SAP TM – Report POD event in the Freight orders and update the POS status in the outbound deliveries and also revoke already reported POD event in case of error.

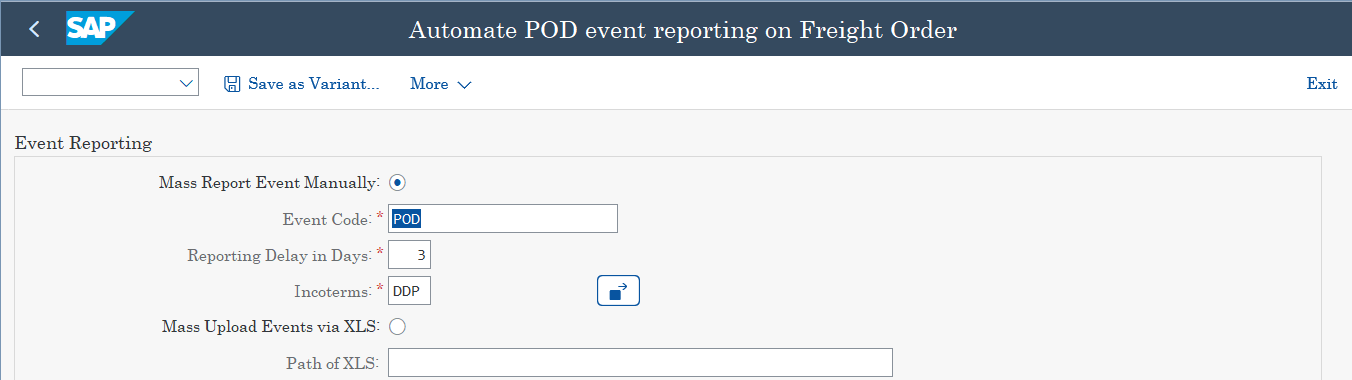
\*----------------------------------------------------------------------\*  
\*RICEFW ID          :                                                  \*  
\*Title              : Automate POD event reporting on Freight Order    \*  
\*Author             : EXT-AGOEL                                        \*  
\*Creation Date      : 15.03.2024                                       \*  
\*Package            : ZTM01                                            \*  
\*Transaction Code   :  ZTM\_AUTO\_POD\_EVENT                               \*  
\*Transport Request  : DS4K902314                                       \*  
\*Description        : Automate POD event reporting on Freight Order    \*  
\*----------------------------------------------------------------------\*  
\*                         CHANGE HISTORY                               \*  
\*----------------------------------------------------------------------\*  
\*Date       | Author  | Transport |Change ID - Description             \*                                                                     \*  
\*DD-MMM-YYYY|          |            |                                     \*  
\*----------------------------------------------------------------------\*  
REPORT ztmr\_automate\_pod\_event\_report.  
  
TYPES:BEGIN OF ty\_output,  
        freight\_order    TYPE /scmtms/tor\_id,  
        event\_code\_descp TYPE /scmtms/description\_s,  
        event\_timestamp  TYPE timestamp,  
        location         TYPE /scmtms/location\_id,  
        remarks          TYPE string,  
      END OF   ty\_output.  
  
TYPES:BEGIN OF ty\_excel\_data,  
        freight\_order(20) TYPE c,  
        event\_code(20)    TYPE c,  
        event\_date(10)    TYPE c,  
        event\_time(8)     TYPE c,  
      END OF ty\_excel\_data.  
  
TYPES:BEGIN OF ty\_event\_update,  
        fo\_key          TYPE /bobf/conf\_key,  
        fo\_id           TYPE /scmtms/tor\_id,  
        exe\_key         TYPE /bobf/conf\_key,  
        event\_timestamp TYPE timestamp,  
      END OF ty\_event\_update.  
  
DATA:lv\_incoterm         TYPE inco1,  
     gv\_event\_code\_descp TYPE /scmtms/description\_s.  
  
DATA:gt\_output       TYPE STANDARD TABLE OF ty\_output,  
     gt\_excel\_data   TYPE STANDARD TABLE OF ty\_excel\_data,  
     gt\_event\_update TYPE STANDARD TABLE OF ty\_event\_update.  
  
\*-----------------------------------------------------------------------  
\*Selection Screen  
\*-----------------------------------------------------------------------  
SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME TITLE TEXT-001.  
  PARAMETERS:p\_manual RADIOBUTTON GROUP g1,  
             p\_evtcd  TYPE /scmtms/c\_ev\_ty-tor\_event OBLIGATORY DEFAULT 'POD',  
             p\_days   TYPE anzta OBLIGATORY DEFAULT '3'.  
  SELECT-OPTIONS:s\_inctrm FOR lv\_incoterm OBLIGATORY NO INTERVALS.  
  PARAMETERS:p\_bulkup RADIOBUTTON GROUP g1.  
  PARAMETERS:p\_file TYPE rlgrap-filename.  
SELECTION-SCREEN END OF BLOCK b1.  
  
AT SELECTION-SCREEN ON VALUE-REQUEST FOR p\_file.  
  CALL FUNCTION 'F4\_FILENAME'  
    EXPORTING  
      field\_name = 'P\_FILE'  
    IMPORTING  
      file\_name  = p\_file.  
  
AT SELECTION-SCREEN OUTPUT.  
  s\_inctrm = VALUE #( sign   = 'I'  
                      option = 'EQ'  
                      low    = 'DDP' ).  
  APPEND s\_inctrm.  
  
  s\_inctrm = VALUE #( sign   = 'I'  
                      option = 'EQ'  
                      low    = 'DAP' ).  
  APPEND s\_inctrm.  
  
  s\_inctrm = VALUE #( sign   = 'I'  
                      option = 'EQ'  
                      low    = 'DPU' ).  
  APPEND s\_inctrm.  
  
\*-----------------------------------------------------------------------  
\*Processing Logic  
\*-----------------------------------------------------------------------  
  INCLUDE ztmi\_automate\_pod\_event\_report.  
  
\*-----------------------------------------------------------------------  
\*START-OF-SELECTION.  
\*-----------------------------------------------------------------------  
START-OF-SELECTION.  
  
  DATA:lo\_automate\_pod\_event\_report TYPE REF TO lcl\_automate\_pod\_event\_report.  
  
  lo\_automate\_pod\_event\_report = NEW #( ).  
  
  IF lo\_automate\_pod\_event\_report IS BOUND.  
  
    TRY.  
\*-----------------------------------------------------------------------  
\*Get Frieght orders  
\*-----------------------------------------------------------------------  
        IF p\_manual EQ abap\_true.  
          lo\_automate\_pod\_event\_report->get\_freight\_orders\_man( IMPORTING et\_fo\_key = DATA(lt\_fo\_create) ).  
        ELSE.  
          lo\_automate\_pod\_event\_report->get\_freight\_orders\_xls( IMPORTING et\_fo\_create = lt\_fo\_create  
                                                                          et\_fo\_update = DATA(lt\_fo\_update) ).  
        ENDIF.  
  
\*-----------------------------------------------------------------------  
\*Report POD event on the Freight Orders  
\*-----------------------------------------------------------------------  
        lo\_automate\_pod\_event\_report->report\_pod\_event( EXPORTING it\_fo\_key = lt\_fo\_create ).  
  
\*-----------------------------------------------------------------------  
\*Update POD event on the Freight Orders  
\*-----------------------------------------------------------------------  
        lo\_automate\_pod\_event\_report->update\_pod\_event( EXPORTING it\_fo\_update = lt\_fo\_update ).  
  
\*-----------------------------------------------------------------------  
\*Display event reporting output  
\*-----------------------------------------------------------------------  
        lo\_automate\_pod\_event\_report->display\_event\_output( ).  
  
      CATCH /bobf/cx\_frw\_contrct\_violation INTO DATA(lo\_exception).  
        DATA(ls\_message) = lo\_exception->if\_t100\_message~t100key.  
        MESSAGE ID ls\_message-msgid TYPE sy-msgty NUMBER ls\_message-msgno  
        WITH ls\_message-attr1 ls\_message-attr2 ls\_message-attr3 ls\_message-attr4.  
    ENDTRY.  
  
  ENDIF.  
\*-----------------------------------------------------------------------

\*Code of INCLUDE ztmi\_automate\_pod\_event\_report.

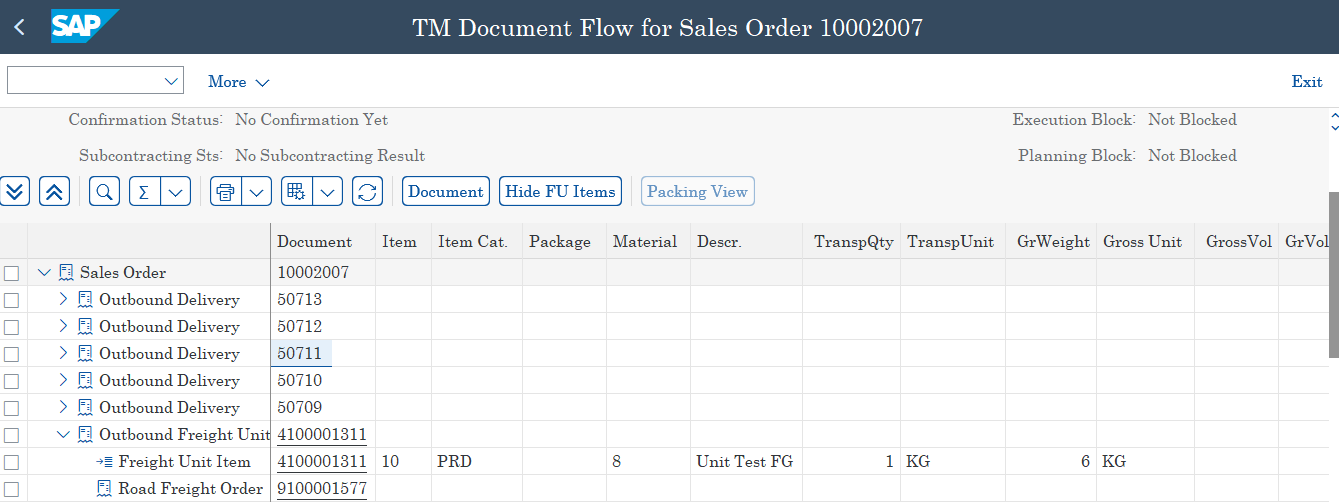
\*&---------------------------------------------------------------------\*  
\*& Include ZTMI\_AUTOMATE\_POD\_EVENT\_REPORT  
\*&---------------------------------------------------------------------\*  
  
CLASS lcl\_automate\_pod\_event\_report DEFINITION FINAL.  
  
  PUBLIC SECTION.  
  
    METHODS:get\_freight\_orders\_man EXPORTING et\_fo\_key TYPE /bobf/t\_frw\_key RAISING /bobf/cx\_frw\_contrct\_violation.  
  
    METHODS:get\_freight\_orders\_xls EXPORTING et\_fo\_create TYPE /bobf/t\_frw\_key  
                                             et\_fo\_update LIKE gt\_event\_update.  
  
    METHODS:report\_pod\_event IMPORTING it\_fo\_key TYPE /bobf/t\_frw\_key RAISING /bobf/cx\_frw\_contrct\_violation.  
  
    METHODS:create\_pod\_event IMPORTING iv\_fo\_key          TYPE /bobf/conf\_key  
                                       iv\_last\_stop\_key   TYPE /bobf/conf\_key  
                                       iv\_evt\_timestamp   TYPE timestamp  
                             EXPORTING ev\_event\_timestamp TYPE timestamp  
                                       et\_failed\_key      TYPE /bobf/t\_frw\_key  
                                       eo\_message         TYPE REF TO /bobf/if\_frw\_message.  
  
    METHODS:update\_pod\_event IMPORTING it\_fo\_update LIKE gt\_event\_update.  
  
    METHODS:revoke\_pod\_event IMPORTING it\_fo\_key      TYPE /bobf/t\_frw\_key  
                                       io\_srv\_manager TYPE REF TO /bobf/if\_tra\_service\_manager  
                                       io\_tra\_manager TYPE REF TO /bobf/if\_tra\_transaction\_mgr  
                             EXPORTING et\_failed\_key  TYPE /bobf/t\_frw\_key  
                                       eo\_message     TYPE REF TO /bobf/if\_frw\_message.  
  
    METHODS:update\_pod\_status IMPORTING is\_item\_node\_data TYPE /scmtms/s\_tor\_item\_tr\_k  
                              EXPORTING ev\_outbound\_delv  TYPE vbeln\_vl  
                                        et\_prot           TYPE tab\_prott.  
  
    METHODS:val\_pod\_status\_upd IMPORTING is\_item\_node\_data TYPE /scmtms/s\_tor\_item\_tr\_k  
                               EXPORTING ev\_outbound\_delv  TYPE vbeln\_vl  
                                         et\_prot           TYPE tab\_prott.  
  
    METHODS:display\_event\_output.  
  
    METHODS:upload\_excel\_file.  
  
ENDCLASS.  
  
CLASS lcl\_automate\_pod\_event\_report IMPLEMENTATION.  
  
  METHOD get\_freight\_orders\_man.  
  
    DATA:lo\_srv\_tor TYPE REF TO /bobf/if\_tra\_service\_manager,  
         lo\_message TYPE REF TO /bobf/if\_frw\_message.  
  
    DATA:lv\_current\_timestamp TYPE timestamp,  
         lv\_from\_timestamp    TYPE timestamp,  
         lv\_to\_timestamp      TYPE timestamp,  
         lv\_from\_date         TYPE sy-datum,  
         lv\_to\_date           TYPE sy-datum.  
  
    DATA:ls\_query\_info TYPE /bobf/s\_frw\_query\_info.  
  
    DATA:lt\_sel\_parameters TYPE /bobf/t\_frw\_query\_selparam,  
         lt\_stop\_node\_key  TYPE  /bobf/t\_frw\_key,  
         lt\_root\_node\_key  TYPE  /bobf/t\_frw\_key,  
         lt\_stop\_node\_data TYPE /scmtms/t\_tor\_stop\_k,  
         lt\_failed\_key     TYPE /bobf/t\_frw\_key,  
         lt\_d\_exec\_tr      TYPE /scmtms/t\_tor\_exec\_tr\_k,  
         lt\_item\_node\_data TYPE /scmtms/t\_tor\_item\_tr\_k.  
  
    DATA:lr\_execution\_key TYPE RANGE OF /bobf/s\_frw\_key-key,  
         lr\_item\_key      TYPE RANGE OF /bobf/s\_frw\_key-key.  
  
    CLEAR et\_fo\_key.  
  
\*Instantiate service manager  
    lo\_srv\_tor = /bobf/cl\_tra\_serv\_mgr\_factory=>get\_service\_manager( /scmtms/if\_tor\_c=>sc\_bo\_key ).  
  
\*-----------------------------------------------------------------------------------------------  
\*Filter Freight Orders with stop category as 'I' and also where the arrival date is 3-4 days old  
\*-----------------------------------------------------------------------------------------------  
  
\*Prepare selection creteria  
    lt\_sel\_parameters = VALUE #( ( attribute\_name = /scmtms/if\_tor\_c=>sc\_query\_attribute-stop-query\_by\_attributes-stop\_cat  
                                   option = 'EQ'  
                                   sign   = 'I'  
                                   low    = 'I') ).  
  
    GET TIME STAMP FIELD lv\_current\_timestamp.  
    CONVERT TIME STAMP lv\_current\_timestamp TIME ZONE 'UTC' INTO DATE DATA(lv\_date) TIME DATA(lv\_time).  
  
    lv\_from\_date = lv\_date - ( p\_days + 1 ).  
    CONVERT DATE lv\_from\_date TIME lv\_time DAYLIGHT SAVING TIME 'X' INTO TIME STAMP lv\_from\_timestamp TIME ZONE 'UTC'.  
  
    lv\_to\_date   = lv\_date - p\_days.  
    CONVERT DATE lv\_to\_date TIME lv\_time DAYLIGHT SAVING TIME 'X' INTO TIME STAMP lv\_to\_timestamp TIME ZONE 'UTC'.  
  
    APPEND VALUE #( attribute\_name = /scmtms/if\_tor\_c=>sc\_query\_attribute-stop-query\_by\_attributes-plan\_trans\_time  
                    option = 'LT'  
                    sign   = 'E'  
                    low    = lv\_from\_timestamp ) TO lt\_sel\_parameters.  
  
    APPEND VALUE #( attribute\_name = /scmtms/if\_tor\_c=>sc\_query\_attribute-stop-query\_by\_attributes-plan\_trans\_time  
                    option = 'GT'  
                    sign   = 'E'  
                    low    = lv\_to\_timestamp ) TO lt\_sel\_parameters.  
  
\*Get key of the selected records as per the selection creteria  
    lo\_srv\_tor->query(  
      EXPORTING  
        iv\_query\_key            = /scmtms/if\_tor\_c=>sc\_query-stop-query\_by\_attributes  
        it\_selection\_parameters = lt\_sel\_parameters  
      IMPORTING  
        eo\_message              = lo\_message  
        es\_query\_info           = ls\_query\_info  
        et\_key                  = lt\_stop\_node\_key ).  
  
\*Get data of the selected records as per the selection creteria  
    IF lt\_stop\_node\_key  IS NOT INITIAL.  
      lo\_srv\_tor->retrieve( EXPORTING iv\_node\_key   = /scmtms/if\_tor\_c=>sc\_node-stop  
                                      it\_key        = lt\_stop\_node\_key  
                                      iv\_edit\_mode  = /bobf/if\_conf\_c=>sc\_edit\_read\_only  
                            IMPORTING eo\_message    = lo\_message  
                                      et\_data       = lt\_stop\_node\_data  
                                      et\_failed\_key = lt\_failed\_key ).  
    ENDIF.  
  
    LOOP AT lt\_stop\_node\_data ASSIGNING FIELD-SYMBOL(<lfs\_stop\_node\_data>).  
      APPEND INITIAL LINE TO lt\_root\_node\_key ASSIGNING FIELD-SYMBOL(<lfs\_root\_node\_key>).  
      IF <lfs\_root\_node\_key> IS ASSIGNED.  
        <lfs\_root\_node\_key>-key = <lfs\_stop\_node\_data>-root\_key.  
      ENDIF.  
    ENDLOOP.  
  
\*--------------------------------------------------------------------------------------  
\*Filter Freight Orders where no POD event reported  
\*--------------------------------------------------------------------------------------  
  
    IF lt\_root\_node\_key IS NOT INITIAL.  
      lo\_srv\_tor->retrieve\_by\_association(  
        EXPORTING  
          iv\_node\_key    = /scmtms/if\_tor\_c=>sc\_node-root  
          it\_key         = lt\_root\_node\_key  
          iv\_association = /scmtms/if\_tor\_c=>sc\_association-root-executioninformation\_tr  
          iv\_fill\_data   = abap\_true  
        IMPORTING  
          eo\_message     = lo\_message  
          et\_data        = lt\_d\_exec\_tr  
          et\_failed\_key  = lt\_failed\_key ).  
  
      DELETE lt\_d\_exec\_tr WHERE event\_code NE 'POD'.  
  
      IF lt\_d\_exec\_tr IS NOT INITIAL.  
        lr\_execution\_key = VALUE #( FOR ls\_d\_exec\_tr IN lt\_d\_exec\_tr  
                                   ( sign           = 'I'  
                                     option         = 'EQ'  
                                     low            = ls\_d\_exec\_tr-root\_key ) ).  
  
        DELETE lt\_root\_node\_key WHERE key NOT IN lr\_execution\_key.  
      ENDIF.  
    ENDIF.  
  
\*--------------------------------------------------------------------------------------  
\*Filter Freight Orders by Incoterm  
\*--------------------------------------------------------------------------------------  
    IF lt\_root\_node\_key IS NOT INITIAL.  
      lo\_srv\_tor->retrieve\_by\_association(  
        EXPORTING  
          iv\_node\_key    = /scmtms/if\_tor\_c=>sc\_node-root  
          it\_key         = lt\_root\_node\_key  
          iv\_association = /scmtms/if\_tor\_c=>sc\_association-root-item\_tr  
          iv\_fill\_data   = abap\_true  
        IMPORTING  
          eo\_message     = lo\_message  
          et\_data        = lt\_item\_node\_data  
          et\_failed\_key  = lt\_failed\_key ).  
  
      DELETE lt\_item\_node\_data WHERE item\_type NE 'PRD'.  
      DELETE lt\_item\_node\_data WHERE inc\_class\_code NOT IN s\_inctrm.  
  
      IF lt\_item\_node\_data IS NOT INITIAL.  
        lr\_item\_key = VALUE #( FOR ls\_item\_node\_data IN lt\_item\_node\_data  
                              ( sign           = 'I'  
                                option         = 'EQ'  
                                low            = ls\_item\_node\_data-root\_key ) ).  
  
        DELETE lt\_root\_node\_key WHERE key NOT IN lr\_item\_key.  
      ENDIF.  
    ENDIF.  
  
  ENDMETHOD.  
  
  METHOD report\_pod\_event.  
  
    TYPES:BEGIN OF ty\_outbound\_delv,  
            outbound\_delv TYPE vbeln\_vl,  
          END OF ty\_outbound\_delv.  
  
    DATA:lv\_event\_date      TYPE sy-datum,  
         lv\_event\_timestamp TYPE timestamp.  
  
    DATA:lo\_srv\_tor     TYPE REF TO /bobf/if\_tra\_service\_manager,  
         lo\_message     TYPE REF TO /bobf/if\_frw\_message,  
         lo\_exception   TYPE REF TO /bobf/cx\_frw\_contrct\_violation,  
         lo\_exp\_message TYPE REF TO /bobf/cm\_frw,  
         lo\_tra\_tor     TYPE REF TO /bobf/if\_tra\_transaction\_mgr.  
  
    DATA:lt\_failed\_key     TYPE /bobf/t\_frw\_key,  
         lt\_root\_node\_key  TYPE /bobf/t\_frw\_key,  
         lt\_root\_node\_data TYPE /scmtms/t\_tor\_root\_k,  
         lt\_item\_node\_data TYPE /scmtms/t\_tor\_item\_tr\_k,  
         lt\_outbound\_delv  TYPE STANDARD TABLE OF ty\_outbound\_delv,  
         lt\_stop\_last\_data TYPE STANDARD TABLE OF /scmtms/s\_tor\_stop\_k  
                           WITH NON-UNIQUE SORTED KEY parent\_cat\_role  
                           COMPONENTS parent\_key stop\_cat stop\_role.  
  
    CLEAR gv\_event\_code\_descp.  
  
\*---------------------------------------------------------------------------------  
\*Get event code description  
\*---------------------------------------------------------------------------------  
    IF p\_manual EQ abap\_true.  
      SELECT SINGLE description\_s  
      FROM /scmtms/c\_ev\_tyt  
      INTO @DATA(lv\_evt\_code\_descp)  
      WHERE tor\_event EQ @p\_evtcd AND  
            langu     EQ @sy-langu.  
      IF sy-subrc EQ 0.  
        gv\_event\_code\_descp = lv\_evt\_code\_descp.  
      ENDIF.  
    ELSE.  
      SELECT SINGLE description\_s  
      FROM /scmtms/c\_ev\_tyt  
      INTO @lv\_evt\_code\_descp  
      WHERE tor\_event EQ 'POD' AND  
            langu     EQ @sy-langu.  
      IF sy-subrc EQ 0.  
        gv\_event\_code\_descp = lv\_evt\_code\_descp.  
      ENDIF.  
    ENDIF.  
  
\*Instantiate service manager  
    lo\_srv\_tor = /bobf/cl\_tra\_serv\_mgr\_factory=>get\_service\_manager( /scmtms/if\_tor\_c=>sc\_bo\_key ).  
    lo\_tra\_tor = /bobf/cl\_tra\_trans\_mgr\_factory=>get\_transaction\_manager( ).  
  
    IF lo\_srv\_tor IS BOUND AND it\_fo\_key IS NOT INITIAL.  
  
\*---------------------------------------------------------------------------------  
\*Get root node data  
\*---------------------------------------------------------------------------------  
      lo\_srv\_tor->retrieve( EXPORTING iv\_node\_key   = /scmtms/if\_tor\_c=>sc\_node-root  
                                      it\_key        = it\_fo\_key  
                                      iv\_edit\_mode  = /bobf/if\_conf\_c=>sc\_edit\_read\_only  
                            IMPORTING eo\_message    = lo\_message  
                                      et\_data       = lt\_root\_node\_data  
                                      et\_failed\_key = lt\_failed\_key ).  
  
\*---------------------------------------------------------------------------------  
\*Get Item node data  
\*---------------------------------------------------------------------------------  
      lo\_srv\_tor->retrieve\_by\_association( EXPORTING iv\_node\_key    = /scmtms/if\_tor\_c=>sc\_node-root  
                                                     it\_key         = it\_fo\_key  
                                                     iv\_association = /scmtms/if\_tor\_c=>sc\_association-root-item\_tr  
                                                     iv\_fill\_data   = abap\_true  
                                           IMPORTING eo\_message     = lo\_message  
                                                     et\_data        = lt\_item\_node\_data  
                                                     et\_failed\_key  = lt\_failed\_key ).  
  
\*---------------------------------------------------------------------------------  
\*Get stop node data  
\*---------------------------------------------------------------------------------  
      lo\_srv\_tor->retrieve\_by\_association( EXPORTING iv\_node\_key    = /scmtms/if\_tor\_c=>sc\_node-root  
                                                     it\_key         = it\_fo\_key  
                                                     iv\_fill\_data   = abap\_true  
                                                     iv\_association = /scmtms/if\_tor\_c=>sc\_association-root-stop\_last  
                                           IMPORTING eo\_message     = lo\_message  
                                                     et\_data        = lt\_stop\_last\_data  
                                                     et\_failed\_key  = lt\_failed\_key ).  
  
\*---------------------------------------------------------------------------------  
\*Report POD Event and Update POD status for the Freight Order  
\*---------------------------------------------------------------------------------  
      IF lt\_stop\_last\_data IS NOT INITIAL AND lt\_root\_node\_data IS NOT INITIAL.  
        LOOP AT it\_fo\_key ASSIGNING FIELD-SYMBOL(<lfs\_fo\_key>).  
  
          CLEAR lt\_root\_node\_key.  
          lt\_root\_node\_key = VALUE #( ( key = <lfs\_fo\_key>-key ) ).  
  
\*Read last stop key  
          IF line\_exists( lt\_stop\_last\_data[ root\_key = <lfs\_fo\_key>-key ] ).  
            DATA(lv\_last\_stop\_key) = lt\_stop\_last\_data[ root\_key = <lfs\_fo\_key>-key ]-key.  
            DATA(lv\_last\_stop\_id)  = lt\_stop\_last\_data[ root\_key = <lfs\_fo\_key>-key ]-log\_locid.  
          ENDIF.  
  
\*Append data to output  
          APPEND INITIAL LINE TO gt\_output ASSIGNING FIELD-SYMBOL(<lfs\_output>).  
          IF <lfs\_output> IS ASSIGNED.  
            <lfs\_output>-event\_code\_descp = gv\_event\_code\_descp.  
            <lfs\_output>-location         = lv\_last\_stop\_id.  
  
            IF line\_exists( lt\_root\_node\_data[ key = <lfs\_fo\_key>-key ] ).  
              <lfs\_output>-freight\_order = lt\_root\_node\_data[ key = <lfs\_fo\_key>-key ]-tor\_id.  
            ENDIF.  
  
            CALL FUNCTION 'CONVERSION\_EXIT\_ALPHA\_OUTPUT'  
              EXPORTING  
                input  = <lfs\_output>-freight\_order  
              IMPORTING  
                output = <lfs\_output>-freight\_order.  
  
\*Read event date and time from excel data  
            IF p\_bulkup EQ abap\_true.  
              IF line\_exists( gt\_excel\_data[ freight\_order = <lfs\_output>-freight\_order ] ).  
  
                DATA(ls\_excel\_data) = gt\_excel\_data[ freight\_order = <lfs\_output>-freight\_order ].  
  
                CLEAR lv\_event\_date.  
                CALL FUNCTION 'CONVERSION\_EXIT\_IDATE\_INPUT'  
                  EXPORTING  
                    input  = ls\_excel\_data-event\_date  
                  IMPORTING  
                    output = lv\_event\_date.  
  
                CLEAR lv\_event\_timestamp.  
                CONVERT DATE lv\_event\_date TIME ls\_excel\_data-event\_time DAYLIGHT SAVING TIME 'X'  
                INTO TIME STAMP lv\_event\_timestamp TIME ZONE 'UTC'.  
  
                <lfs\_output>-event\_timestamp = lv\_event\_timestamp.  
              ENDIF.  
            ENDIF.  
          ENDIF.  
  
\*---------------------------------------------------------------------------------  
\*Validate POD status update in Outbound deliveries  
\*---------------------------------------------------------------------------------  
          CLEAR lt\_outbound\_delv.  
          LOOP AT lt\_item\_node\_data ASSIGNING FIELD-SYMBOL(<lfs\_d\_items>) WHERE item\_type  EQ 'PRD'  AND  
                                                                                trq\_cat    EQ '06'   AND  
                                                                                base\_btd\_id IS NOT INITIAL.  
            val\_pod\_status\_upd( EXPORTING is\_item\_node\_data = <lfs\_d\_items>  
                                IMPORTING ev\_outbound\_delv  = DATA(lv\_outbound\_delv)  
                                          et\_prot           = DATA(lt\_pott) ).  
  
            IF line\_exists( lt\_pott[ msgty = 'E' ] ).  
  
              DATA(ls\_pott) = lt\_pott[ msgty = 'E' ].  
              CALL FUNCTION 'MESSAGE\_TEXT\_BUILD'  
                EXPORTING  
                  msgid               = ls\_pott-msgid  
                  msgnr               = ls\_pott-msgno  
                  msgv1               = ls\_pott-msgv1  
                  msgv2               = ls\_pott-msgv2  
                  msgv3               = ls\_pott-msgv3  
                  msgv4               = ls\_pott-msgv4  
                IMPORTING  
                  message\_text\_output = <lfs\_output>-remarks.  
  
              DATA(lv\_error) = abap\_true.  
              EXIT.  
            ELSE.  
  
              IF lv\_outbound\_delv IS NOT INITIAL.  
                APPEND INITIAL LINE TO lt\_outbound\_delv ASSIGNING FIELD-SYMBOL(<lfs\_outbound\_delv>).  
                IF <lfs\_outbound\_delv> IS ASSIGNED.  
                  <lfs\_outbound\_delv>-outbound\_delv = lv\_outbound\_delv.  
                ENDIF.  
              ENDIF.  
  
              CLEAR lv\_outbound\_delv.  
            ENDIF.  
  
            CLEAR:ls\_pott,lt\_pott.  
          ENDLOOP.  
  
  
          IF lv\_error EQ abap\_false.  
  
\*---------------------------------------------------------------------------------  
\*Dequeue Sales Order  
\*---------------------------------------------------------------------------------  
            IF lt\_outbound\_delv IS NOT INITIAL.  
              SELECT vbeln,  
                     vgbel  
              FROM lips  
              INTO TABLE @DATA(lt\_sales\_order)  
              FOR ALL ENTRIES IN @lt\_outbound\_delv  
              WHERE vbeln EQ @lt\_outbound\_delv-outbound\_delv.  
              IF sy-subrc EQ 0.  
                SORT lt\_sales\_order BY vgbel.  
                DELETE ADJACENT DUPLICATES FROM lt\_sales\_order COMPARING vgbel.  
  
                LOOP AT lt\_sales\_order ASSIGNING FIELD-SYMBOL(<lfs\_sales\_order>).  
                  CALL FUNCTION 'DEQUEUE\_EVVBAKE'  
                    EXPORTING  
                      mode\_vbak = 'E'  
                      mandt     = sy-mandt  
                      vbeln     = <lfs\_sales\_order>-vgbel.  
                ENDLOOP.  
              ENDIF.  
            ENDIF.  
  
\*---------------------------------------------------------------------------------  
\*Create POD Event  
\*---------------------------------------------------------------------------------  
            CLEAR:lt\_failed\_key,lo\_message.  
            create\_pod\_event( EXPORTING iv\_fo\_key          = <lfs\_fo\_key>-key  
                                        iv\_last\_stop\_key   = lv\_last\_stop\_key  
                                        iv\_evt\_timestamp   = lv\_event\_timestamp  
                              IMPORTING ev\_event\_timestamp = <lfs\_output>-event\_timestamp  
                                        et\_failed\_key      = lt\_failed\_key  
                                        eo\_message         = lo\_message ).  
  
            IF lt\_failed\_key IS NOT INITIAL AND lo\_message IS BOUND.  
              lo\_message->get\_messages( IMPORTING et\_message = DATA(lt\_message) ).  
              IF lt\_message IS NOT INITIAL.  
                <lfs\_output>-remarks = lt\_message[ 1 ]-message->get\_text( ).  
              ENDIF.  
  
              lv\_error = abap\_true.  
            ELSE.  
  
\*---------------------------------------------------------------------------------  
\*Update POD status  
\*---------------------------------------------------------------------------------  
              LOOP AT lt\_item\_node\_data ASSIGNING <lfs\_d\_items> WHERE item\_type  EQ 'PRD'  AND  
                                                                      trq\_cat    EQ '06'   AND  
                                                                      base\_btd\_id IS NOT INITIAL.  
  
                CLEAR:lv\_outbound\_delv,lt\_pott.  
                update\_pod\_status( EXPORTING is\_item\_node\_data = <lfs\_d\_items>  
                                   IMPORTING ev\_outbound\_delv  = lv\_outbound\_delv  
                                             et\_prot           = lt\_pott ).  
  
                IF line\_exists( lt\_pott[ msgty = 'E' ] ).  
  
                  CLEAR ls\_pott.  
                  ls\_pott = lt\_pott[ msgty = 'E' ].  
                  CALL FUNCTION 'MESSAGE\_TEXT\_BUILD'  
                    EXPORTING  
                      msgid               = ls\_pott-msgid  
                      msgnr               = ls\_pott-msgno  
                      msgv1               = ls\_pott-msgv1  
                      msgv2               = ls\_pott-msgv2  
                      msgv3               = ls\_pott-msgv3  
                      msgv4               = ls\_pott-msgv4  
                    IMPORTING  
                      message\_text\_output = <lfs\_output>-remarks.  
  
                  revoke\_pod\_event( EXPORTING it\_fo\_key      = lt\_root\_node\_key  
                                              io\_srv\_manager = lo\_srv\_tor  
                                              io\_tra\_manager = lo\_tra\_tor  
                                    IMPORTING et\_failed\_key  = lt\_failed\_key  
                                              eo\_message     = lo\_message ).  
  
                  lv\_error = abap\_true.  
                  EXIT.  
                ELSE.  
                  IF line\_exists( lt\_sales\_order[ vbeln = lv\_outbound\_delv ] ).  
                    CALL FUNCTION 'DEQUEUE\_EVVBAKE'  
                      EXPORTING  
                        mode\_vbak = 'E'  
                        mandt     = sy-mandt  
                        vbeln     = lt\_sales\_order[ vbeln = lv\_outbound\_delv ]-vgbel.  
                  ENDIF.  
                ENDIF.  
              ENDLOOP.  
  
            ENDIF.  
          ENDIF.  
  
          IF lv\_error EQ abap\_false.  
            <lfs\_output>-remarks = TEXT-008.  
          ENDIF.  
  
          CLEAR:lv\_last\_stop\_id,lv\_error,lv\_last\_stop\_key.  
        ENDLOOP.  
  
      ENDIF.  
    ENDIF.  
  
  ENDMETHOD.  
  
  METHOD create\_pod\_event.  
  
    DATA:lt\_last\_stop\_key TYPE /bobf/t\_frw\_key\_link,  
         lt\_fo\_key        TYPE /bobf/t\_frw\_key,  
         lt\_failed\_key    TYPE /bobf/t\_frw\_key.  
  
    DATA:lo\_tra\_tor TYPE REF TO /bobf/if\_tra\_transaction\_mgr.  
  
    CLEAR:et\_failed\_key,eo\_message,ev\_event\_timestamp.  
  
\*Instantiate transaction manager  
    lo\_tra\_tor = /bobf/cl\_tra\_trans\_mgr\_factory=>get\_transaction\_manager( ).  
  
    CLEAR lt\_last\_stop\_key.  
    lt\_last\_stop\_key = VALUE #( ( source\_key = iv\_fo\_key  
                                  target\_key = iv\_last\_stop\_key ) ).  
  
    IF p\_bulkup EQ abap\_true.  
      ev\_event\_timestamp = iv\_evt\_timestamp.  
    ELSE.  
      GET TIME STAMP FIELD ev\_event\_timestamp.  
    ENDIF.  
  
    CLEAR lt\_fo\_key.  
    lt\_fo\_key = VALUE #( ( key = iv\_fo\_key ) ).  
  
    CALL METHOD /scmtms/cl\_tor\_helper\_event=>report\_event  
      EXPORTING  
        iv\_event\_code       = 'POD'  
        iv\_ref\_event\_code   = 'POD'  
        it\_tor\_key          = lt\_fo\_key  
        it\_tor\_root\_stop    = lt\_last\_stop\_key  
        iv\_event\_status     = 'R'  
        iv\_actual\_date      = ev\_event\_timestamp  
        iv\_actual\_tz        = 'UTC'  
        iv\_exec\_info\_source = 'T'  
      IMPORTING  
        et\_failed\_key       = et\_failed\_key  
        eo\_message          = eo\_message.  
  
    IF et\_failed\_key IS INITIAL AND lo\_tra\_tor IS BOUND.  
      lo\_tra\_tor->save( ).  
    ENDIF.  
  
  ENDMETHOD.  
  
  METHOD get\_freight\_orders\_xls.  
  
    DATA:lv\_event\_date      TYPE sy-datum,  
         lv\_event\_timestamp TYPE timestamp.  
  
    DATA:lt\_sel\_parameters TYPE /bobf/t\_frw\_query\_selparam,  
         lt\_failed\_key     TYPE /bobf/t\_frw\_key,  
         lt\_execution\_key  TYPE /bobf/t\_frw\_key,  
         lt\_d\_exec\_tr      TYPE /scmtms/t\_tor\_exec\_tr\_k,  
         lt\_root\_node\_data TYPE /scmtms/t\_tor\_root\_k.  
  
    DATA:lo\_srv\_tor TYPE REF TO /bobf/if\_tra\_service\_manager,  
         lo\_message TYPE REF TO /bobf/if\_frw\_message.  
  
    CLEAR:gt\_event\_update,et\_fo\_create,et\_fo\_update.  
  
\*Instantiate service manager  
    lo\_srv\_tor = /bobf/cl\_tra\_serv\_mgr\_factory=>get\_service\_manager( /scmtms/if\_tor\_c=>sc\_bo\_key ).  
  
\*Upload data from excel file  
    upload\_excel\_file( ).  
  
    IF gt\_excel\_data IS NOT INITIAL.  
  
      CLEAR lt\_sel\_parameters.  
      LOOP AT gt\_excel\_data ASSIGNING FIELD-SYMBOL(<lfs\_excel\_data>).  
        APPEND VALUE #( attribute\_name = /scmtms/if\_tor\_c=>sc\_query\_attribute-root-fo\_data\_by\_attr-tor\_id  
                        option = 'EQ'  
                        sign   = 'I'  
                        low    =  <lfs\_excel\_data>-freight\_order ) TO lt\_sel\_parameters.  
      ENDLOOP.  
  
      lo\_srv\_tor->query(  
        EXPORTING  
          iv\_query\_key            = /scmtms/if\_tor\_c=>sc\_query-root-fo\_data\_by\_attr  
          it\_selection\_parameters = lt\_sel\_parameters  
        IMPORTING  
          eo\_message              = lo\_message  
          et\_key                  = et\_fo\_create ).  
  
      lo\_srv\_tor->retrieve( EXPORTING iv\_node\_key   = /scmtms/if\_tor\_c=>sc\_node-root  
                                      it\_key        = et\_fo\_create  
                                      iv\_edit\_mode  = /bobf/if\_conf\_c=>sc\_edit\_read\_only  
                            IMPORTING eo\_message    = lo\_message  
                                      et\_data       = lt\_root\_node\_data  
                                      et\_failed\_key = lt\_failed\_key ).  
  
\*Check if POD event reported or not  
      IF et\_fo\_create IS NOT INITIAL.  
        lo\_srv\_tor->retrieve\_by\_association(  
          EXPORTING  
            iv\_node\_key    = /scmtms/if\_tor\_c=>sc\_node-root  
            it\_key         = et\_fo\_create  
            iv\_association = /scmtms/if\_tor\_c=>sc\_association-root-executioninformation\_tr  
            iv\_fill\_data   = abap\_true  
          IMPORTING  
            eo\_message     = lo\_message  
            et\_data        = lt\_d\_exec\_tr  
            et\_failed\_key  = lt\_failed\_key ).  
  
        LOOP AT et\_fo\_create ASSIGNING FIELD-SYMBOL(<lfs\_fo\_key>).  
  
          DATA(lv\_index) = sy-tabix.  
          IF line\_exists( lt\_d\_exec\_tr[ parent\_key = <lfs\_fo\_key>-key ] ).  
  
            DATA(lt\_d\_exec\_tr\_temp) = lt\_d\_exec\_tr.  
            DELETE lt\_d\_exec\_tr\_temp WHERE parent\_key NE <lfs\_fo\_key>-key.  
  
\*Get execution key of the POD event  
            CLEAR lt\_sel\_parameters.  
            lt\_sel\_parameters = VALUE #( ( attribute\_name = /scmtms/if\_tor\_c=>sc\_query\_attribute-executioninformation-query\_by\_attributes-execution\_id  
                                           option = 'EQ'  
                                           sign   = 'I'  
                                           low    = lt\_d\_exec\_tr\_temp[ 1 ]-execution\_id ) ).  
  
            APPEND VALUE #(  attribute\_name = /scmtms/if\_tor\_c=>sc\_query\_attribute-executioninformation-query\_by\_attributes-parent\_key  
                             option = 'EQ'  
                             sign   = 'I'  
                             low    = lt\_d\_exec\_tr\_temp[ 1 ]-parent\_key )  TO lt\_sel\_parameters.  
  
            CLEAR lt\_execution\_key.  
            lo\_srv\_tor->query(  
              EXPORTING  
                iv\_query\_key            = /scmtms/if\_tor\_c=>sc\_query-executioninformation-query\_by\_attributes  
                it\_selection\_parameters = lt\_sel\_parameters  
              IMPORTING  
                eo\_message              = lo\_message  
                et\_key                  = lt\_execution\_key ).  
  
\*Read event date and time from excel file  
            IF line\_exists( lt\_root\_node\_data[ key = <lfs\_fo\_key>-key ] ).  
              DATA(lv\_fo\_id) = lt\_root\_node\_data[ key = <lfs\_fo\_key>-key ]-tor\_id.  
  
              CALL FUNCTION 'CONVERSION\_EXIT\_ALPHA\_OUTPUT'  
                EXPORTING  
                  input  = lv\_fo\_id  
                IMPORTING  
                  output = lv\_fo\_id.  
  
              IF line\_exists( gt\_excel\_data[ freight\_order = lv\_fo\_id ] ).  
                DATA(ls\_excel\_data) = gt\_excel\_data[ freight\_order = lv\_fo\_id ].  
  
                CALL FUNCTION 'CONVERSION\_EXIT\_IDATE\_INPUT'  
                  EXPORTING  
                    input  = ls\_excel\_data-event\_date  
                  IMPORTING  
                    output = lv\_event\_date.  
  
                CLEAR lv\_event\_timestamp.  
                CONVERT DATE lv\_event\_date TIME ls\_excel\_data-event\_time DAYLIGHT SAVING TIME 'X'  
                INTO TIME STAMP lv\_event\_timestamp TIME ZONE 'UTC'.  
              ENDIF.  
            ENDIF.  
  
            APPEND VALUE #( fo\_key          = <lfs\_fo\_key>-key  
                            fo\_id           = lv\_fo\_id  
                            exe\_key         = lt\_execution\_key[ 1 ]-key  
                            event\_timestamp = lv\_event\_timestamp ) TO et\_fo\_update.  
  
            DELETE et\_fo\_create INDEX lv\_index.  
            CLEAR:lv\_fo\_id,lv\_index,lv\_event\_date,lv\_event\_timestamp,lt\_d\_exec\_tr\_temp.  
          ENDIF.  
  
        ENDLOOP.  
      ENDIF.  
  
    ENDIF.  
  
  ENDMETHOD.  
  
  METHOD update\_pod\_status.  
  
    DATA:lv\_outbound\_del TYPE vbeln\_vl,  
         lv\_del\_item\_no  TYPE posnr\_vl.  
  
    DATA:ls\_vbkok TYPE vbkok.  
  
    DATA:lt\_vbpok TYPE vbpok\_t,  
         lt\_verpo TYPE tvpodvb\_t,  
         lt\_prot  TYPE tab\_prott.  
  
    CLEAR:ev\_outbound\_delv,et\_prot.  
  
    DATA(lv\_base\_btd\_id) = is\_item\_node\_data-base\_btd\_id.  
    CALL FUNCTION 'CONVERSION\_EXIT\_ALPHA\_OUTPUT'  
      EXPORTING  
        input  = lv\_base\_btd\_id  
      IMPORTING  
        output = lv\_base\_btd\_id.  
  
    CLEAR lv\_outbound\_del.  
    lv\_outbound\_del = lv\_base\_btd\_id.  
  
    CALL FUNCTION 'CONVERSION\_EXIT\_ALPHA\_INPUT'  
      EXPORTING  
        input  = lv\_outbound\_del  
      IMPORTING  
        output = lv\_outbound\_del.  
  
\*Check if the delivery is not yet processed  
    SELECT SINGLE @abap\_true  
    FROM likp  
    INTO @DATA(lv\_exists)  
    WHERE vbeln EQ @lv\_outbound\_del AND  
          pdstk EQ 'A'.  
    IF sy-subrc EQ 0.  
  
      CLEAR ls\_vbkok.  
      ls\_vbkok = VALUE #( podat =  sy-datum  
                          potim =  sy-uzeit  
                          kzpod =  'B' ).  
  
      CLEAR:lv\_del\_item\_no,lt\_vbpok,lt\_verpo.  
      lv\_del\_item\_no = is\_item\_node\_data-base\_btditem\_id.  
      lt\_vbpok = VALUE #( ( vbeln\_vl = lv\_outbound\_del  
                            posnr\_vl = lv\_del\_item\_no  
                            kzpod    = 'X') ).  
  
      CALL FUNCTION 'WS\_DELIVERY\_UPDATE'  
        EXPORTING  
          vbkok\_wa                 = ls\_vbkok  
          delivery                 = lv\_outbound\_del  
          if\_error\_messages\_send\_0 = space  
          commit                   = abap\_true  
        TABLES  
          tvpod\_tab                = lt\_verpo  
          vbpok\_tab                = lt\_vbpok  
          prot                     = et\_prot.  
  
      IF line\_exists( et\_prot[ msgty = 'E' ] ).  
        RETURN.  
      ELSE.  
        ev\_outbound\_delv = lv\_outbound\_del.  
      ENDIF.  
  
    ENDIF.  
  
  ENDMETHOD.  
  
  METHOD revoke\_pod\_event.  
  
    DATA:lt\_sel\_parameters TYPE /bobf/t\_frw\_query\_selparam,  
         lt\_d\_exec\_tr      TYPE /scmtms/t\_tor\_exec\_tr\_k,  
         lt\_execution\_key  TYPE /bobf/t\_frw\_key.  
  
    CLEAR:eo\_message,et\_failed\_key.  
  
    io\_srv\_manager->retrieve\_by\_association(  
      EXPORTING  
        iv\_node\_key    = /scmtms/if\_tor\_c=>sc\_node-root  
        it\_key         = it\_fo\_key  
        iv\_association = /scmtms/if\_tor\_c=>sc\_association-root-executioninformation\_tr  
        iv\_fill\_data   = abap\_true  
      IMPORTING  
        eo\_message     = eo\_message  
        et\_data        = lt\_d\_exec\_tr  
        et\_failed\_key  = et\_failed\_key ).  
  
    IF lt\_d\_exec\_tr IS NOT INITIAL.  
      DELETE lt\_d\_exec\_tr WHERE event\_code NE 'POD'.  
  
\*Get execution key of the recently added POD event  
      CLEAR lt\_sel\_parameters.  
      lt\_sel\_parameters = VALUE #( ( attribute\_name = /scmtms/if\_tor\_c=>sc\_query\_attribute-executioninformation-query\_by\_attributes-execution\_id  
                                     option = 'EQ'  
                                     sign   = 'I'  
                                     low    = lt\_d\_exec\_tr[ 1 ]-execution\_id ) ).  
  
      APPEND VALUE #(  attribute\_name = /scmtms/if\_tor\_c=>sc\_query\_attribute-executioninformation-query\_by\_attributes-parent\_key  
                       option = 'EQ'  
                       sign   = 'I'  
                       low    = it\_fo\_key[ 1 ]-key )  TO lt\_sel\_parameters.  
  
      CLEAR lt\_execution\_key.  
      io\_srv\_manager->query(  
        EXPORTING  
          iv\_query\_key            = /scmtms/if\_tor\_c=>sc\_query-executioninformation-query\_by\_attributes  
          it\_selection\_parameters = lt\_sel\_parameters  
        IMPORTING  
          eo\_message              = eo\_message  
          et\_key                  = lt\_execution\_key ).  
  
      io\_srv\_manager->do\_action( EXPORTING it\_key        = lt\_execution\_key  
                                           iv\_act\_key    = /scmtms/if\_tor\_c=>sc\_action-executioninformation-revoke\_event  
                                 IMPORTING et\_failed\_key = et\_failed\_key  
                                           eo\_message    = eo\_message ).  
  
      IF io\_tra\_manager IS BOUND AND et\_failed\_key IS INITIAL.  
        io\_tra\_manager->save( ).  
      ENDIF.  
  
    ENDIF.  
  
  ENDMETHOD.  
  
  METHOD display\_event\_output.  
  
    DATA:lo\_column       TYPE REF TO cl\_salv\_column,  
         lo\_alv\_toolbar  TYPE REF TO cl\_salv\_functions\_list,  
         lo\_cols         TYPE REF TO cl\_salv\_columns,  
         lo\_table\_descr  TYPE REF TO cl\_abap\_tabledescr,  
         lo\_struct\_descr TYPE REF TO cl\_abap\_structdescr,  
         lo\_alv          TYPE REF TO cl\_salv\_table,  
         lo\_print        TYPE REF TO cl\_salv\_print..  
  
    DATA:lv\_long\_text   TYPE scrtext\_l,  
         lv\_medium\_text TYPE scrtext\_m,  
         lv\_short\_text  TYPE scrtext\_s.  
  
    DATA:ls\_fieldcat TYPE lvc\_t\_fcat,  
         ls\_pparams  TYPE pri\_params.  
  
    DATA:lt\_columns TYPE abap\_compdescr\_tab.  
  
    IF gt\_output IS NOT INITIAL.  
      TRY.  
          cl\_salv\_table=>factory(  
            IMPORTING  
              r\_salv\_table = lo\_alv  
            CHANGING  
              t\_table      = gt\_output ).  
  
\*Add column heading  
          lo\_cols = lo\_alv->get\_columns( ).  
          lo\_cols->set\_optimize( abap\_true ).  
  
\*Get columns  
          lo\_table\_descr ?= cl\_abap\_typedescr=>describe\_by\_data( gt\_output ).  
          lo\_struct\_descr ?= lo\_table\_descr->get\_table\_line\_type( ).  
          lt\_columns = lo\_struct\_descr->components.  
  
          LOOP AT lt\_columns ASSIGNING FIELD-SYMBOL(<lfs\_column>).  
            TRY.  
                lo\_column = lo\_cols->get\_column( <lfs\_column>-name ).  
  
                CLEAR:lv\_long\_text,  
                      lv\_medium\_text,  
                      lv\_short\_text.  
  
                IF <lfs\_column>-name EQ 'FREIGHT\_ORDER'.  
                  lv\_long\_text   = TEXT-002.  
                  lv\_medium\_text = TEXT-002.  
                  lv\_short\_text  = TEXT-002.  
                ELSEIF <lfs\_column>-name EQ 'EVENT\_CODE\_DESCP'.  
                  lv\_long\_text   = TEXT-003.  
                  lv\_medium\_text = TEXT-003.  
                  lv\_short\_text  = TEXT-003.  
                ELSEIF <lfs\_column>-name EQ 'EVENT\_TIMESTAMP'.  
                  lv\_long\_text   = TEXT-004.  
                  lv\_medium\_text = TEXT-004.  
                  lv\_short\_text  = TEXT-004.  
                ELSEIF <lfs\_column>-name EQ 'LOCATION'.  
                  lv\_long\_text   = TEXT-005.  
                  lv\_medium\_text = TEXT-005.  
                  lv\_short\_text  = TEXT-005.  
                ELSEIF <lfs\_column>-name EQ 'REMARKS'.  
                  lv\_long\_text   = TEXT-006.  
                  lv\_medium\_text = TEXT-006.  
                  lv\_short\_text  = TEXT-006.  
                ENDIF.  
  
                lo\_column->set\_long\_text( lv\_long\_text ).  
                lo\_column->set\_medium\_text( lv\_medium\_text ).  
                lo\_column->set\_short\_text( lv\_short\_text ).  
  
              CATCH cx\_salv\_not\_found.  
            ENDTRY.  
          ENDLOOP.  
  
\*Display ALV output in foreground mode or print same output in spool in background mode  
          IF sy-batch EQ abap\_false.  
            lo\_alv\_toolbar = lo\_alv->get\_functions( ).  
            lo\_alv\_toolbar->set\_all( value = if\_salv\_c\_bool\_sap=>true ).  
          ELSE.  
            lo\_print = lo\_alv->get\_print( ).  
  
            CALL FUNCTION 'GET\_PRINT\_PARAMETERS'  
              EXPORTING  
                no\_dialog              = abap\_true  
              IMPORTING  
                out\_parameters         = ls\_pparams  
              EXCEPTIONS  
                archive\_info\_not\_found = 1  
                invalid\_print\_params   = 2  
                invalid\_archive\_params = 3  
                OTHERS                 = 4.  
  
            CONCATENATE TEXT-007 '\_' sy-datum '\_' sy-uzeit INTO ls\_pparams-prtxt SEPARATED BY space.  
            CONDENSE ls\_pparams-prtxt.  
  
            IF lo\_print IS BOUND.  
              DATA(ls\_print\_ctrl) = lo\_print->get\_print\_control( ).  
              ls\_print\_ctrl-pri\_params = ls\_pparams.  
              lo\_print->set\_print\_control( ls\_print\_ctrl ).  
              lo\_print->set\_print\_only( if\_salv\_c\_bool\_sap=>true ).  
            ENDIF.  
  
          ENDIF.  
  
          lo\_alv->display( ).  
  
        CATCH cx\_salv\_msg INTO DATA(lx\_msg).  
      ENDTRY.  
    ELSE.  
      MESSAGE s000(ztm01) DISPLAY LIKE 'E'.  
      LEAVE LIST-PROCESSING.  
    ENDIF.  
  
  ENDMETHOD.  
  
  METHOD upload\_excel\_file.  
  
    DATA:lt\_raw TYPE truxs\_t\_text\_data.  
  
    CALL FUNCTION 'TEXT\_CONVERT\_XLS\_TO\_SAP'  
      EXPORTING  
        i\_line\_header        = 'X'  
        i\_tab\_raw\_data       = lt\_raw  
        i\_filename           = p\_file  
      TABLES  
        i\_tab\_converted\_data = gt\_excel\_data[]  
      EXCEPTIONS  
        conversion\_failed    = 1  
        OTHERS               = 2.  
    IF sy-subrc <> 0.  
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
    ENDIF.  
  
  ENDMETHOD.  
  
  METHOD update\_pod\_event.  
  
    DATA:lo\_srv\_tor TYPE REF TO /bobf/if\_tra\_service\_manager,  
         lo\_tra\_tor TYPE REF TO /bobf/if\_tra\_transaction\_mgr,  
         lo\_message TYPE REF TO /bobf/if\_frw\_message.  
  
    DATA:lr\_exec TYPE REF TO /scmtms/s\_tor\_exec\_k.  
  
    DATA:lt\_mod            TYPE /bobf/t\_frw\_modification,  
         lt\_failed\_key     TYPE /bobf/t\_frw\_key,  
         lt\_root\_node\_key  TYPE /bobf/t\_frw\_key,  
         lt\_stop\_last\_data TYPE STANDARD TABLE OF /scmtms/s\_tor\_stop\_k WITH NON-UNIQUE SORTED KEY parent\_cat\_role  
                           COMPONENTS parent\_key stop\_cat stop\_role.  
  
    IF it\_fo\_update IS NOT INITIAL.  
  
\*Instantiate service and transaction manager  
      lo\_srv\_tor = /bobf/cl\_tra\_serv\_mgr\_factory=>get\_service\_manager( /scmtms/if\_tor\_c=>sc\_bo\_key ).  
      lo\_tra\_tor = /bobf/cl\_tra\_trans\_mgr\_factory=>get\_transaction\_manager( ).  
  
\*Get stop node data  
      lt\_root\_node\_key = VALUE #( FOR ls\_fo\_update IN it\_fo\_update  
                                ( key = ls\_fo\_update-fo\_key ) ).  
  
      lo\_srv\_tor->retrieve\_by\_association( EXPORTING iv\_node\_key    = /scmtms/if\_tor\_c=>sc\_node-root  
                                                     it\_key         = lt\_root\_node\_key  
                                                     iv\_fill\_data   = abap\_true  
                                                     iv\_association = /scmtms/if\_tor\_c=>sc\_association-root-stop\_last  
                                           IMPORTING eo\_message     = lo\_message  
                                                     et\_data        = lt\_stop\_last\_data  
                                                     et\_failed\_key  = lt\_failed\_key ).  
  
      DELETE lt\_stop\_last\_data WHERE stop\_cat NE 'I'.  
  
      LOOP AT it\_fo\_update ASSIGNING FIELD-SYMBOL(<lfs\_fo\_update>).  
  
        APPEND INITIAL LINE TO gt\_output ASSIGNING FIELD-SYMBOL(<lfs\_output>).  
        IF <lfs\_output> IS ASSIGNED.  
          <lfs\_output>-freight\_order    = <lfs\_fo\_update>-fo\_id.  
          <lfs\_output>-event\_code\_descp = gv\_event\_code\_descp.  
          <lfs\_output>-event\_timestamp  = <lfs\_fo\_update>-event\_timestamp.  
  
          IF line\_exists( lt\_stop\_last\_data[ root\_key = <lfs\_fo\_update>-fo\_key ] ).  
            <lfs\_output>-location = lt\_stop\_last\_data[ root\_key = <lfs\_fo\_update>-fo\_key ]-log\_locid.  
          ENDIF.  
        ENDIF.  
  
        CREATE DATA lr\_exec.  
        lr\_exec->actual\_date = <lfs\_fo\_update>-event\_timestamp.  
  
        CLEAR lt\_mod.  
        INSERT VALUE #( change\_mode = /bobf/if\_frw\_c=>sc\_modify\_update  
                        node        = /scmtms/if\_tor\_c=>sc\_node-executioninformation  
                        key         = <lfs\_fo\_update>-exe\_key  
                        data        = lr\_exec  
                        changed\_fields = VALUE #( ( /scmtms/if\_tor\_c=>sc\_node\_attribute-executioninformation-actual\_date ) ) )  
        INTO TABLE lt\_mod.  
  
        CALL METHOD lo\_srv\_tor->modify  
          EXPORTING  
            it\_modification = lt\_mod  
          IMPORTING  
            eo\_message      = lo\_message.  
  
        CALL METHOD lo\_tra\_tor->save( ).  
  
        <lfs\_output>-remarks = TEXT-009.  
  
      ENDLOOP.  
  
    ENDIF.  
  
  ENDMETHOD.  
  
  METHOD val\_pod\_status\_upd.  
  
    DATA:lv\_outbound\_del TYPE vbeln\_vl,  
         lv\_del\_item\_no  TYPE posnr\_vl.  
  
    DATA:ls\_vbkok TYPE vbkok.  
  
    DATA:lt\_vbpok TYPE vbpok\_t,  
         lt\_verpo TYPE tvpodvb\_t.  
  
    CLEAR:ev\_outbound\_delv,et\_prot.  
  
    DATA(lv\_base\_btd\_id) = is\_item\_node\_data-base\_btd\_id.  
    CALL FUNCTION 'CONVERSION\_EXIT\_ALPHA\_OUTPUT'  
      EXPORTING  
        input  = lv\_base\_btd\_id  
      IMPORTING  
        output = lv\_base\_btd\_id.  
  
    CLEAR lv\_outbound\_del.  
    lv\_outbound\_del = lv\_base\_btd\_id.  
  
    CALL FUNCTION 'CONVERSION\_EXIT\_ALPHA\_INPUT'  
      EXPORTING  
        input  = lv\_outbound\_del  
      IMPORTING  
        output = lv\_outbound\_del.  
  
\*Check if the delivery is not yet processed  
    SELECT SINGLE @abap\_true  
    FROM likp  
    INTO @DATA(lv\_exists)  
    WHERE vbeln EQ @lv\_outbound\_del AND  
          pdstk EQ 'A'.  
    IF sy-subrc EQ 0.  
  
      CLEAR ls\_vbkok.  
      ls\_vbkok = VALUE #( podat =  sy-datum  
                          potim =  sy-uzeit  
                          kzpod =  'B' ).  
  
      CLEAR:lv\_del\_item\_no,lt\_vbpok,lt\_verpo.  
      lv\_del\_item\_no = is\_item\_node\_data-base\_btditem\_id.  
      lt\_vbpok = VALUE #( ( vbeln\_vl = lv\_outbound\_del  
                            posnr\_vl = lv\_del\_item\_no  
                            kzpod    = 'X') ).  
  
      CALL FUNCTION 'WS\_DELIVERY\_UPDATE'  
        EXPORTING  
          vbkok\_wa                 = ls\_vbkok  
          delivery                 = lv\_outbound\_del  
          if\_error\_messages\_send\_0 = space  
          commit                   = abap\_false  
          if\_database\_update       = '2'  
        TABLES  
          tvpod\_tab                = lt\_verpo  
          vbpok\_tab                = lt\_vbpok  
          prot                     = et\_prot.  
  
      IF line\_exists( et\_prot[ msgty = 'E' ] ).  
        RETURN.  
      ELSE.  
        ev\_outbound\_delv = lv\_outbound\_del.  
  
        CALL FUNCTION 'DEQUEUE\_EVVBLKE'  
          EXPORTING  
            mode\_likp = 'E'  
            mandt     = sy-mandt  
            vbeln     = lv\_outbound\_del.  
      ENDIF.  
    ENDIF.  
  
  ENDMETHOD.  
  
ENDCLASS.

Unit test Results:

n below:

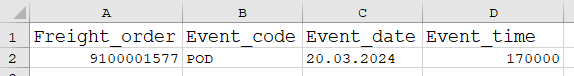


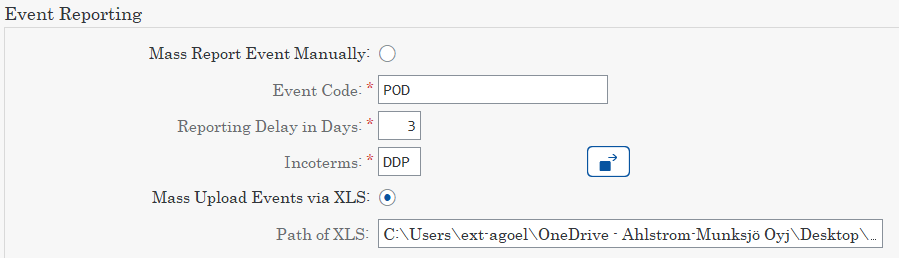
Foreground Mode:

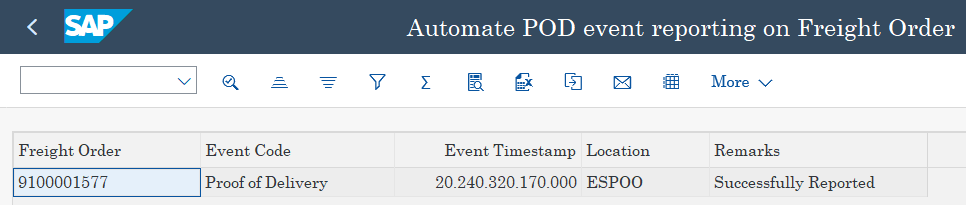


Note: Outbound deliveries - 50710,50711,50712 are skipped at runtime for now as PGI is not done for them.

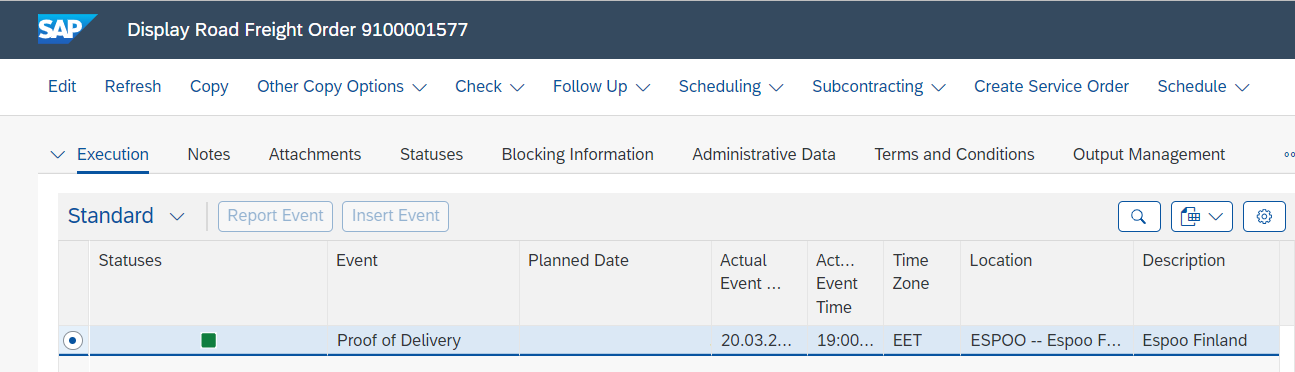
Upload attached file with below data:



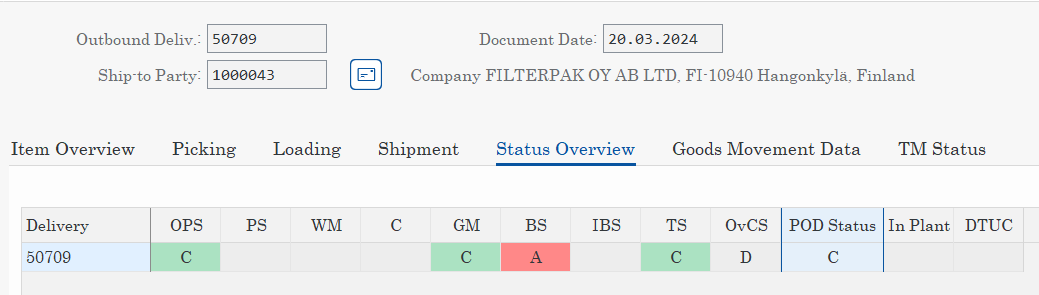


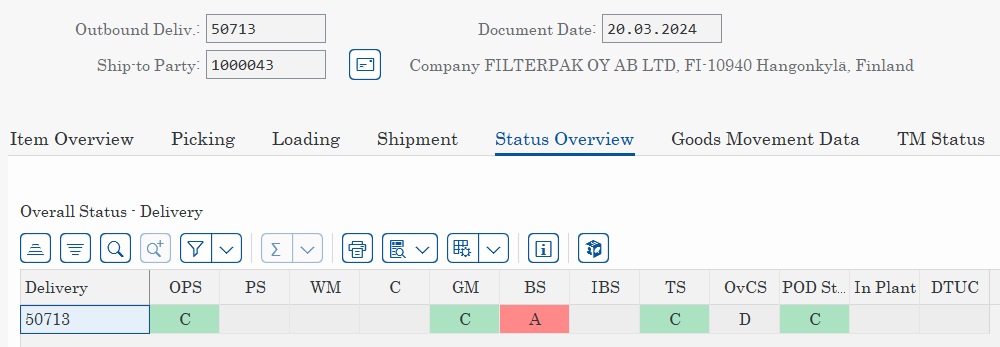


POD Event successfully reported



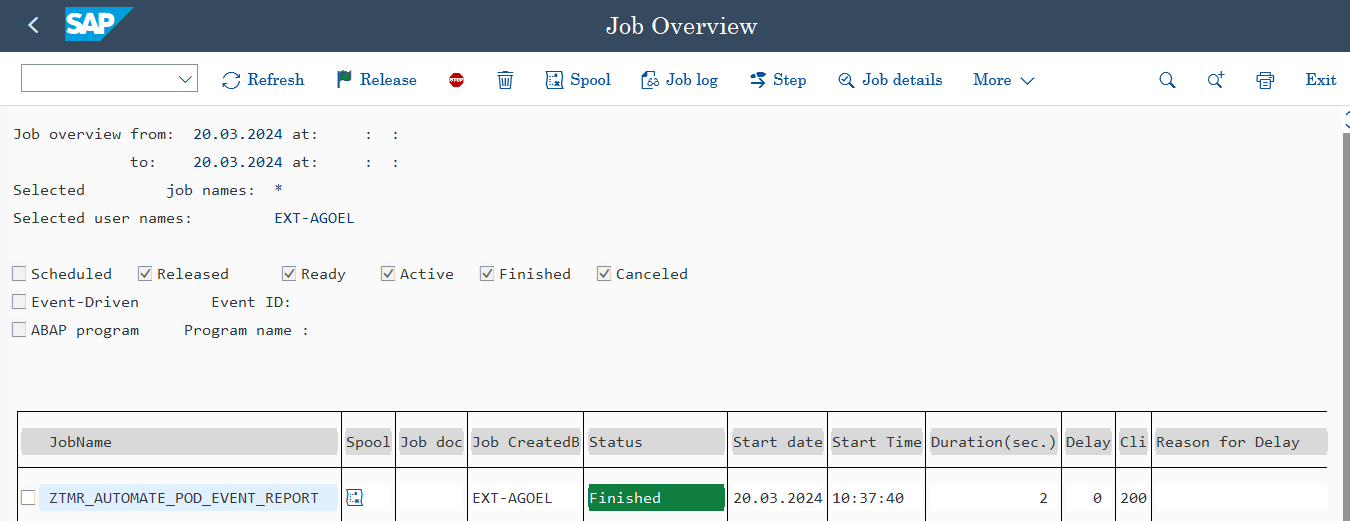
POD status successfully updated in the outbound deliveries:

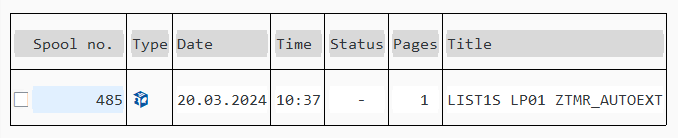


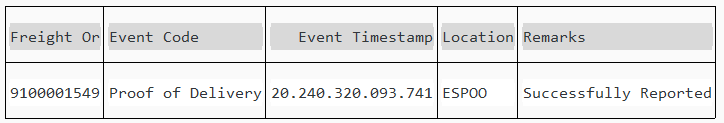


Note:

1. In case of executing the report in background mode, output will be available in spool as shown below:







1. If the POD event is already existing against the Freight order uploaded in the excel file, then its actual date timestamp will be updated from the excel file.