**Custom IDOC- outbound**

The term IDoc stands for intermediate document. It is simply a data container used to exchange information between any two processes that can understand the semantics of the data. When an IDoc is generated in the system, a unique number is assigned to it via a Number Range Object.

In this document, we’ll be creating ‘*Custom IDOC’* for Purchase Order and same will be triggered for K2 Specific Purchase Orders if there is any change in SAP.

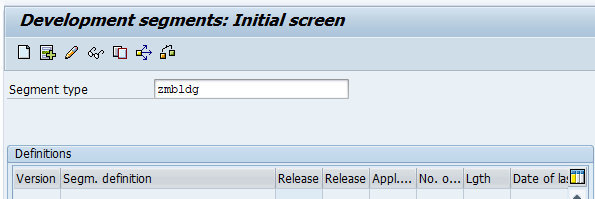
Interface

|  |  |  |
| --- | --- | --- |
| RFC | Web Service | IDOC |
|  | Consume | EDI |
|  | Provide | Change Pointer |

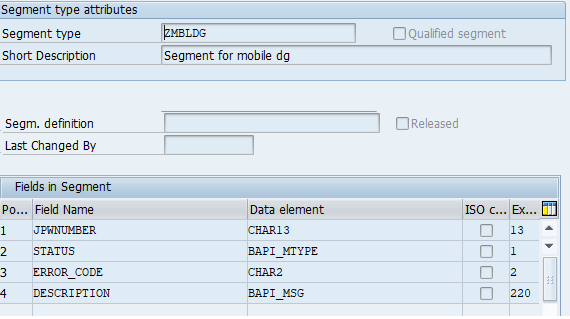
**Create Segment:**

A segment defines the format and structure of a data record. Segments are reusable components, which means they can be used in more than one IDoc type. A segment consists of various fields that represent data in a data record.

Steps to create a segment: GoTo the **TCode:** **WE31** 🡪 Provide name for Segment type 🡪 Create 🡪 You’ll get following screen:

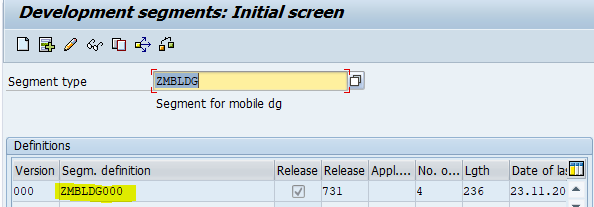


Enter the description 🡪 Provide the required fields and Data elements in the Segment 🡪 Save

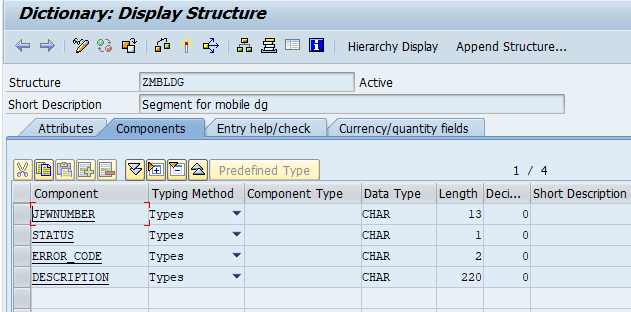


After creating the segment, segment definition and equivalent structure is created in the DDIC automatically.

Segment Definition:



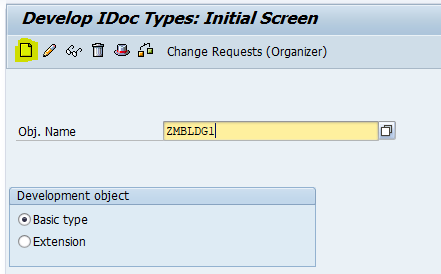
Auto generated structure for Segment in SE11:

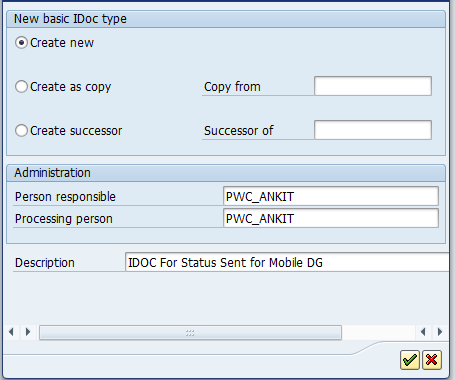


**Create IDOC:**

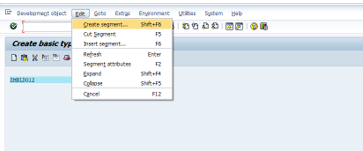
IDOC is the collection of segments. Each segment is the collection of fields.

GoTo the **TCode: WE30** 🡪 Enter IDOC name 🡪 Create



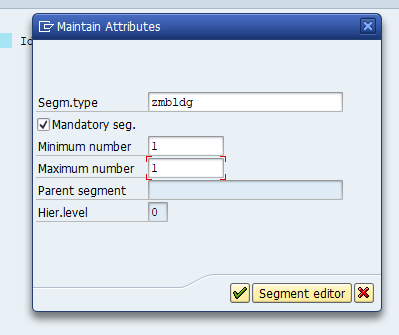


Assign Segment to IDOC:



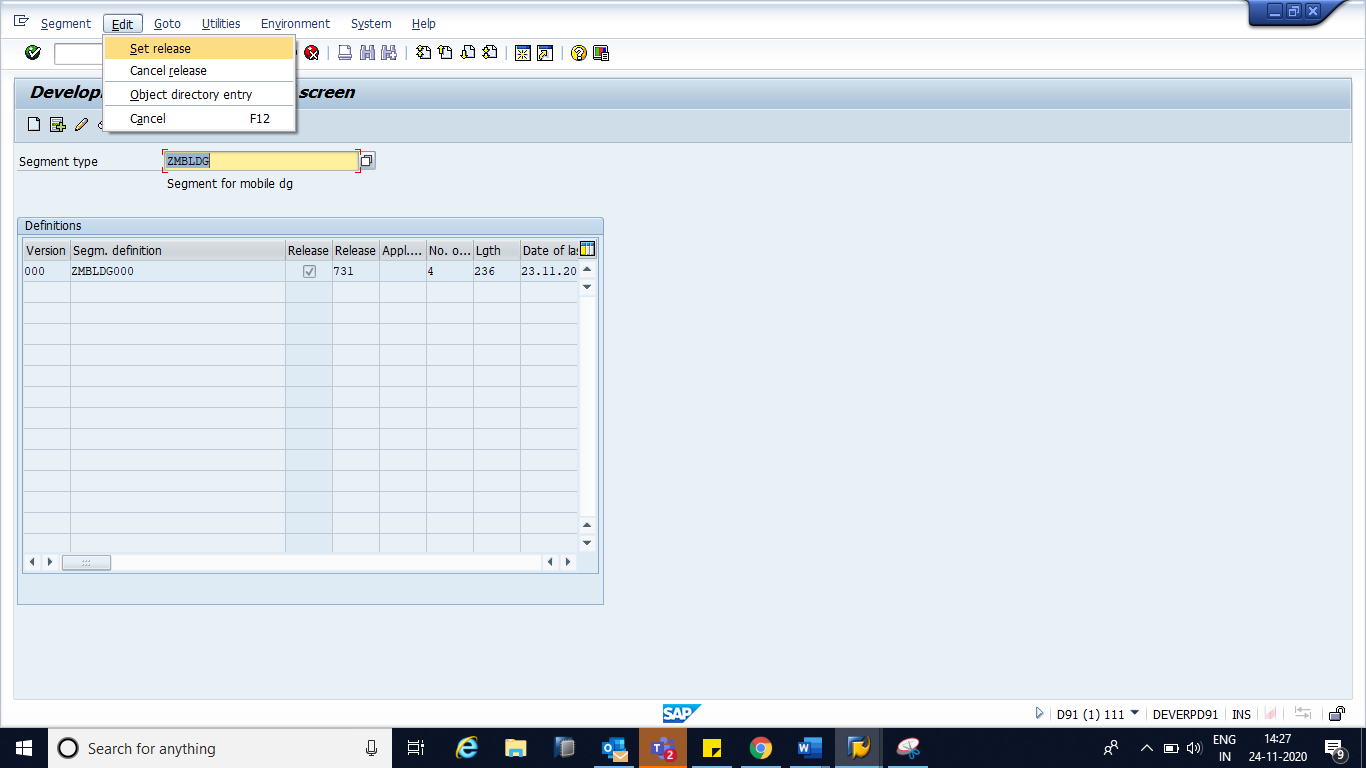
Whenever you release a segment (in WE31) a function module called 'SEGMENTDEFINITION\_CLOSE' is called. In this function module a series of useful checks are implemented.

We can make whole segment mandatory or optional, but we can’t make any particular field of segment as mandatory.

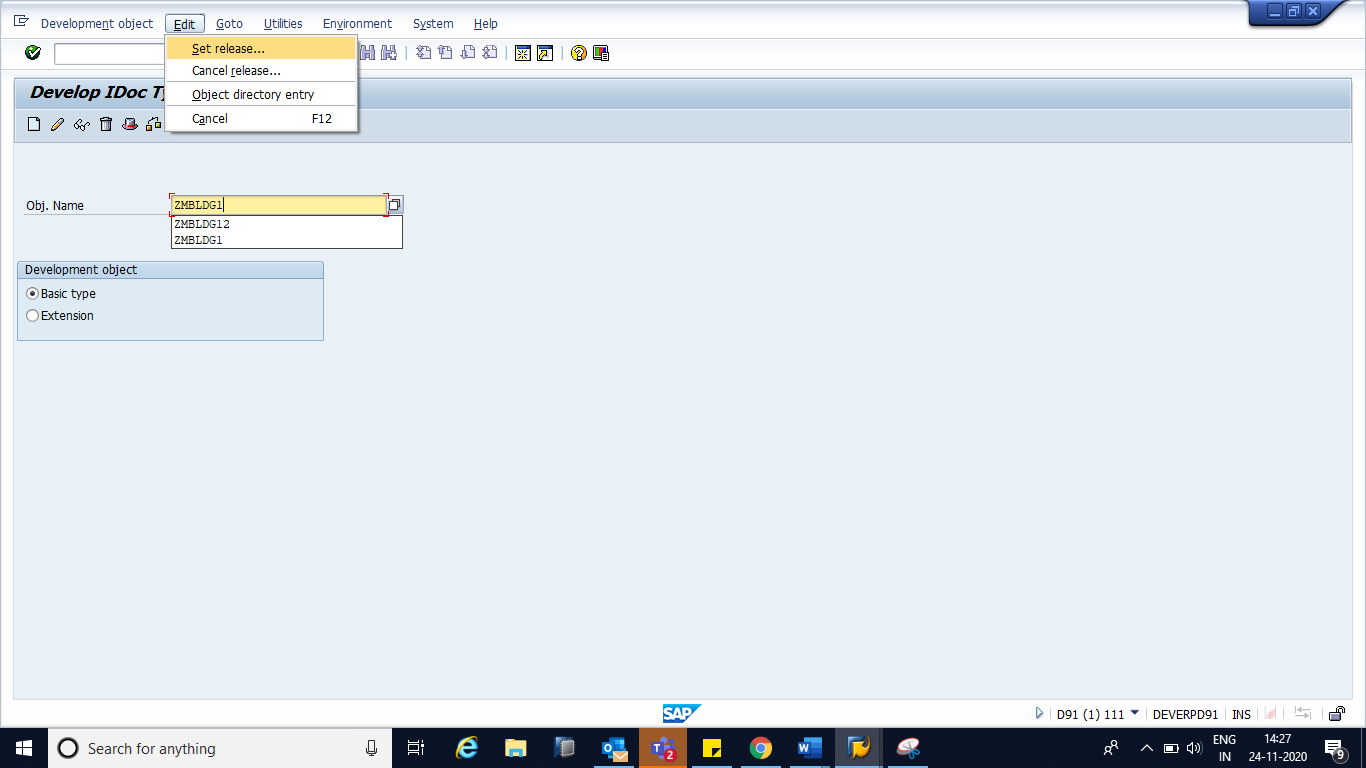


**Now Release the Segment and IDOC:**

Releasing Segment: WE31 🡪 Enter Segment name 🡪 Edit 🡪 Set release

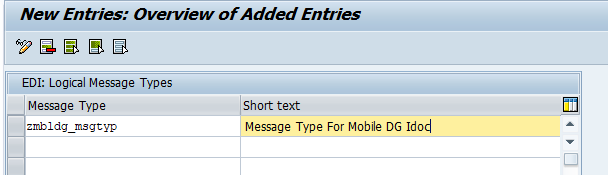


Releasing IDOC: WE30 🡪 Enter IDOC name 🡪 Edit 🡪 Set release



**Create Message Type:**

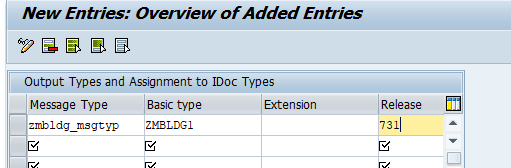
GoTo **TCode: WE81** 🡪 Edit Mode 🡪 New Entries 🡪 Enter Message type and description



Message type describes the structure of a message. Table: EDIMSG.

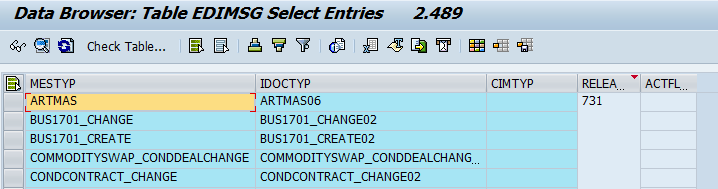
**Link the Message Type to IDOC:**

GoTo **TCode: WE82** 🡪 Edit Mode 🡪 New Entries 🡪 Enter Message Type, Basic Type (IDOC Name) & Release:



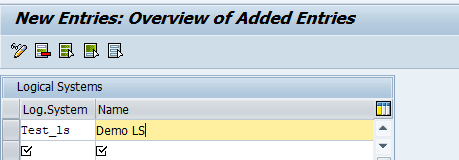
***Steps to identify release:***

Sort RELEASED field of Table: EDIMSG in descending format and pick the latest release.



**Define logical system-**

GoTo **TCode: SALE** 🡪 Basic Settings 🡪 Logical Systems 🡪 Define Logical System 🡪 Edit mode 🡪 New entries 🡪 Save:



Generally we use Existing LS if it is already configured.

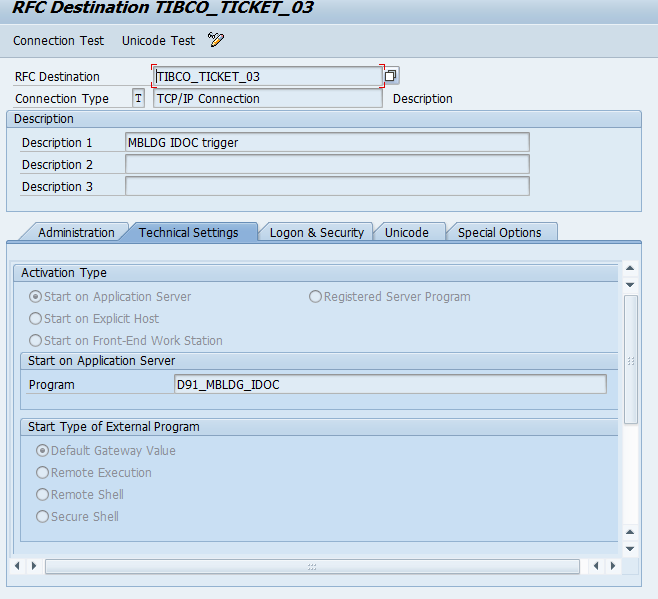
Logical system' is used to identify an individual client in a system.

**Maintain RFC destination (TCP/IP Connections):** **Need to do in Test client**

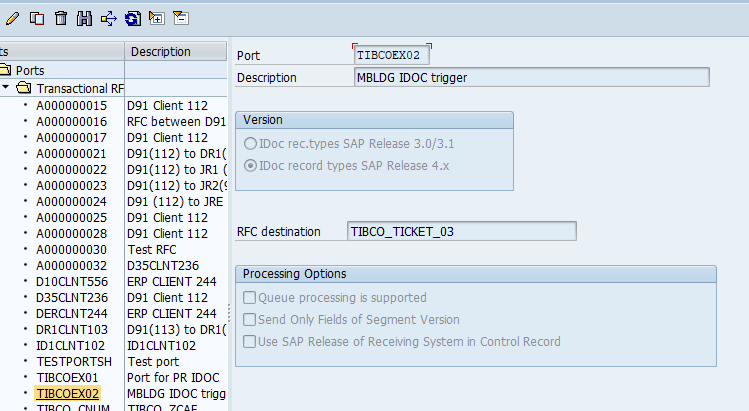
Provide Program\_ID to Tibco Team and they will create

1. RFC destination ie. TIBCO\_TICKET\_03

Under RFC Destination Program id is configured



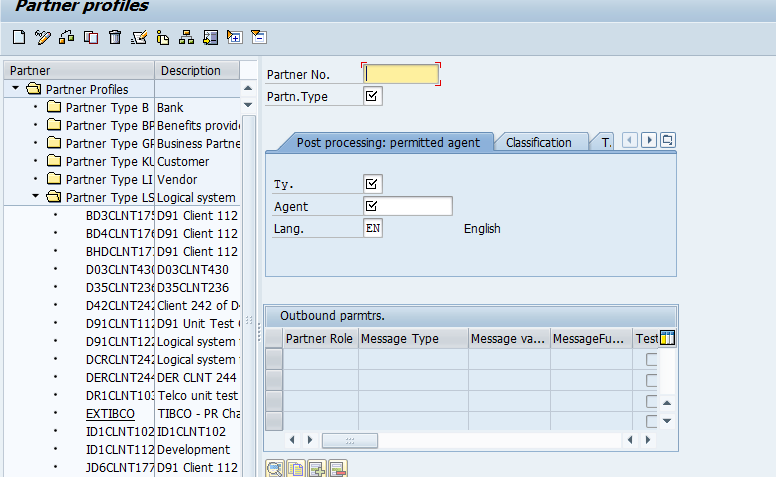
Port is created by technical team where we have to configure RFC destination which is created by TIBCO team ie. TIBCOEX02



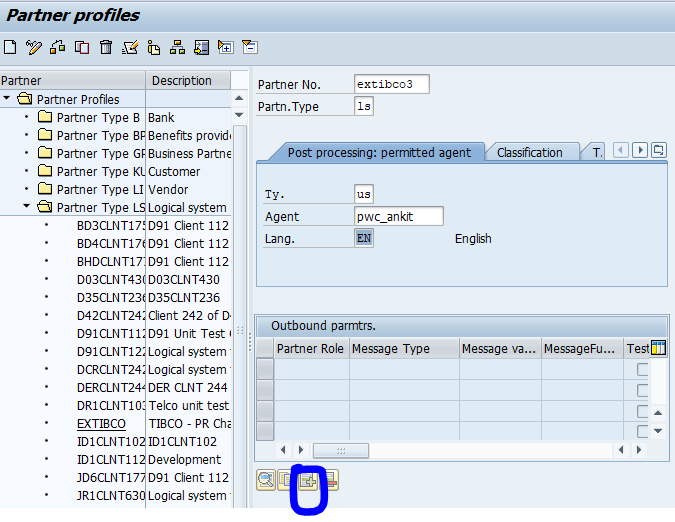
When the SAP system needs to transfer IDocs - it needs to know how to communicate with the other system. RFC destination of the communicating system is stored in ports and IDoc communication occurs through ports. Type of port defines the type of other system with which SAP communicates and how it can communicate.

**Create Partner Profile:**

GoTo TCode: WE20 🡪 Partner Profiles 🡪Select Partner Type LS 🡪 Create:



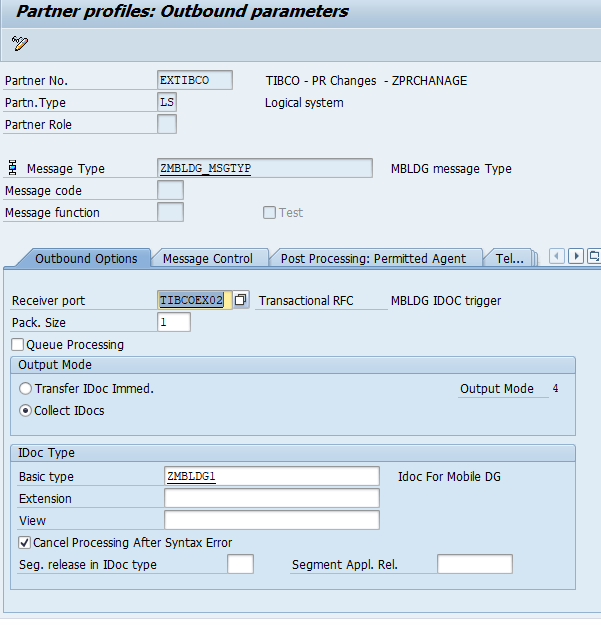
After Saving 🡪 We can add the details of Outbound Parameters 🡪 Click on ‘+’ button:



Below screen will be appear 🡪 Provide required details 🡪 Save:

We need to provide

1. Message type
2. Receiver port
3. Pack size
4. Basic Type (IDOC NAME)



Now IDOC Configuration is completed.

**Code for custom Outbound IDOC:**

*Fill Master IDOC: Filling the master IDOC is nothing but filling the internal table which contain two fields:*

Segment name (SEGNAM)

Segment data (SDATA)

*Collect the control record information*

Collect the control record information is nothing but filling the internal table which contains the following the fields (these can be extracted from Table: EDP13)

1. RCVPOR—Receiver port
2. RCVPRT- Receiver partner type
3. DOCTYP—IDOC type
4. MESTYP—message type
5. RCVPRN---receiver partner number

EDIDC table contains the above fields so we simply declare our control record internal table by referring EDIDC.

**MASTER\_IDOC\_DISTRIBUTE** is a function module to trigger IDOC.

**Fill MASTER\_IDOC\_CONTROL:**

TYPES : BEGIN OF ty\_edp13,  
                  rcvprn TYPE edp13-rcvprn,  
                  rcvprt TYPE edp13-rcvprt,  
                  rcvpor TYPE edp13-rcvpor,  
                  END OF ty\_edp13.  
  DATA : gt\_idoc\_data    TYPE edidd\_tt,   *"Data Record - Segment Data and Name*  
         gw\_idoc\_data    TYPE edidd,  
         gt\_idoc\_comm    TYPE STANDARD TABLE OF edidc, *"Returns IDOC No.*  
         gw\_edp13        TYPE ty\_edp13,  
         gv\_logsys       TYPE logsys,  
         gw\_idoc\_control TYPE edidc,  
         gw\_mbldg        TYPE zmbldg.  
  CONSTANTS : gc\_segnam TYPE edilsegtyp VALUE 'ZMBLDG', *"Segment Name*  
              gc\_sndprt TYPE edi\_sndprt VALUE 'LS'.  
  
  FIELD-SYMBOLS : <fs\_idoc\_comm> TYPE edidc.  
  DATA :  lt\_idoc\_comm     TYPE STANDARD TABLE OF edidc.

 CLEAR : gw\_edp13.  
          SELECT rcvprn  
                 rcvprt  
                 rcvpor  
            FROM edp13 INTO gw\_edp13  
            UP TO 1 ROWS  
            WHERE rcvprn  EQ 'EXTIBCO'   *"Logical System, Partner No.*  
              AND rcvprt  EQ 'LS'  
              AND mestyp  EQ 'ZMBLDG\_MSGTYP'   *"Message Type*  
              AND rcvpor  EQ 'TIBCOEX02'  *"PORT*  
              AND idoctyp EQ 'ZMBLDG1'. *"IDoc Type*  
          ENDSELECT.  
          IF sy-subrc IS INITIAL.  
            CLEAR : gv\_logsys.  
            SELECT SINGLE logsys  
              FROM t000  
              INTO gv\_logsys  
              WHERE mandt EQ sy-mandt.  
            IF sy-subrc IS NOT INITIAL.  
              CLEAR : gv\_logsys.  
            ENDIF.  
          ENDIF.  
          gw\_idoc\_control-sndprn = gv\_logsys.  
          gw\_idoc\_control-sndprt = gc\_sndprt.  
          gw\_idoc\_control-rcvprn = gw\_edp13-rcvprn.  
          gw\_idoc\_control-rcvprt = gw\_edp13-rcvprt.  
          gw\_idoc\_control-rcvpor = gw\_edp13-rcvpor.  
          gw\_idoc\_control-doctyp = 'ZMBLDG1'.  
          gw\_idoc\_control-mestyp = 'ZMBLDG\_MSGTYP'.  
  
          REFRESH : gt\_idoc\_data[].  
          gw\_mbldg-description     = wa\_return-message .  
          gw\_mbldg-error\_code      = wa\_return-id.  
          gw\_mbldg-jpwnumber       = wa\_ogpin-jpwnumber.  
          gw\_mbldg-status          = wa\_return-type.  
  
          gw\_idoc\_data-segnam = gc\_segnam.  
          gw\_idoc\_data-sdata  = gw\_mbldg.  
          APPEND gw\_idoc\_data TO gt\_idoc\_data.  
  
  
          CALL FUNCTION 'MASTER\_IDOC\_DISTRIBUTE'  
            EXPORTING  
              master\_idoc\_control            = gw\_idoc\_control  
            TABLES  
              communication\_idoc\_control     = gt\_idoc\_comm  
              master\_idoc\_data               = gt\_idoc\_data  
            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 IS INITIAL.  
COMMIT WORK.  
            UNASSIGN <fs\_idoc\_comm>.  
            LOOP AT gt\_idoc\_comm ASSIGNING <fs\_idoc\_comm> WHERE docnum IS NOT INITIAL.  
              CALL FUNCTION 'EDI\_DOCUMENT\_DEQUEUE\_LATER'  
                EXPORTING  
                  docnum                 = <fs\_idoc\_comm>-docnum  
                EXCEPTIONS  
                  idoc\_is\_not\_to\_dequeue = 1  
                  OTHERS                 = 2.  
              IF sy-subrc IS INITIAL.  
COMMIT WORK.  
*\*         Do Nothing*  
              ENDIF.  
            ENDLOOP.  
            UNASSIGN <fs\_idoc\_comm>.  
          ENDIF.

After executing, Table: ***lt\_idoc\_comm*** will have generated IDOC No.

Execute TCode: WE02 for the generated IDOCs:



Double click on IDOC No. to see the Data Records:

