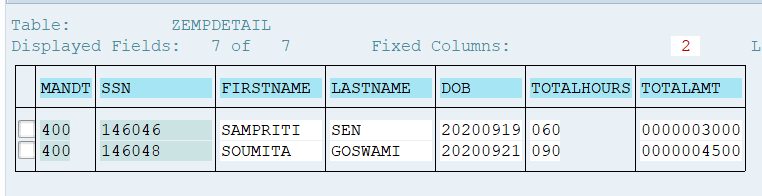
Create 2 Database Tables: ZEMPDETAIL and ZEMPWKDET

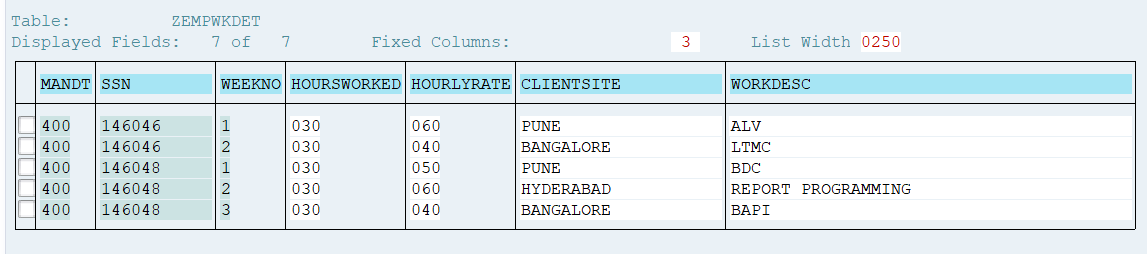




**Step 2:** Create a Table Maintenance Generator for Input of Data Records in SDI-100.

We won’t require the table maintenance Generator in SDD-400 as will we be moving data from SDI-100 to SDD-400 using IDoc’s.





The above steps are for creation of application and it’s data for our example.

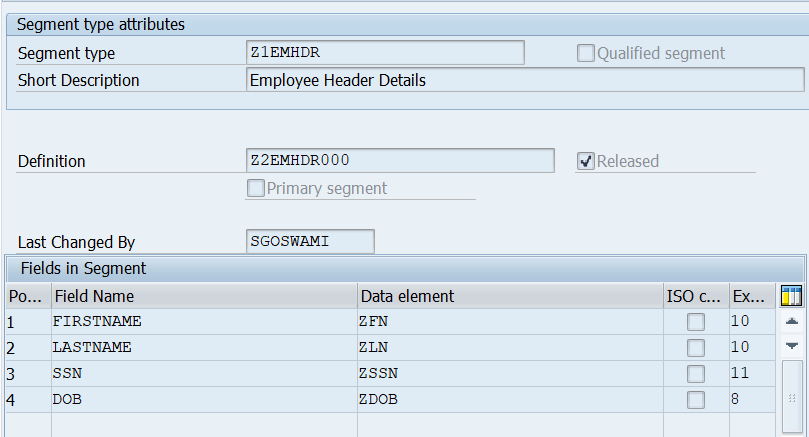
From here on we will be creating the steps required for IDoc.

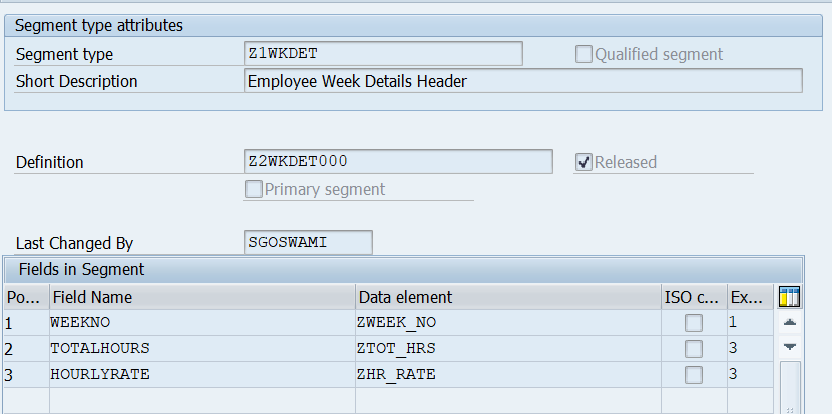


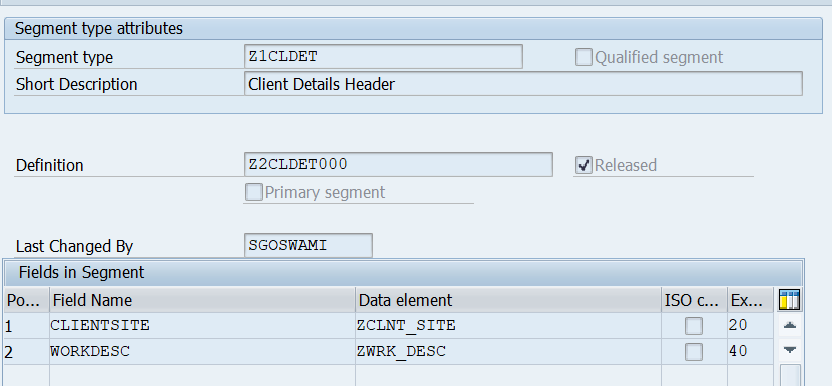
**Step 3:** Create the segments (Perform this step in both the systems)

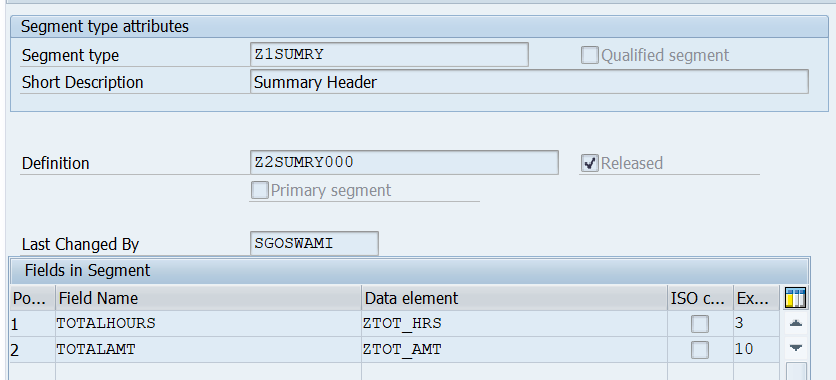
Transaction Code: WE31.

Create 4 segments namely, Z1EMHDR, Z1WKDET, Z1CLDET, Z1SUMRY.

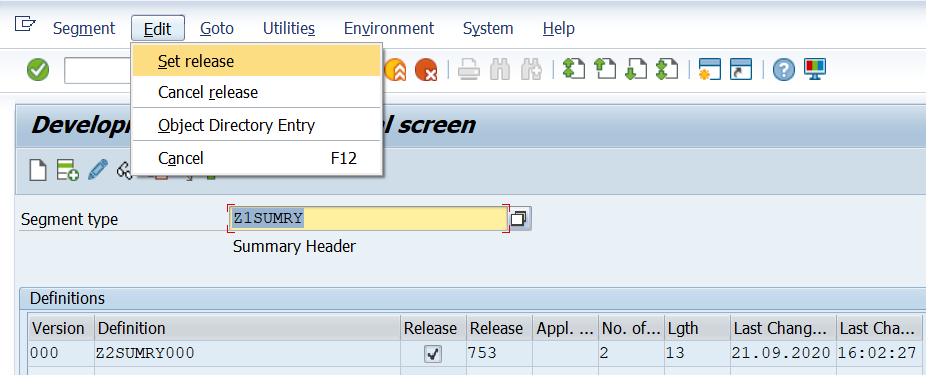








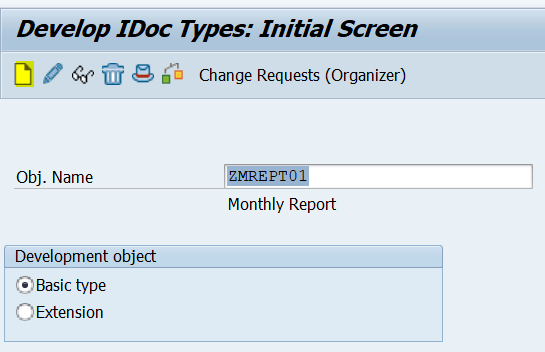
After Creation of 4 segments, these have to be released individually, using Edit > Set release menu option.



**Step 4:** Create a Basic IDoc Type. (This step should be performed in both the systems).

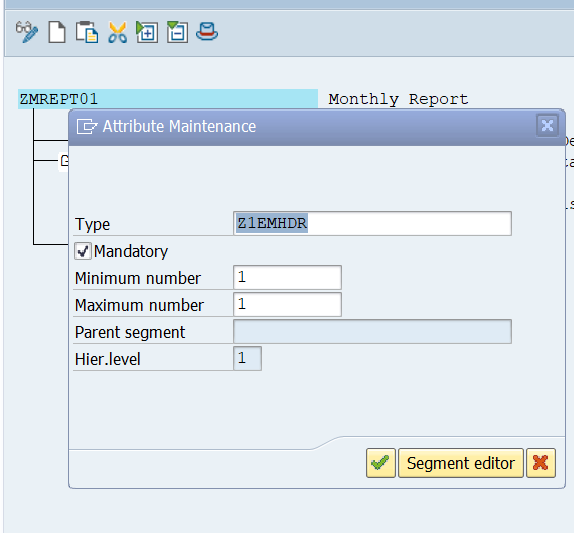
Transcation Code : WE30.

Creaate a Basic IDoc Type with name ZMREPT01.

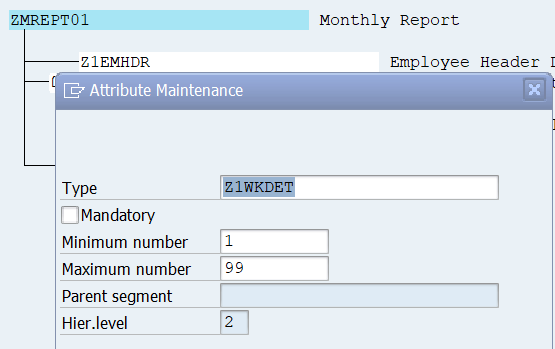


Click on create and insert the segments.

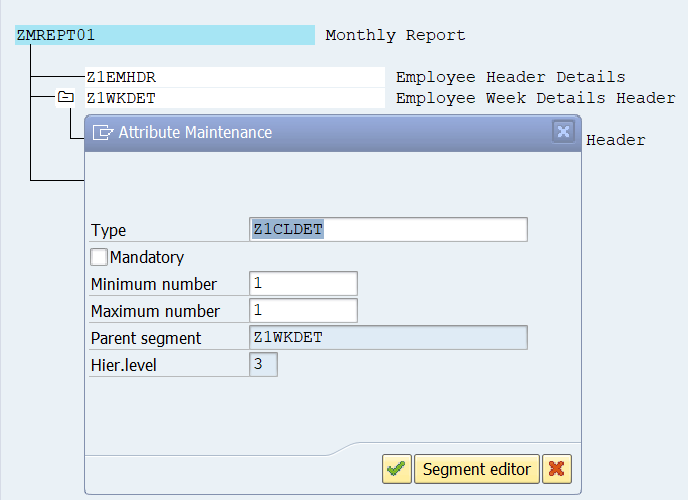
Z1EMHDR segment is Mandatory. Without header we will not transfer the data as it does it makes any sense.



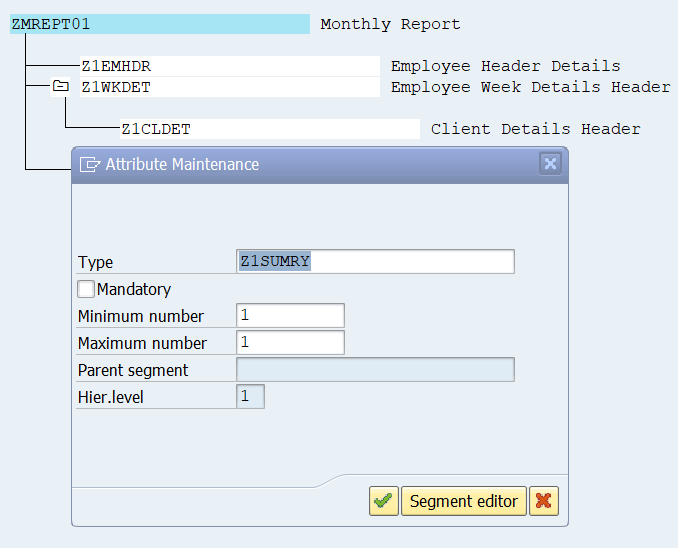
Next, insert the segment Z1WKDET for weekly details. It is not a mandatory segment during transfer of data. As it is weekly data it can be repeated multiple times.



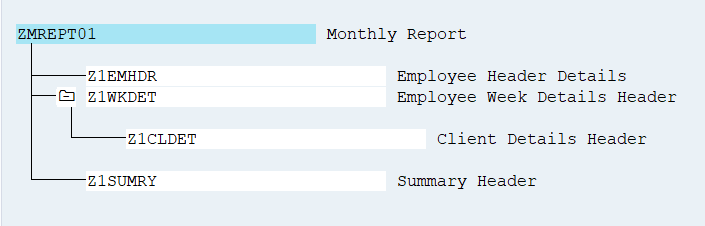
Next, insert the segment Z1CLDET for client/site details. It will be a child segment for Z1WKDET



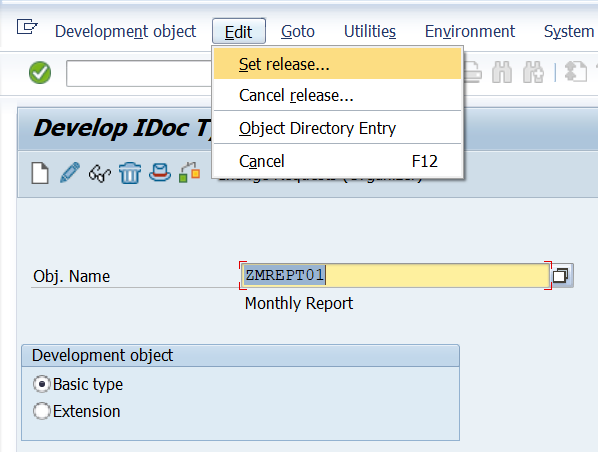
Next , insert the summary segment Z1SUMRY at the end.



Now the Basic IDoc type looks like below:



Once it is created, we need to release the Basic IDoc Type.



**Step 5: Create a Message Type.**

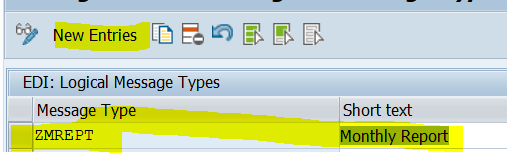
Transaction: WE81.

This Step is carried in both the systems.

You assign a message type to the data contents transferred in the IDoc and give it a short description.

Customer−defined messages begin with Z. This should be the same name used in your IDoc programs.

monthly report IDoc, message type ZMREPT is chosen.

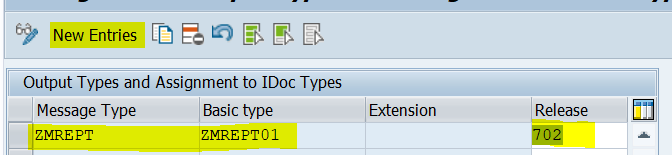


**Step 6: Link the IDoc Type to the Message Type.**

Transaction WE82

This step is carried out in both the systems.

You assign the message type created in the preceding step to the IDoc type, This step not only serves to document which message is based on which IDoc type, but it also checks this link in the process when IDocs are generated.



Now we have to maintain the

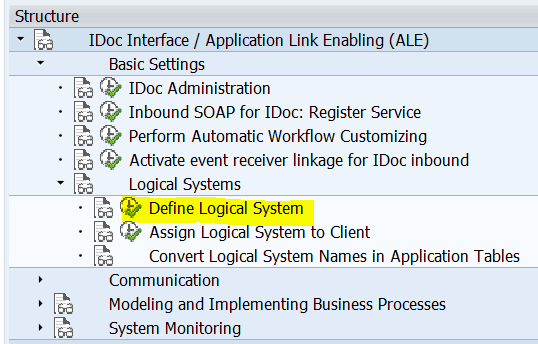
**Communication settings.**

**Step 7:** Now we have to maintain the **logical Systems** that are involved in the distribution of IDoc.

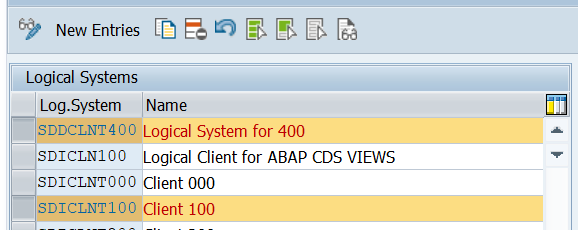
This step should be performed in both the systems.

Note: This is a one-time step. If the system is configured previously, for the exchange of data for other master data transfer, we can skip the step.

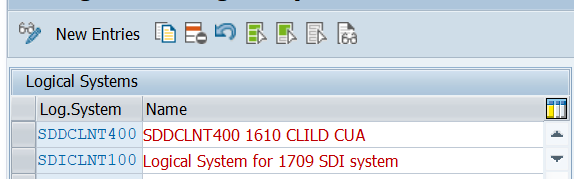
Transaction Code: SALE.



Define Logical systems in SDI-100:



Define Logical Systems in SDD-400 :

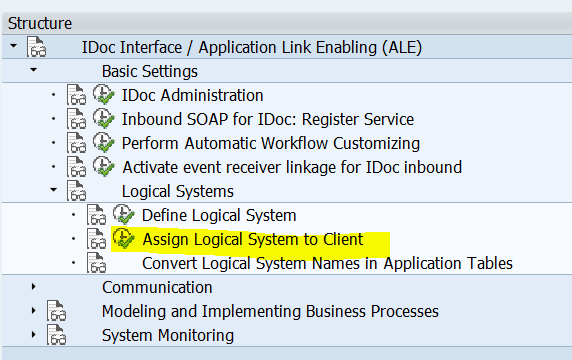


**Step 8: Assign the logical system to client.**

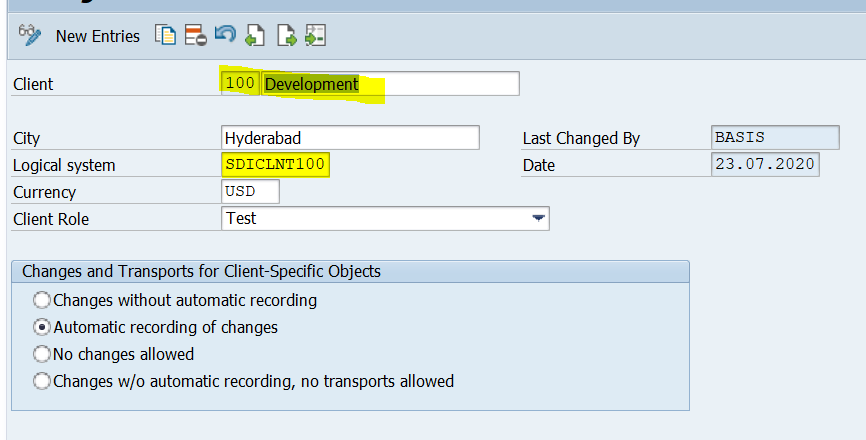
This step should be performed in both the systems.

Note: This is a one-time step. If the system is configured previously, for the exchange of data for other master data transfer, we can skip the step.

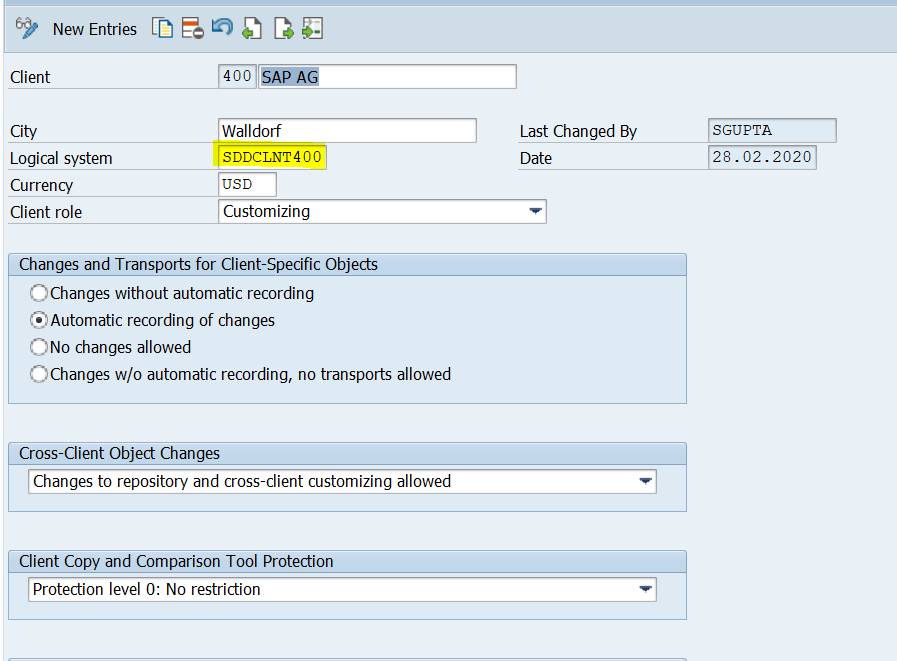
Transaction Code: SALE.



In SDI-100, assign the logical system to client 100. The step establishes the link to client 100.



In SDI-400, assign the logical system to client 400. The step establishes the link to client 400.



**Step 9:** **Create RFC Destinations**

Transaction code: SM59.

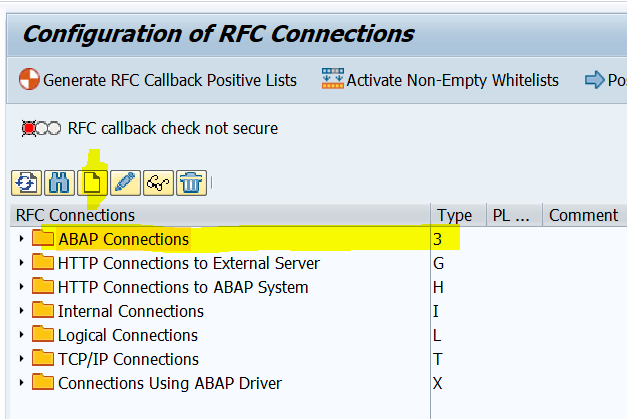
As the data flows from SD1-100 to SDD-400.

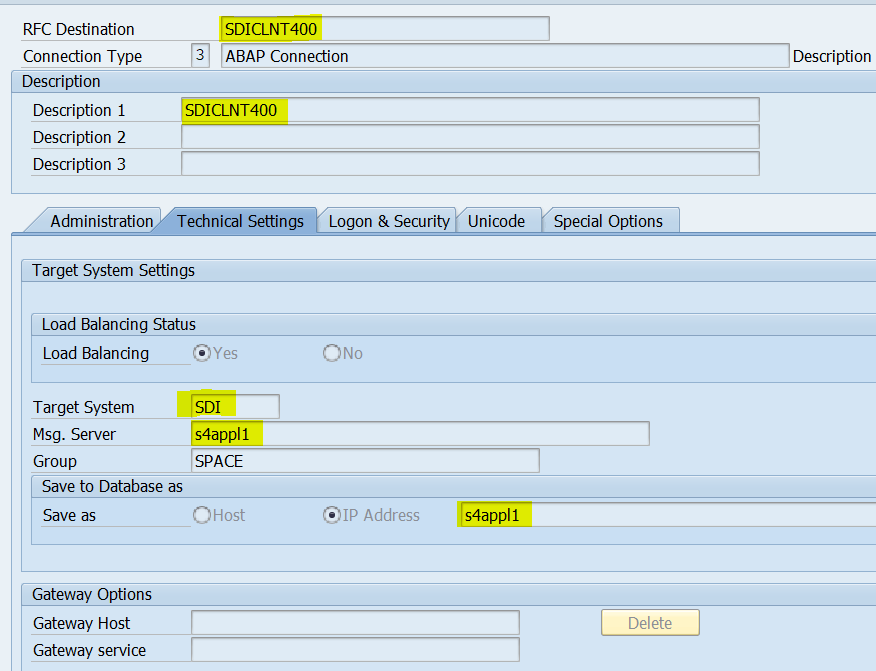
RFC destination for SDD-400 should be maintained in SD1-100.

If there is two way communication, the we need to create the Destinations in both the systems.

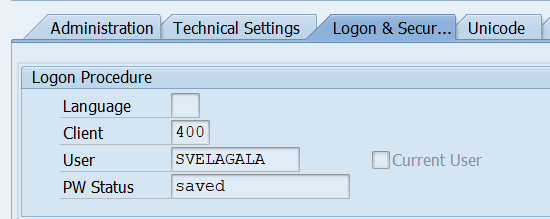
Note: if there exists a destination already configured for other messages, this is not required. This is one time connection job from System A to System B.

We need to create ABAP connections, so, keep the cursor on ABAP connections and then press create button.





Usually, the User will be of system user of non-dailog type configured by basis team.



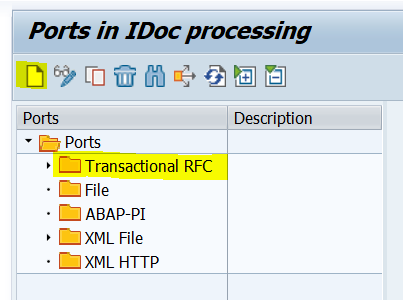
**Step 10: Create a PORT**

Transaction WE21

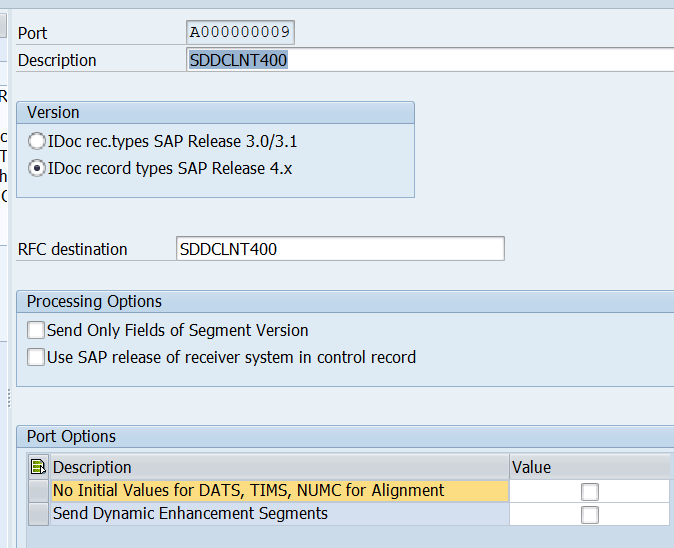
As the data flows from SD1-100 to SDD-400. PORT in SD1 for SDD-400 should be maintained in SD1-100.

If there is two way communication, the we need to create the PORT’s in both the systems.

Note: if there exists a Port already configured for other messages, this is not required. This is one time connection job from System A to System B.



We create Transactional RFC ports for the transfer of data in ALE. (TRFC means data is transferred in Memory blocks).



**Step 11: Create a Partner Profile**

Transaction Code: WE20

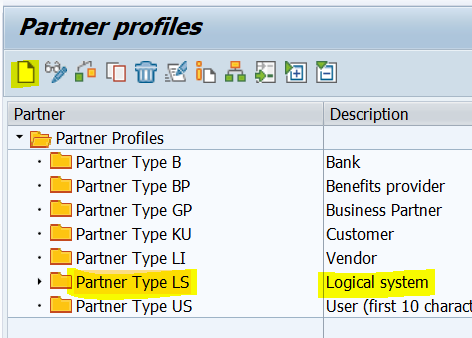
The step should be performed, in both the systems. Partner type will be LS (as we transfer data within SAP Connected Systems)

For every IDoc message we need to configure the partner profiles.

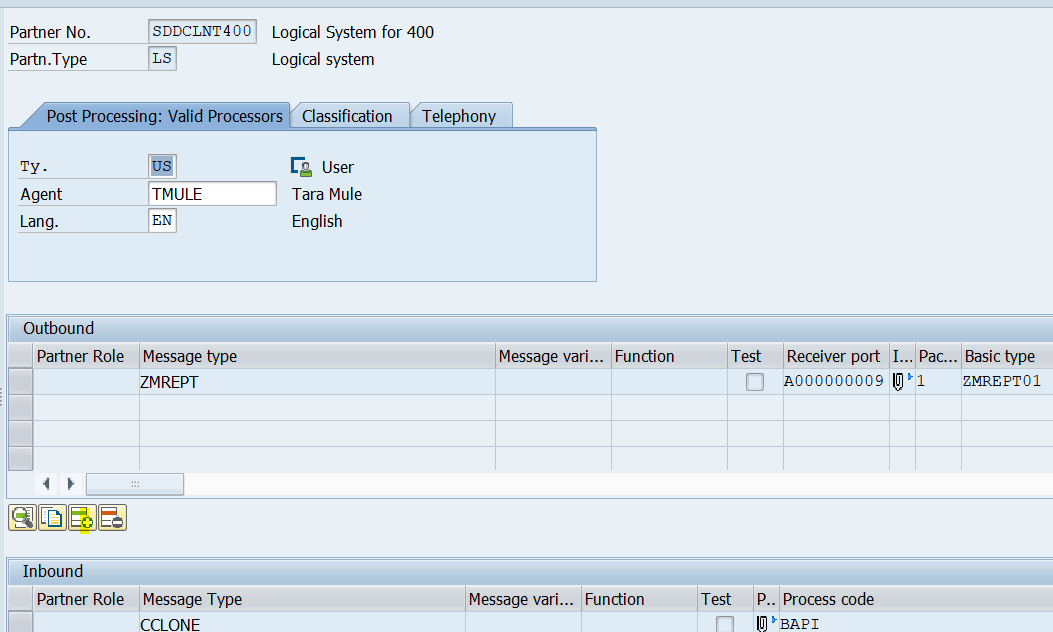
In SDI-100 system, we need to create the outbound partner profile.

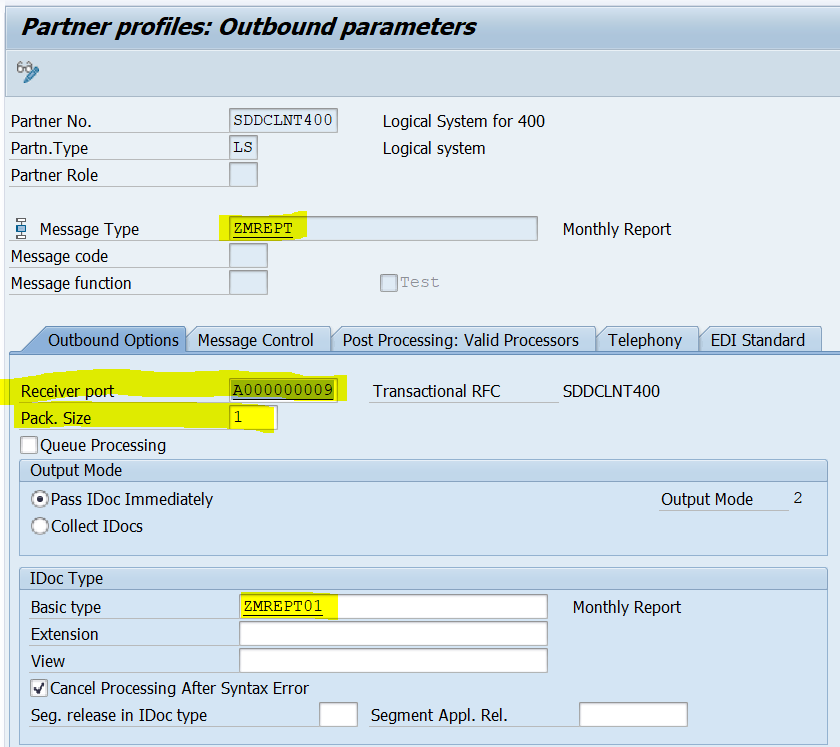
In SDD- 400 system, we need to create the inbound partner profile.

**Configuration in SDI-100:**



Click on + icon below the outbound Messages.

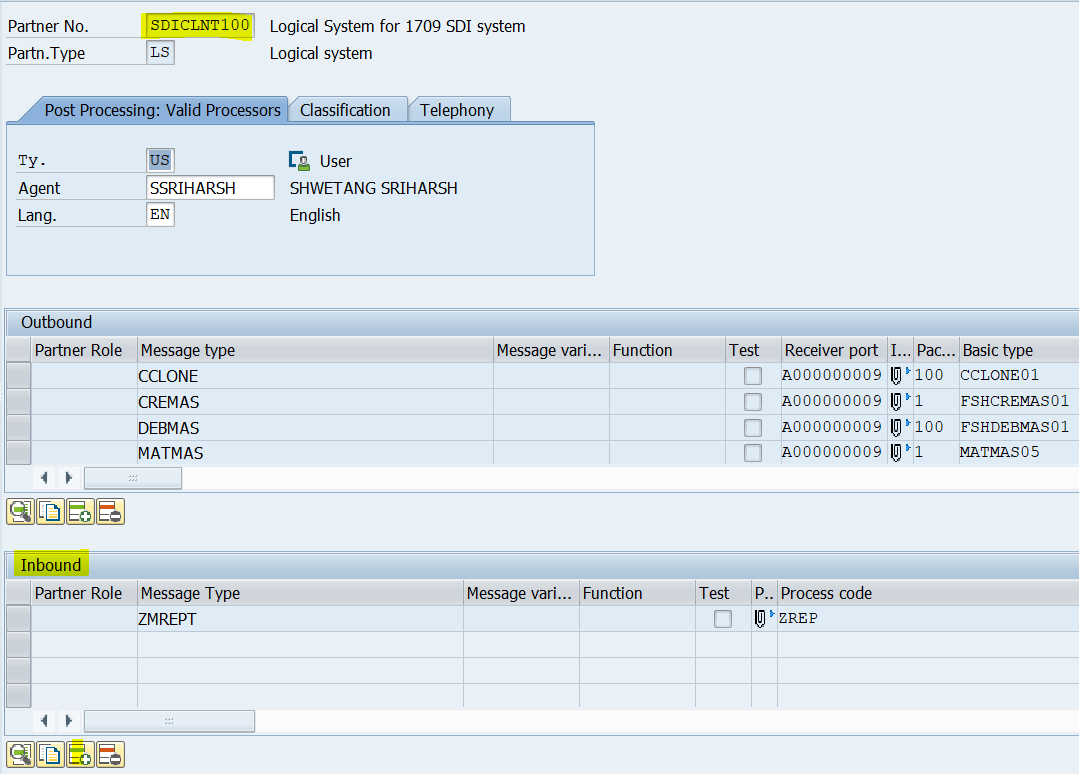




**Configuration in SDD-400:**

Click on + icon below the inbound Messages.

**Note: The inbound Partner Profile configuration should be done once the Inbound function module for posting data is ready in SDD-400, and a process code is attached to it.**



**Step 12: Create a Stand Alone program for extraction of Data from tables and to create a IDoc from it.**

Outbound Programs for Custom Basic IDoc Types

Program Flow

The program logic contains the following blocks.

1. Provide a selection screen to enable a user to specify the objects for which IDocs are to be generated.

2. Determine the key of the application document from the object specified in Step 1.

3. Select application data from the database, using the object key identified in Step 2.

4. Populate the control record information.

5. Populate an internal table of type EDIDD with data records for the various segments.

6. Call the ALE service layer (MASTER\_IDOC\_DISTRIBUTE) to create the IDocs in the database.

7. Execute a Commit work.

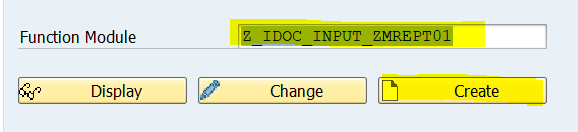
**In SD1-100 Create a Z-Report for Pushing the Data.**

\*&---------------------------------------------------------------------\*  
\*& Report ZALE\_OUT  
\*&---------------------------------------------------------------------\*  
\*&  
\*&---------------------------------------------------------------------\*  
REPORT ZALE\_OUT MESSAGE-ID ZE.  
  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* Parameters  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* object key (social security number for the employee)  
PARAMETERS: p\_ssn LIKE zempdetail-ssn.  
\* message type  
PARAMETERS: p\_mestyp LIKE edmsg-msgtyp OBLIGATORY.  
\* destination system  
PARAMETERS: p\_logsys LIKE tbdlst-logsys.  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* Constants  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
DATA:  
\* segment names  
  c\_header\_segment           LIKE edidd-segnam VALUE 'Z1EMHDR',  
  c\_weekly\_details\_segment   LIKE edidd-segnam VALUE 'Z1WKDET',  
  c\_client\_details\_segment   LIKE edidd-segnam VALUE 'Z1CLDET',  
  c\_summary\_segment          LIKE edidd-segnam VALUE 'Z1SUMRY',  
\* IDoc type  
  c\_monthly\_report\_idoc\_type LIKE edidc-idoctp VALUE 'ZMREPT01'.  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* Data declarations  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* IDoc control record  
DATA: control\_record\_out LIKE edidc.  
\* employee header data  
DATA: fs\_emphdr\_data LIKE z1emhdr.  
\* employee weekly details data  
DATA: fs\_weekdet\_data LIKE z1wkdet.  
\* client details data  
DATA: fs\_clientdet\_data LIKE z1cldet.  
\* employee monthly summary data  
DATA: fs\_summary\_data LIKE z1sumry.  
\* total hours and amount for the summary segment  
DATA: total\_hrs\_month TYPE i,  
      total\_amt\_month TYPE i.  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* Database Tables  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
  
\* Application data tables  
TABLES: zempdetail, zempwkdet.  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* Internal tables  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
DATA:  
\* weekly details − application data  
  it\_wkdet      LIKE zempwkdet OCCURS 0 WITH HEADER LINE,  
\* data records  
  int\_edidd     LIKE edidd OCCURS 0 WITH HEADER LINE,  
\* communication IDocs generated  
  it\_comm\_idocs LIKE edidc OCCURS 0 WITH HEADER LINE.  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* Program logic  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Select Application Data\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
SELECT SINGLE \* FROM zempdetail WHERE ssn = p\_ssn.  
IF sy-subrc NE 0.  
  MESSAGE e001 WITH p\_ssn.  
  EXIT.  
ENDIF.  
SELECT \* FROM zempwkdet INTO TABLE it\_wkdet WHERE ssn = p\_ssn.  
IF sy-subrc NE 0.  
  MESSAGE e002 WITH p\_ssn.  
  EXIT.  
ENDIF.  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Build Control Record\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* Fill control record information  
control\_record\_out-mestyp = p\_mestyp.  
control\_record\_out-idoctp = c\_monthly\_report\_idoc\_type.  
control\_record\_out-rcvprt = 'LS'.  
control\_record\_out-rcvprn = p\_logsys.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Build Data Records\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*−−−−−−−−−−−−−−−−−−Employee header−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* fill the employee header information  
fs\_emphdr\_data-lastname = zempdetail-lastname.  
fs\_emphdr\_data-firstname = zempdetail-firstname.  
fs\_emphdr\_data-ssn = zempdetail-ssn.  
fs\_emphdr\_data-dob = zempdetail-dob.  
\* fill the administrative section of the data record  
int\_edidd-segnam = c\_header\_segment.  
int\_edidd-sdata = fs\_emphdr\_data.  
\* append the employee header data record to the IDoc data  
APPEND int\_edidd.  
\*−−−−−−−−−−−−−−−−−−−Employee weekly details−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
LOOP AT it\_wkdet.  
\* fill the weekly details for each week  
  fs\_weekdet\_data-weekno = it\_wkdet-weekno.  
  
  fs\_weekdet\_data-totalhours = it\_wkdet-HOURSWORKED.  
  fs\_weekdet\_data-hourlyrate = it\_wkdet-hourlyrate.  
\* add administrative information to the data record  
  int\_edidd-segnam = c\_weekly\_details\_segment.  
  int\_edidd-sdata = fs\_weekdet\_data.  
\* append the data for the week to the IDoc data  
  APPEND int\_edidd.  
\* Client details of each week  
  fs\_clientdet\_data-clientsite = it\_wkdet-clientsite.  
  fs\_clientdet\_data-workdesc = it\_wkdet-workdesc.  
\* add administrative information to the data record  
  int\_edidd-segnam = c\_client\_details\_segment.  
  int\_edidd-sdata = fs\_clientdet\_data.  
\* append the client details for the week to the IDoc data  
  APPEND int\_edidd.  
ENDLOOP.  
\*−−−−−−−−−−−−−−−−−−−Employee monthly summary−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* compute total hours and amount for the month  
LOOP AT it\_wkdet.  
  total\_hrs\_month = total\_hrs\_month + it\_wkdet-HOURSWORKED.  
  total\_amt\_month = total\_amt\_month + ( it\_wkdet-HOURSWORKED \*  
  it\_wkdet-hourlyrate ).  
ENDLOOP.  
\* fill the summary information  
fs\_summary\_data-totalhours = total\_hrs\_month.  
fs\_summary\_data-totalamt = total\_amt\_month.  
\* condense the summary record fields to remove spaces  
CONDENSE fs\_summary\_data-totalhours.  
CONDENSE fs\_summary\_data-totalamt.  
\* add administrative information to the data record  
int\_edidd-segnam = c\_summary\_segment.  
int\_edidd-sdata = fs\_summary\_data.  
\* append summary data to the IDoc data  
APPEND int\_edidd.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*Pass control to the ALE layer\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
CALL FUNCTION 'MASTER\_IDOC\_DISTRIBUTE'  
  EXPORTING  
    master\_idoc\_control            = control\_record\_out  
  TABLES  
    communication\_idoc\_control     = it\_comm\_idocs  
    master\_idoc\_data               = int\_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 NE 0.  
  MESSAGE e003 WITH p\_ssn.  
ELSE.  
  LOOP AT it\_comm\_idocs.  
    WRITE: / 'IDoc generated', it\_comm\_idocs-docnum.  
  ENDLOOP.  
  CALL FUNCTION 'DB\_COMMIT'.  
  CALL FUNCTION 'DEQUEUE\_ALL'.  
  COMMIT WORK.  
ENDIF.

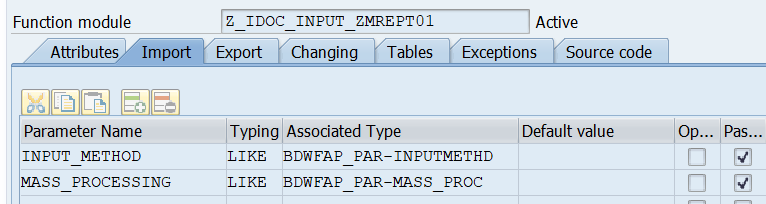
**Step 13: Create an inbound function module for INBOUND IDOC in SDD-400.**

The inbound IDoc is implemented as a function module.

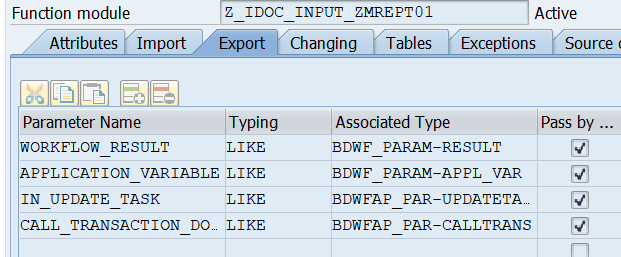
This function module will post the data by reading the inbound IDoc.



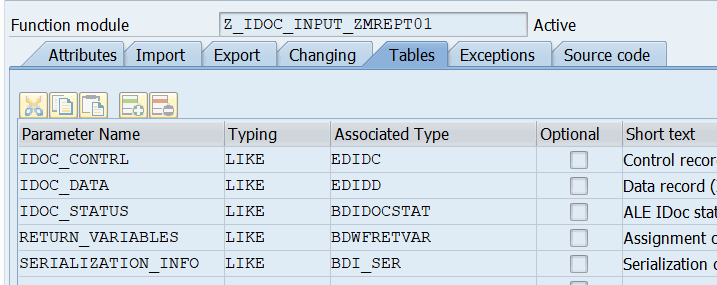
The interface for Import parameters will be as below: This is same for all the INBOUND Idocs.



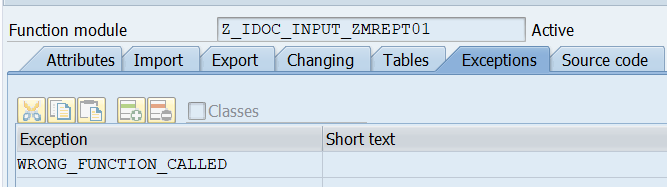
Interface for Export parameter is as below:



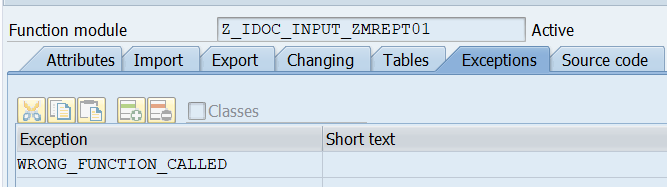
Interface for Tables parameters is as below:



Exceptions:



Source Code:



FUNCTION Z\_IDOC\_INPUT\_ZMREPT01 .  
\*"----------------------------------------------------------------------  
\*"\*"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  
\*"----------------------------------------------------------------------  
  INCLUDE mbdconwf.  
  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* Database Tables  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* Application data tables (Defined in global data)  
\*tables: zempdetail, zempwkdet.  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* Data declarations  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* employee details − IDoc  
  DATA: fs\_emphdr\_data LIKE z1emhdr.  
\* employee weekly details data − IDoc  
  DATA: fs\_weekdet\_data LIKE z1wkdet.  
\* client details data − IDoc  
  DATA: fs\_clientdet\_data LIKE z1cldet.  
\* employee monthly summary data − IDoc  
  DATA: fs\_summary\_data LIKE z1sumry.  
\* total hours and amount  
  DATA: total\_hrs\_month TYPE i,  
        total\_amt\_month TYPE i.  
  
\* employee details − application data  
  DATA: fs\_app\_empdet LIKE zempdetail.  
\* weekly details − application data  
  DATA: it\_app\_wkdet LIKE zempwkdet OCCURS 0 WITH HEADER LINE.  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* Program logic  
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−  
\* initialize workflow result  
  workflow\_result = c\_wf\_result\_ok.  
  LOOP AT idoc\_contrl.  
\* make sure we have the correct message passed to us  
    IF idoc\_contrl-mestyp NE 'ZMREPT'.  
      RAISE wrong\_function\_called.  
    ENDIF.  
\* clear application buffers before reading new employee  
    CLEAR: it\_app\_wkdet, fs\_app\_empdet.  
    REFRESH it\_app\_wkdet.  
\* process all data records in an IDoc and transfer them to  
\* application buffers  
    LOOP AT idoc\_data WHERE docnum EQ idoc\_contrl-docnum.  
      CASE idoc\_data-segnam.  
  
        WHEN 'Z1EMHDR'. " employee header  
          fs\_emphdr\_data = idoc\_data-sdata.  
          move-corresponding fs\_emphdr\_data to fs\_app\_empdet.  
        WHEN 'Z1WKDET'. " employee weekly details  
          fs\_weekdet\_data = idoc\_data-sdata.  
          move-corresponding fs\_weekdet\_data to it\_app\_wkdet.  
          it\_app\_wkdet-hoursworked = fs\_weekdet\_data-totalhours.  
        WHEN 'Z1CLDET'. " client details  
          fs\_clientdet\_data = idoc\_data-sdata.  
          move-corresponding fs\_clientdet\_data to it\_app\_wkdet.  
          MOVE fs\_app\_empdet-ssn TO it\_app\_wkdet-ssn.  
          APPEND it\_app\_wkdet. " append weekly details  
        WHEN 'Z1SUMRY'. " summary data  
          fs\_summary\_data = idoc\_data-sdata.  
      ENDCASE.  
    ENDLOOP.  
  
\* verify totals in the data records against the summary record  
    CLEAR: total\_hrs\_month, total\_amt\_month.  
\* compute total hours and amount for the month from weekly details  
    LOOP AT it\_app\_wkdet.  
      total\_hrs\_month = total\_hrs\_month + it\_app\_wkdet-hoursworked.  
      total\_amt\_month = total\_amt\_month + ( it\_app\_wkdet-hoursworked \*  
      it\_app\_wkdet-hourlyrate ).  
    ENDLOOP.  
    fs\_app\_empdet-totalhours = total\_hrs\_month.  
    fs\_app\_empdet-totalamt = total\_amt\_month.  
\* compare the values with values in the summary record  
    IF total\_hrs\_month NE fs\_summary\_data-totalhours OR  
    total\_amt\_month NE fs\_summary\_data-totalamt.  
  
\* totals in the summary record do not match with weekly details  
\* fill IDOC\_Status  
      idoc\_status-docnum = idoc\_contrl-docnum.  
      idoc\_status-status = '51'.  
      idoc\_status-msgty = 'E'.  
      idoc\_status-msgid = 'ZE'.  
      idoc\_status-msgno = '005'.  
      idoc\_status-msgv1 = fs\_app\_empdet-ssn.  
      APPEND idoc\_status.  
      workflow\_result = c\_wf\_result\_error.  
      return\_variables-wf\_param = 'Error\_IDOCs'.  
      return\_variables-doc\_number = idoc\_contrl-docnum.  
      APPEND return\_variables.  
    ELSE.  
\* Data looks good. Create weekly report if it does not already exist  
\* If weekly report exists, then simply update the records  
      SELECT SINGLE \* FROM zempdetail INTO @DATA(ls\_det)  WHERE ssn = @fs\_app\_empdet-ssn.  
      IF sy-subrc NE 0.  
        INSERT INTO zempdetail VALUES fs\_app\_empdet.  
        INSERT zempwkdet FROM TABLE it\_app\_wkdet.  
      ELSE.  
        UPDATE zempdetail FROM fs\_app\_empdet.  
        UPDATE zempwkdet FROM TABLE it\_app\_wkdet.  
      ENDIF.  
      IF sy-subrc EQ 0.  
\* populate return variables for success  
        return\_variables-wf\_param = 'Processed\_IDOCs'.  
        return\_variables-doc\_number = idoc\_contrl-docnum.  
        return\_variables-wf\_param = 'Appl\_Objects'.  
        return\_variables-doc\_number = fs\_app\_empdet-ssn.  
        APPEND return\_variables.  
\* add status record indicating success  
        idoc\_status-docnum = idoc\_contrl-docnum.  
        idoc\_status-status = '53'.  
        idoc\_status-msgty = 'I'.  
        idoc\_status-msgid = 'ZE'.  
        idoc\_status-msgno = '006'.  
        idoc\_status-msgv1 = fs\_app\_empdet-ssn.  
        APPEND idoc\_status.  
      ELSE.  
\* populate return variables indicating error  
        workflow\_result = c\_wf\_result\_error.  
        return\_variables-wf\_param = 'Error\_IDOCs'.  
        return\_variables-doc\_number = idoc\_contrl-docnum.  
        APPEND return\_variables.  
\* add status record indicating failure in updating  
        idoc\_status-docnum = idoc\_contrl-docnum.  
        idoc\_status-status = '51'.  
        idoc\_status-msgty = 'E'.  
        idoc\_status-msgid = 'ZE'.  
        idoc\_status-msgno = '007'.  
        idoc\_status-msgv1 = fs\_app\_empdet-ssn.  
        APPEND idoc\_status.  
      ENDIF.  
    ENDIF.  
  
  ENDLOOP. "End loop at idoc\_contrl.  
  
ENDFUNCTION.

**Configuring an Inbound Process for New IDocs**

**Process Configuration in SDD-400.**

1. Create a new message type. (Already specified in the above steps Check Step 5)

2. Link the message type to the IDoc type.(Already specified in the above steps Step 6)

3. Allocate the function module to the logical message type.( Check Step 14)

4. Define the attributes for the inbound function module.

5. Create a new process code.

6. Assign input methods

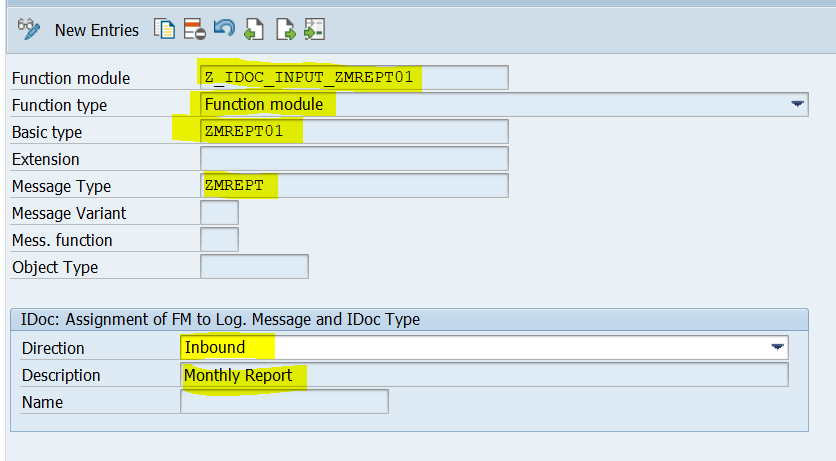
7. Create (or change) a partner profile

**Step 14: Allocate the Function Module to the Logical Message (only in SDD-400)**

Transaction: WE57

Go to Change mode.

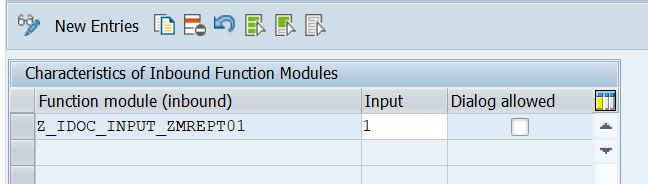
Click on New Entires



**Step 15: Define the Settings for the Inbound Function Module(only in SDD-400)**

Transaction: BD51

Click on change mode, And then New Entries.

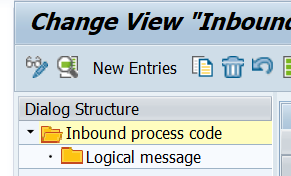


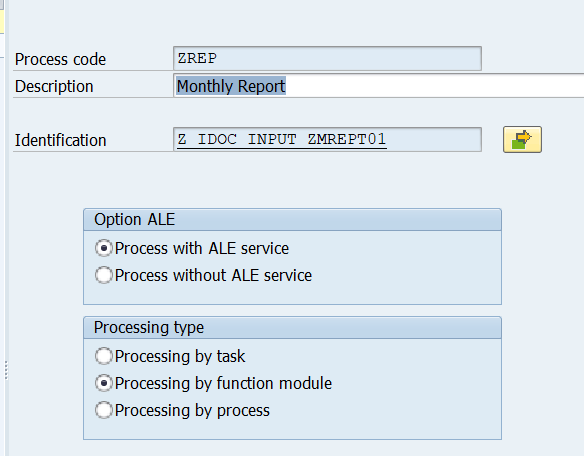
**Step 16:** **Create a New Process Code**

Transaction: WE42

This step defines a process code that points to the function module developed for the inbound process.

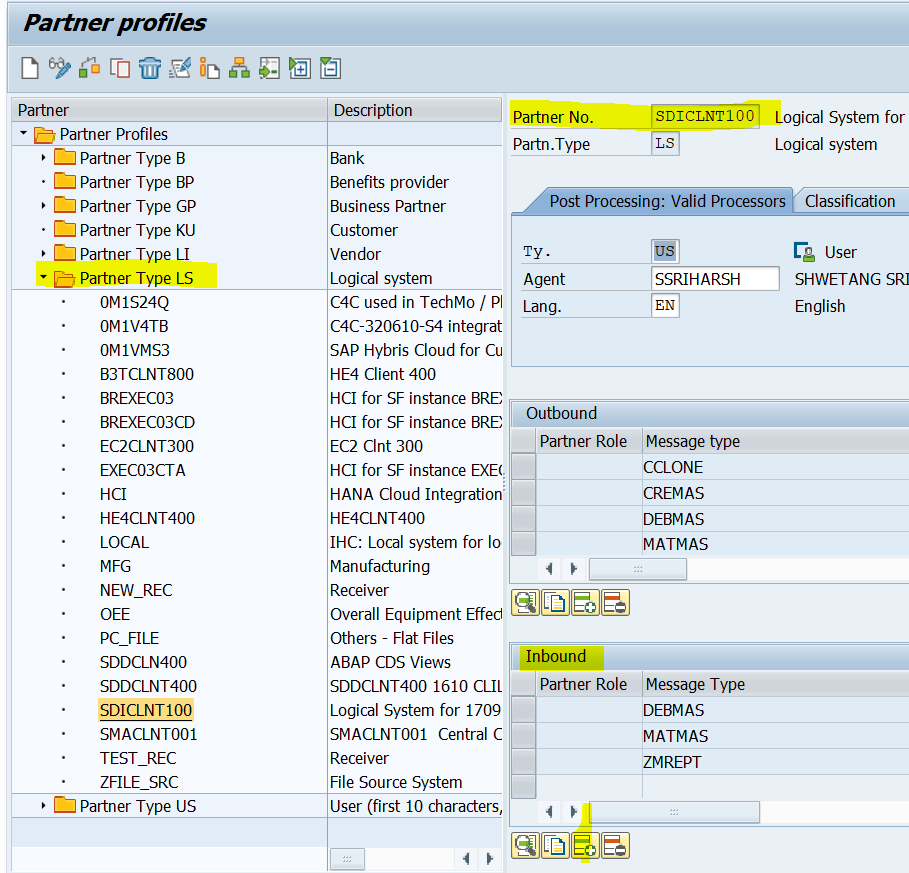
Click on change mode >> New Entries.

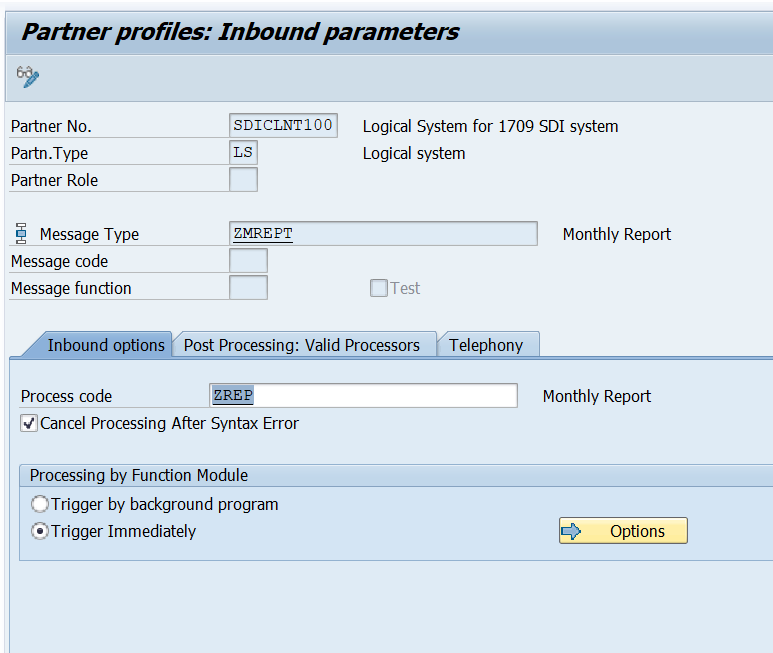




**Step 17 :** **Create or Change a partner profile in Receiving system (SDD-400)**

Click on logical system partner >> and for partner SDICLNT100 we configure the inbound IDoc Message.

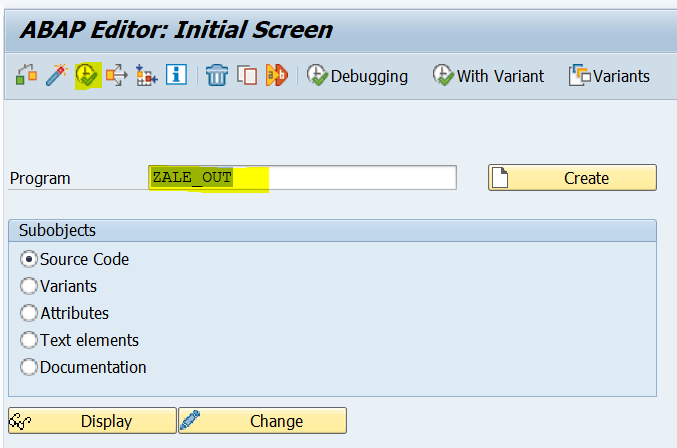


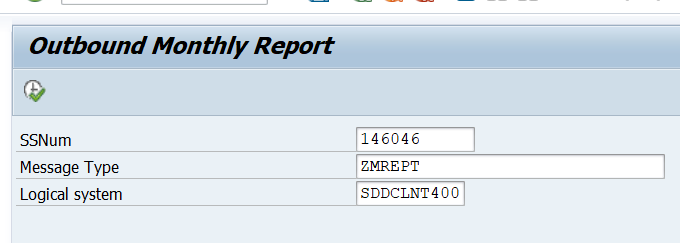


**Now all the Inbound and outbound configuration steps and Development is done.**

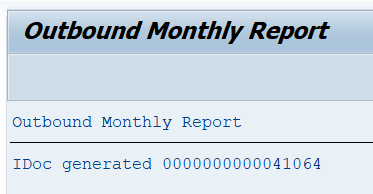
**Now we have to test the Distribution of Master Data.**

**Step 18: Execute the Report ZALE\_OUT.**



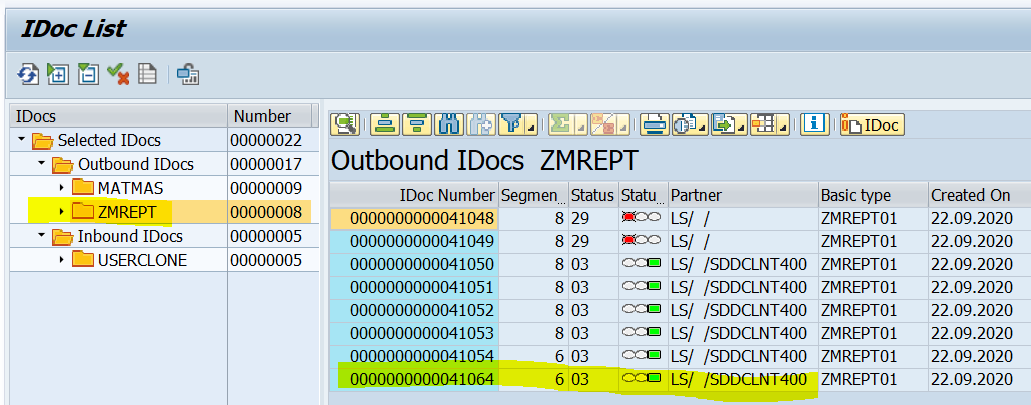


Upon successful execution, an IDoc will be created in the system SDI-100 (Outbound IDOC).

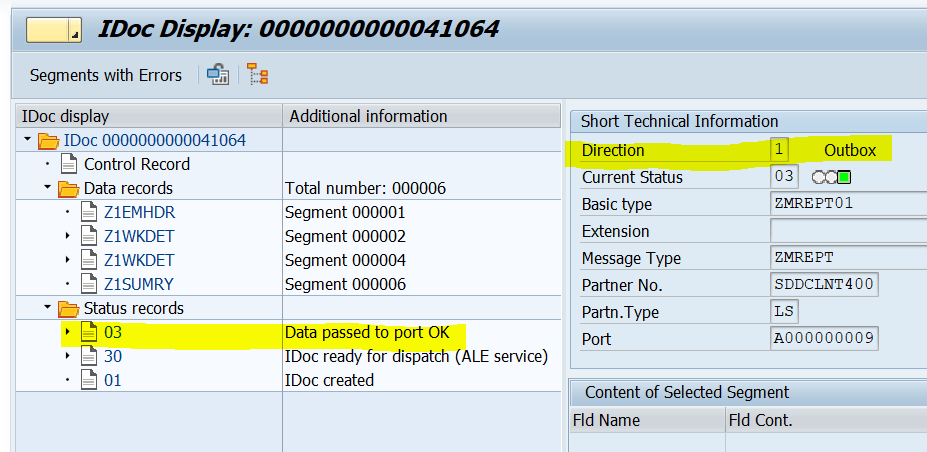


**Step 19: To check the Created Out Bound IDoc in Sender system (SDI-100)**

**Transaction : WE05 or WE02.**

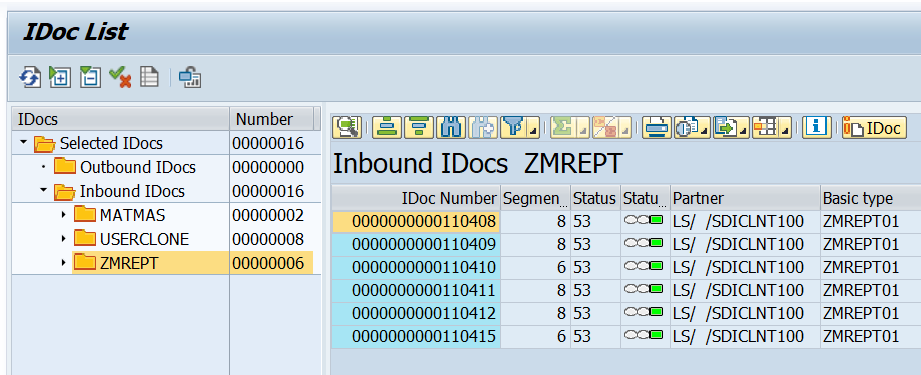


If the IDoc is transferred to other system, then it will be in status 03. (if in partner profile, the IDoc is in set to Send immediately). If the partner profile is set to collect IDocs, the we have to run the program RSEOUT00, to release the IDocs.

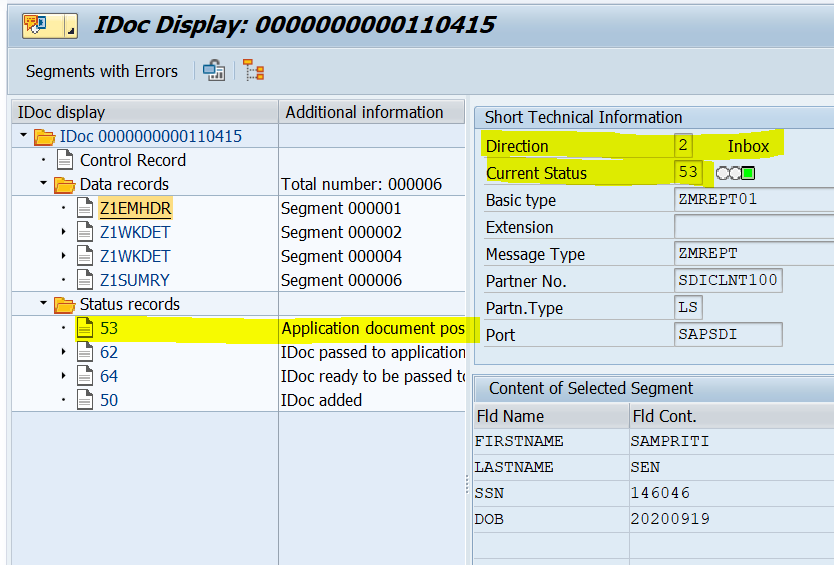


**Step 20: To check the Created Inbound IDoc in Receiver system (SDD-400)**

**Transaction : WE05 or WE02.**



**If the IDoc is in Status 53 then the Inbound IDoc is posted successfully.**



**If it is in Error, it shows a status of 51.**

**And the reprocessing can be done using BD87. We can keep a break point in inbound function module and check the causes of error.**

**Also we can use the test tool WE19. Here a copy of IDoc is create and we will have a scope to change the data and check the posting. Debuggin is also possible here.**

**Step 21: Check the Table in SDD 400 system to see if the data is updated in the 2 Tables.**