Receiving data directly from CPI:

SICF:  
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

class /BAY0/O2C\_CL\_DEFENSIVES\_HNDL definition

public

inheriting from CL\_REST\_HTTP\_HANDLER

create public .

public section.

methods IF\_REST\_APPLICATION~GET\_ROOT\_HANDLER

redefinition .

protected section.

private section.

ENDCLASS.

CLASS /BAY0/O2C\_CL\_DEFENSIVES\_HNDL IMPLEMENTATION.

METHOD IF\_REST\_APPLICATION~GET\_ROOT\_HANDLER.

DATA(lo\_rest\_handler) = NEW cl\_rest\_router( ).

\* define a route that is mentioned after the url to the service

lo\_rest\_handler->attach( iv\_template = '/update\_nfe' iv\_handler\_class = '/BAY0/O2C\_CL\_DEFENSIVES\_SRV' ).

ro\_root\_handler = lo\_rest\_handler.

ENDMETHOD.

ENDCLASS.

class /BAY0/O2C\_CL\_DEFENSIVES\_SRV definition

public

inheriting from CL\_REST\_RESOURCE

create public .

public section.

types:

BEGIN OF ty\_msg,

message TYPE string,

END OF ty\_msg .

types:

tt\_msg TYPE TABLE OF string WITH EMPTY KEY .

types:

\*\*/ Types

BEGIN OF ty\_request\_body,

docnum TYPE string,

uf TYPE string,

GO\_Request TYPE string,

GO\_Response TYPE string,

END OF ty\_request\_body .

types:

tt\_message TYPE STANDARD TABLE OF string WITH EMPTY KEY .

types:

BEGIN OF ty\_request\_go,

result TYPE xfeld,

codigo\_receita TYPE string,

nu\_receita TYPE string,

messages TYPE tt\_message,

total TYPE i,

END OF ty\_request\_go .

methods BASE64\_DECODE

importing

!IV\_ENCODED type STRING

returning

value(RV\_DECODED) type STRING .

methods IF\_REST\_RESOURCE~GET

redefinition .

methods IF\_REST\_RESOURCE~POST

redefinition .

PROTECTED SECTION.

PRIVATE SECTION.

ENDCLASS.

CLASS /BAY0/O2C\_CL\_DEFENSIVES\_SRV IMPLEMENTATION.

METHOD if\_rest\_resource~get.

\* mo\_response->create\_entity( )->set\_string\_data( `Fetch Token` ).

ENDMETHOD.

METHOD if\_rest\_resource~post.

\*\*/ Constants

CONSTANTS: lc\_e TYPE c VALUE 'E',

lc\_retorno\_uf TYPE char10 VALUE 'RET\_UF',

lc\_finish TYPE c VALUE '5',

lc\_rejected TYPE c VALUE '4',

lc\_json\_true TYPE char4 VALUE 'true',

lc\_json\_false TYPE char5 VALUE 'false'.

\*\*/ Work Area

DATA: ls\_request\_body TYPE ty\_request\_body,

ls\_payload\_go TYPE ty\_request\_go.

\*\*/ Variables

DATA(lv\_request\_body) = mo\_request->get\_entity( )->get\_string\_data( ).

DATA(lv\_content\_type) = mo\_request->get\_header\_field( 'content-type' ).

/ui2/cl\_json=>deserialize(

EXPORTING

json = lv\_request\_body

CHANGING

data = ls\_request\_body ).

DATA(lv\_response\_go) = me->base64\_decode( ls\_request\_body-go\_response ).

IF lv\_response\_go IS NOT INITIAL.

lv\_response\_go = replace( val = lv\_response\_go sub = lc\_json\_true with = |{ ` "` }{ abap\_true }{ `"` }| ).

lv\_response\_go = replace( val = lv\_response\_go sub = lc\_json\_false with = |{ ` "` }{ abap\_false }{ `"` }| ).

/ui2/cl\_json=>deserialize(

EXPORTING

json = lv\_response\_go

CHANGING

data = ls\_payload\_go ).

DATA(lo\_def) = NEW /bay0/o2c\_cl\_def( ).

CASE ls\_payload\_go-result.

WHEN abap\_true.

lo\_def->cpi\_set\_finished( iv\_docnum = CONV #( ls\_request\_body-docnum )

iv\_uf = CONV #( ls\_request\_body-uf )

iv\_receita = CONV #( ls\_payload\_go-nu\_receita )

iv\_mime\_type = CONV #( lv\_content\_type )

iv\_file\_raw = /ui2/cl\_json=>string\_to\_raw( iv\_string = me->base64\_decode( ls\_request\_body-go\_request ) ) ).

WHEN abap\_false.

lo\_def->cpi\_set\_rejected( iv\_docnum = CONV #( ls\_request\_body-docnum )

iv\_uf = CONV #( ls\_request\_body-uf )

iv\_receita = CONV #( ls\_payload\_go-nu\_receita )

iv\_mime\_type = CONV #( lv\_content\_type )

iv\_file\_raw = /ui2/cl\_json=>string\_to\_raw( iv\_string = me->base64\_decode( ls\_request\_body-go\_request ) )

it\_messages = CONV #( ls\_payload\_go-messages ) ).

WHEN OTHERS.

ENDCASE.

\* lo\_def->save\_log( iv\_docnum = CONV #( ls\_request\_body-docnum )

\* iv\_uf = CONV #( ls\_request\_body-uf )

\* iv\_item = ls\_def\_itm-itmnum

\* iv\_save\_file = abap\_false

\* iv\_status = COND #( WHEN ls\_payload\_go-result EQ abap\_true THEN lc\_finish ELSE lc\_rejected )

\* it\_balmsg = COND #( WHEN ls\_payload\_go-messages IS NOT INITIAL

\* THEN VALUE #( FOR ls\_message IN ls\_payload\_go-messages

\* ( msgty = lc\_e

\* msgid = lc\_retorno\_uf

\* msgno = 1

\* msgv1 = substring( val = ls\_message len = strlen( ls\_message ) )

\* msgv2 = COND #( WHEN strlen( ls\_message ) GT 50 THEN substring( val = ls\_message off = 50 len = strlen( ls\_message ) - 50 ) ELSE '' )

\* msgv3 = COND #( WHEN strlen( ls\_message ) GT 100 THEN substring( val = ls\_message off = 100 len = strlen( ls\_message ) - 100 ) ELSE '' )

\* msgv4 = COND #( WHEN strlen( ls\_message ) GT 150 THEN substring( val = ls\_message off = 150 len = strlen( ls\_message ) - 150 ) ELSE '' ) ) )

\* ELSE VALUE #( ( msgty = lc\_e

\* msgid = lc\_retorno\_uf

\* msgno = 1

\* msgv1 = 'Receita Emitida' ) ) ) ).

\* ENDLOOP.

\*

ENDIF.

ENDMETHOD.

METHOD base64\_decode.

\*\*/ Variables

DATA lv\_xstring TYPE xstring.

CALL FUNCTION 'SCMS\_BASE64\_DECODE\_STR'

EXPORTING

input = iv\_encoded

IMPORTING

output = lv\_xstring.

cl\_bcs\_convert=>xstring\_to\_string(

EXPORTING

iv\_xstr = lv\_xstring

iv\_cp = 4110 " SAP character set identification

RECEIVING

rv\_string = rv\_decoded ).

ENDMETHOD.

ENDCLASS.

class /BAY0/O2C\_CL\_DEF definition

public

final

create public .

public section.

constants GC\_LOG\_OBJECT type BALOBJ\_D value '/BAY0/O2C\_DEF\_LOG' ##NO\_TEXT.

constants GC\_LOG\_MSG\_CLASS type ARBGB value '/BAY0/O2C\_DEF\_MSG' ##NO\_TEXT.

methods CONSTRUCTOR .

methods SAVE\_LOG

importing

!IV\_DOCNUM type J\_1BDOCNUM

!IV\_UF type /BAY0/O2C\_DEF\_UF\_REGUL

!IV\_ITEM type J\_1BITMNUM

!IV\_STATUS type /BAY0/O2C\_DEF\_STATUS

!IT\_BALMSG type BAL\_T\_MSG

!IV\_SAVE\_FILE type FLAG optional

!IV\_MIME\_TYPE type W3CONTTYPE optional

!IV\_FILE\_RAW type /BAY0/FILEX\_RAW optional

!IT\_MESSAGES type /AIF/STRING\_TT optional

!IS\_DEFHDR type /BAY0/O2C\_DEFHDR optional

!IT\_DEFITM type /BAY0/O2C\_T\_DEFITM optional .

methods GET\_LOG

importing

!IV\_DOCNUM type J\_1BDOCNUM

returning

value(RT\_LOG) type /BAY0/O2C\_DEF\_LOG\_TT .

methods CHECK\_VALID\_NF

importing

!IV\_DOCNUM type J\_1BDOCNUM

!IV\_CODE type J\_1BSTATUSCODE optional

exporting

value(ET\_BRAN\_DEST) type /BAY0/O2C\_T\_DEF\_BRAN\_DEST

returning

value(EV\_VALID) type FLAG .

methods CHECK\_COMPLETED\_NF

importing

!IV\_DOCNUM type J\_1BDOCNUM

!IV\_UF type /BAY0/O2C\_DEF\_UF\_REGUL

!IS\_MEM\_DEFHDR type /BAY0/O2C\_DEFHDR optional

!IT\_MEM\_DEFITM type /BAY0/O2C\_T\_DEFITM optional

exceptions

NFE\_NOT\_COMPLETED

DYNAMIC\_SQL\_FAILED .

methods DEF\_NF\_FEED

importing

value(IV\_DOCNUM) type J\_1BDOCNUM

value(IV\_UF\_TYPE) type /BAY0/O2C\_DEF\_UF\_TYPE

value(IV\_UF) type /BAY0/O2C\_DEF\_UF\_REGUL

value(IV\_RA\_ONLY) type FLAG

value(IV\_UPDATE) type FLAG default ''

value(IV\_COMMIT) type FLAG default ''

value(IV\_LOG\_UPD\_MD) type FLAG optional

exporting

value(ES\_DEFHDR) type /BAY0/O2C\_DEFHDR

value(ET\_DEFITM) type /BAY0/O2C\_T\_DEFITM

exceptions

DOC\_STATUS\_FINISHED

DOC\_NOT\_FOUND .

methods SAVE\_FILE

importing

!IV\_DOCNUM type J\_1BDOCNUM optional

!IS\_DEFHDR type /BAY0/O2C\_DEFHDR optional

!IT\_DEFITM type /BAY0/O2C\_T\_DEFITM optional .

methods SEND\_NF

exporting

!EV\_STATUS type STATUS2\_BR

changing

!CS\_DEFHDR type /BAY0/O2C\_DEFHDR .

methods DEF\_NF\_COMPLETE\_PROCESS

importing

!IV\_DOCNUM type J\_1BDOCNUM

exporting

!ET\_DEFHDR type /BAY0/O2C\_T\_DEFHDR

!ET\_DEFITM type /BAY0/O2C\_T\_DEFITM .

methods CPI\_SET\_FINISHED

importing

!IV\_DOCNUM type J\_1BDOCNUM

!IV\_UF type /BAY0/O2C\_DEF\_UF\_REGUL

!IV\_RECEITA type /BAY0/O2C\_DEF\_NU\_RECEITA optional

!IV\_MIME\_TYPE type W3CONTTYPE optional

!IV\_FILE\_RAW type /BAY0/FILEX\_RAW optional

!IT\_MESSAGES type /AIF/STRING\_TT optional .

methods CPI\_SET\_REJECTED

importing

!IV\_DOCNUM type J\_1BDOCNUM

!IV\_UF type /BAY0/O2C\_DEF\_UF\_REGUL

!IV\_RECEITA type /BAY0/O2C\_DEF\_NU\_RECEITA optional

!IV\_MIME\_TYPE type W3CONTTYPE optional

!IV\_FILE\_RAW type /BAY0/FILEX\_RAW optional

!IT\_MESSAGES type /AIF/STRING\_TT optional .

protected section.

private section.

ENDCLASS.

CLASS /BAY0/O2C\_CL\_DEF IMPLEMENTATION.

method CONSTRUCTOR.

endmethod.

METHOD get\_log.

DATA: lt\_log\_header TYPE balhdr\_t,

lt\_msg\_handle TYPE bal\_t\_msgh,

ls\_msg TYPE bal\_s\_msg,

ls\_log TYPE /bay0/o2c\_def\_log\_s.

\*--------------------------------------------------------------------\*

" Create log filter

\*--------------------------------------------------------------------\*

DATA(lt\_bal\_sel\_extn) = VALUE bal\_r\_extn( ( sign = 'I' option = 'CP' low = '\*' && iv\_docnum && '\*' ) ).

DATA(lt\_bal\_sel\_obj) = VALUE bal\_r\_obj( ( sign = 'I' option = 'EQ' low = gc\_log\_object ) ).

\* DATA(lt\_bal\_sel\_sobj) = VALUE bal\_r\_sub( ( sign = 'I' option = 'EQ' low = iv\_subobject ) ).

DATA(ls\_log\_filter) = VALUE bal\_s\_lfil( extnumber = lt\_bal\_sel\_extn

object = lt\_bal\_sel\_obj ).

\* subobject = lt\_bal\_sel\_sobj ).

\*--------------------------------------------------------------------\*

" Pass the header data and get log header data to find log handle

\*--------------------------------------------------------------------\*

CALL FUNCTION 'BAL\_DB\_SEARCH'

EXPORTING

i\_s\_log\_filter = ls\_log\_filter

IMPORTING

e\_t\_log\_header = lt\_log\_header

EXCEPTIONS

log\_not\_found = 1

no\_filter\_criteria = 2

OTHERS = 3. "#EC CI\_SUBRC

IF sy-subrc = 0.

\*--------------------------------------------------------------------\*

"Get message handler to find actual message

\*--------------------------------------------------------------------\*

CALL FUNCTION 'BAL\_DB\_LOAD'

EXPORTING

i\_t\_log\_header = lt\_log\_header

i\_client = sy-mandt

IMPORTING

e\_t\_msg\_handle = lt\_msg\_handle

EXCEPTIONS

no\_logs\_specified = 1

log\_not\_found = 2

log\_already\_loaded = 3

OTHERS = 4. "#EC CI\_SUBRC

IF sy-subrc = 0.

\*--------------------------------------------------------------------\*

"Get log message and export

\*--------------------------------------------------------------------\*

LOOP AT lt\_msg\_handle ASSIGNING FIELD-SYMBOL(<ls\_msg\_handler>).

CALL FUNCTION 'BAL\_LOG\_MSG\_READ'

EXPORTING

i\_s\_msg\_handle = <ls\_msg\_handler>

i\_langu = sy-langu

IMPORTING

e\_s\_msg = ls\_msg

EXCEPTIONS

log\_not\_found = 1

msg\_not\_found = 2

OTHERS = 3. "#EC CI\_SUBRC

IF sy-subrc = 0.

MOVE-CORRESPONDING ls\_msg TO ls\_log.

MESSAGE ID ls\_msg-msgid TYPE ls\_msg-msgty NUMBER ls\_msg-msgno

WITH ls\_msg-msgv1 ls\_msg-msgv2 ls\_msg-msgv3 ls\_msg-msgv4 INTO ls\_log-message.

ls\_log-docnum = iv\_docnum.

APPEND ls\_log TO rt\_log.

CLEAR: ls\_log, ls\_msg.

ENDIF.

ENDLOOP.

ENDIF.

ENDIF.

ENDMETHOD.

METHOD save\_log.

DATA: ls\_str\_log TYPE bal\_s\_log,

ls\_log\_db TYPE /bay0/o2c\_deflog,

ls\_log\_file TYPE /bay0/o2c\_deflog,

lt\_log\_db TYPE TABLE OF /bay0/o2c\_deflog,

lv\_log\_handle TYPE balloghndl,

lv\_msg\_logged TYPE boolean.

DATA: lt\_json TYPE /bay0/o2c\_def\_json\_tt,

ls\_json TYPE LINE OF /bay0/o2c\_def\_json\_tt,

ls\_json\_item TYPE LINE OF /bay0/o2c\_defitm\_tt.

CHECK ( iv\_docnum IS NOT INITIAL AND it\_balmsg IS NOT INITIAL )

OR ( iv\_save\_file IS NOT INITIAL ).

\* ls\_str\_log-extnumber = iv\_docnum.

\* CONDENSE ls\_str\_log-extnumber.

\* ls\_str\_log-object = gc\_log\_object.

\*

\* "Log create

\* CALL FUNCTION 'BAL\_LOG\_CREATE'

\* EXPORTING

\* i\_s\_log = ls\_str\_log

\* IMPORTING

\* e\_log\_handle = lv\_log\_handle

\* EXCEPTIONS

\* log\_header\_inconsistent = 1

\* OTHERS = 2.

\*

\* IF sy-subrc EQ 0.

\*

\* "add the message

\* LOOP AT it\_balmsg INTO DATA(ls\_balmsg). ##NEEDED.

\*

\* CALL FUNCTION 'BAL\_LOG\_MSG\_ADD'

\* EXPORTING

\* i\_log\_handle = lv\_log\_handle

\* i\_s\_msg = ls\_balmsg

\* IMPORTING

\* e\_msg\_was\_logged = lv\_msg\_logged

\* EXCEPTIONS

\* log\_not\_found = 1

\* msg\_inconsistent = 2

\* log\_is\_full = 3

\* OTHERS = 4.

\*

\* ENDLOOP.

\* "save the log to DB

\* IF sy-subrc EQ 0.

\* CALL FUNCTION 'BAL\_DB\_SAVE'

\* EXPORTING

\* i\_save\_all = abap\_true

\* EXCEPTIONS

\* log\_not\_found = 1

\* save\_not\_allowed = 2

\* numbering\_error = 3

\* OTHERS = 4.

\* ENDIF.

\* ENDIF.

SELECT MAX( sequen )

INTO @DATA(lv\_max\_seq)

FROM /bay0/o2c\_deflog

WHERE uf = @iv\_uf

AND docnum = @iv\_docnum

AND itmnum = @iv\_item.

IF sy-subrc = 0.

lv\_max\_seq = lv\_max\_seq + 1.

ELSE.

lv\_max\_seq = 1.

ENDIF.

IF iv\_save\_file IS NOT INITIAL.

IF iv\_file\_raw IS NOT SUPPLIED. "File coming from CPI return

IF is\_defhdr IS NOT INITIAL AND it\_defitm IS NOT INITIAL.

MOVE-CORRESPONDING is\_defhdr TO ls\_json-nfheader.

LOOP AT it\_defitm ASSIGNING FIELD-SYMBOL(<fs\_defitm>) WHERE docnum = is\_defhdr-docnum

AND uf = is\_defhdr-uf.

MOVE-CORRESPONDING <fs\_defitm> TO ls\_json\_item.

APPEND ls\_json\_item TO ls\_json-nfitems.

CLEAR ls\_json\_item.

ENDLOOP.

APPEND ls\_json TO lt\_json.

CLEAR ls\_json.

ELSEIF iv\_docnum IS NOT INITIAL.

SELECT \*

FROM /bay0/o2c\_defhdr

INTO TABLE @DATA(lt\_def\_hdr)

WHERE docnum = @iv\_docnum.

IF sy-subrc = 0.

SORT lt\_def\_hdr BY docnum.

SELECT \*

FROM /bay0/o2c\_defitm

INTO TABLE @DATA(lt\_def\_itm)

FOR ALL ENTRIES IN @lt\_def\_hdr

WHERE docnum = @lt\_def\_hdr-docnum

AND uf = @lt\_def\_hdr-uf.

IF sy-subrc = 0.

SORT lt\_def\_itm BY docnum uf.

ENDIF.

LOOP AT lt\_def\_hdr ASSIGNING FIELD-SYMBOL(<fs\_def\_hdr>).

MOVE-CORRESPONDING <fs\_def\_hdr> TO ls\_json-nfheader.

LOOP AT lt\_def\_itm ASSIGNING FIELD-SYMBOL(<fs\_def\_itm>) WHERE docnum = <fs\_def\_hdr>-docnum

AND uf = <fs\_def\_hdr>-uf.

MOVE-CORRESPONDING <fs\_def\_itm> TO ls\_json\_item.

APPEND ls\_json\_item TO ls\_json-nfitems.

CLEAR ls\_json\_item.

ENDLOOP.

APPEND ls\_json TO lt\_json.

CLEAR ls\_json.

ENDLOOP.

ENDIF.

ENDIF.

IF lt\_json IS NOT INITIAL.

GET TIME STAMP FIELD DATA(lv\_timestamp).

ls\_log\_file-filename = lv\_timestamp && '\_' && iv\_uf && '\_' && iv\_docnum.

ls\_log\_file-mimetype = 'application/json'.

DATA(lv\_json\_case) = /ui2/cl\_json=>serialize(

data = lt\_json

compress = abap\_false

pretty\_name = /ui2/cl\_json=>pretty\_mode-low\_case ).

DATA(lv\_json\_raw) = /ui2/cl\_json=>string\_to\_raw( iv\_string = lv\_json\_case ).

ls\_log\_file-filex = lv\_json\_case.

ls\_log\_file-file\_raw = lv\_json\_raw.

ENDIF.

ELSE.

GET TIME STAMP FIELD lv\_timestamp.

ls\_log\_file-filename = lv\_timestamp && '\_' && iv\_uf && '\_' && iv\_docnum.

ls\_log\_file-mimetype = COND #( WHEN iv\_mime\_type IS NOT SUPPLIED THEN 'application/json' ELSE iv\_mime\_type ).

ls\_log\_file-filex = /ui2/cl\_json=>raw\_to\_string( iv\_xstring = iv\_file\_raw ).

ls\_log\_file-file\_raw = iv\_file\_raw.

ENDIF.

ENDIF.

LOOP AT it\_balmsg ASSIGNING FIELD-SYMBOL(<fs\_balmsg>).

MESSAGE ID <fs\_balmsg>-msgid TYPE <fs\_balmsg>-msgty NUMBER <fs\_balmsg>-msgno

WITH <fs\_balmsg>-msgv1 <fs\_balmsg>-msgv2 <fs\_balmsg>-msgv3 <fs\_balmsg>-msgv4 INTO DATA(lv\_text).

ls\_log\_db-docnum = iv\_docnum.

ls\_log\_db-itmnum = iv\_item.

ls\_log\_db-uf = iv\_uf.

ls\_log\_db-sequen = lv\_max\_seq.

ls\_log\_db-status = iv\_status.

ls\_log\_db-credat = sy-datum.

ls\_log\_db-cretim = sy-uzeit.

ls\_log\_db-crenam = sy-uname.

ls\_log\_db-descr = lv\_text.

IF lines( it\_balmsg ) = sy-tabix. "Last line

IF iv\_save\_file IS NOT INITIAL AND ( lt\_json IS NOT INITIAL OR iv\_file\_raw IS SUPPLIED ).

ls\_log\_db-filename = ls\_log\_file-filename.

ls\_log\_db-mimetype = ls\_log\_file-mimetype.

ls\_log\_db-filex = ls\_log\_file-filex.

ls\_log\_db-file\_raw = ls\_log\_file-file\_raw.

ENDIF.

ENDIF.

APPEND ls\_log\_db TO lt\_log\_db.

CLEAR ls\_log\_db.

lv\_max\_seq = lv\_max\_seq + 1.

ENDLOOP.

IF it\_messages IS SUPPLIED AND it\_messages IS NOT INITIAL.

LOOP AT it\_messages ASSIGNING FIELD-SYMBOL(<fs\_msg>).

ls\_log\_db-docnum = iv\_docnum.

ls\_log\_db-itmnum = iv\_item.

ls\_log\_db-uf = iv\_uf.

ls\_log\_db-sequen = lv\_max\_seq.

ls\_log\_db-status = iv\_status.

ls\_log\_db-credat = sy-datum.

ls\_log\_db-cretim = sy-uzeit.

ls\_log\_db-crenam = sy-uname.

ls\_log\_db-descr = <fs\_msg>.

IF lines( it\_messages ) = sy-tabix. "Last line

IF iv\_save\_file IS NOT INITIAL AND ( lt\_json IS NOT INITIAL OR iv\_file\_raw IS SUPPLIED ).

ls\_log\_db-filename = ls\_log\_file-filename.

ls\_log\_db-mimetype = ls\_log\_file-mimetype.

ls\_log\_db-filex = ls\_log\_file-filex.

ls\_log\_db-file\_raw = ls\_log\_file-file\_raw.

ENDIF.

ENDIF.

ENDLOOP.

APPEND ls\_log\_db TO lt\_log\_db.

CLEAR ls\_log\_db.

lv\_max\_seq = lv\_max\_seq + 1.

ENDIF.

IF lt\_log\_db IS NOT INITIAL.

MODIFY /bay0/o2c\_deflog FROM TABLE lt\_log\_db.

ENDIF.

ENDMETHOD.

METHOD check\_valid\_nf.

\*&---------------------------------------------------------------------\*

\*& Method CHECK\_VALID\_NF

\*----------------------------------------------------------------------\*

\* Identification

\* Author : Denis Pereira - EURQK

\* Creation date : 26.11.2024

\* Owner : Daniel Golin

\* Basis Release : 755

\*-----------------------------------------------------------------------

\* Description : Check valid defensive NF documents.

\*----------------------------------------------------------------------\*

\* Changes \*

\* Vers. Date Author Request Description \*

\* V001 EURQK S1DK932604 created \*

\*----------------------------------------------------------------------\*

\* Local Internal Types.

TYPES: BEGIN OF ts\_uf\_dest,

regio TYPE kna1-regio,

bukrs TYPE t001-bukrs,

ra\_only TYPE flag,

END OF ts\_uf\_dest,

BEGIN OF ts\_uf\_branch,

regio TYPE kna1-regio,

bukrs TYPE t001-bukrs,

branch TYPE j\_1bnfdoc-branch,

ra\_only TYPE flag,

END OF ts\_uf\_branch.

\* Local Internal tables.

DATA: lt\_uf\_dest TYPE TABLE OF ts\_uf\_dest,

lt\_uf\_branch TYPE TABLE OF ts\_uf\_branch.

\* Local Ranges.

DATA: lt\_r\_param TYPE RANGE OF /bay0/tda\_devkey-param,

lt\_r\_nf\_status TYPE RANGE OF j\_1bstatuscode.

\* Local Structures

DATA: ls\_r\_param LIKE LINE OF lt\_r\_param,

ls\_r\_nf\_status LIKE LINE OF lt\_r\_nf\_status,

ls\_uf\_dest TYPE ts\_uf\_dest,

ls\_uf\_branch TYPE ts\_uf\_branch,

ls\_bran\_dest TYPE /bay0/o2c\_def\_bran\_dest.

\* Local variables

DATA: lv\_invalid TYPE flag,

lv\_dummy(40) TYPE c,

lv\_ra(2) TYPE c.

\* Fill Ranges parameters.

ls\_r\_param-sign = 'I'.

ls\_r\_param-option = 'CP'.

ls\_r\_param-low = 'NF\_STATUS\_\*'.

\* Append range

APPEND ls\_r\_param TO lt\_r\_param.

ls\_r\_param-sign = 'I'.

ls\_r\_param-option = 'CP'.

ls\_r\_param-low = 'REGIO\_BUKRS\_DEST\*'.

\* Append range

APPEND ls\_r\_param TO lt\_r\_param.

ls\_r\_param-sign = 'I'.

ls\_r\_param-option = 'CP'.

ls\_r\_param-low = 'REGIO\_BUKRS\_BRANCH\*'.

\* Append range

APPEND ls\_r\_param TO lt\_r\_param.

\* Select Development key for global paramter

SELECT param, value

FROM /bay0/tda\_devkey

WHERE devkey = '/BAY0/O2C\_ICBR\_DEF'

AND param IN @lt\_r\_param

AND activ = @abap\_true

INTO TABLE @DATA(lt\_param\_value).

\* Process constant lines.

LOOP AT lt\_param\_value INTO DATA(ls\_param\_value).

CLEAR: ls\_uf\_dest,

ls\_uf\_branch,

lv\_ra.

CASE ls\_param\_value-param.

WHEN OTHERS.

\* Fill NF valid status

IF ls\_param\_value-param(10) = 'NF\_STATUS\_'.

ls\_r\_nf\_status-sign = 'I'.

ls\_r\_nf\_status-option = 'EQ'.

ls\_r\_nf\_status-low = ls\_param\_value-value.

APPEND ls\_r\_nf\_status TO lt\_r\_nf\_status.

ENDIF.

\* Fill UF Destination.

IF ls\_param\_value-param(16) = 'REGIO\_BUKRS\_DEST'.

SPLIT ls\_param\_value-value AT '\_' INTO ls\_uf\_dest-regio

ls\_uf\_dest-bukrs

lv\_dummy

lv\_ra.

\* If last part of parameter is RA

IF lv\_ra = 'RA'.

\* Send registers only if prescription (Receituario) is obligatory.

ls\_uf\_dest-ra\_only = abap\_true.

ENDIF.

APPEND ls\_uf\_dest TO lt\_uf\_dest.

ENDIF.

\* Fill UF Branch.

IF ls\_param\_value-param(18) = 'REGIO\_BUKRS\_BRANCH'.

SPLIT ls\_param\_value-value AT '\_' INTO ls\_uf\_branch-regio

ls\_uf\_branch-bukrs

ls\_uf\_branch-branch

lv\_ra.

\* If last part of parameter is RA

IF lv\_ra = 'RA'.

\* Send registers only if prescription (Receituario) is obligatory.

ls\_uf\_dest-ra\_only = abap\_true.

ENDIF.

APPEND ls\_uf\_branch TO lt\_uf\_branch.

ENDIF.

ENDCASE.

ENDLOOP."AT lt\_param\_value INTO DATA(ls\_param\_value).

\* Sort UF destination by key

SORT lt\_uf\_dest BY regio bukrs.

\* Sort UF branch by key

SORT lt\_uf\_branch BY regio bukrs branch.

\* Check if status range from constant is not empty.

IF lt\_r\_nf\_status[] IS INITIAL.

lv\_invalid = abap\_true.

ENDIF.

\* Check if no invalid condition was reached.

CHECK lv\_invalid IS INITIAL.

\* If status code prameter is not empty check if its valid.

IF NOT iv\_code IS INITIAL. " parameter is not empty

IF NOT iv\_code IN lt\_r\_nf\_status. " NF status code is not in the range.

lv\_invalid = abap\_true.

ENDIF.

ELSE.

\* Select Electronic Nota Fiscal: Actual Status

SELECT SINGLE code

FROM j\_1bnfe\_active

WHERE docnum = @iv\_docnum

INTO @DATA(lv\_code).

IF NOT lv\_code IN lt\_r\_nf\_status. " NF status code is not in the range.

lv\_invalid = abap\_true.

ENDIF.

ENDIF."NOT iv\_code IS INITIAL. " parameter is not empty

\* Check if no invalid condition was reached.

CHECK lv\_invalid IS INITIAL.

\* Select NF header

SELECT SINGLE a~bukrs, a~branch, a~regio AS dest, b~adrnr, c~region AS orig

FROM j\_1bnfdoc AS a

LEFT OUTER JOIN j\_1bbranch AS b

ON a~bukrs = b~bukrs AND

a~branch = b~branch AND

b~bupla\_type = @space

LEFT OUTER JOIN adrc AS c

ON c~addrnumber = b~adrnr

WHERE docnum = @iv\_docnum

INTO @DATA(ls\_j\_1bnfdoc).

\* Look for valid origir or destination.

\* Validate destination

CLEAR: ls\_uf\_dest,

ls\_uf\_branch.

\* If origin UF is the same as UF destination.

IF ls\_j\_1bnfdoc-dest = ls\_j\_1bnfdoc-orig.

\* Read UF destination for destination.

READ TABLE lt\_uf\_dest INTO ls\_uf\_dest

WITH KEY regio = ls\_j\_1bnfdoc-dest

bukrs = ls\_j\_1bnfdoc-bukrs.

IF sy-subrc = 0.

ls\_bran\_dest-uf = ls\_uf\_dest-regio.

ls\_bran\_dest-uf\_type = 'D'."Destination.

ls\_bran\_dest-ra\_only = ls\_uf\_dest-ra\_only.

\* Append to output parameter

APPEND ls\_bran\_dest TO et\_bran\_dest.

ELSE.

\* Read UF Branch Origin

READ TABLE lt\_uf\_branch INTO ls\_uf\_branch

WITH KEY regio = ls\_j\_1bnfdoc-orig

bukrs = ls\_j\_1bnfdoc-bukrs

branch = ls\_j\_1bnfdoc-branch.

IF sy-subrc = 0.

ls\_bran\_dest-uf = ls\_uf\_dest-regio.

ls\_bran\_dest-uf\_type = 'B'."Branch - (Origin)

ls\_bran\_dest-ra\_only = ls\_uf\_branch-ra\_only.

\* Append to output parameter

APPEND ls\_bran\_dest TO et\_bran\_dest.

ENDIF."sy-subrc READ TABLE lt\_uf\_branch

ENDIF."sy-subrc READ TABLE lt\_uf\_dest

ELSE.

\* Read UF destination for destination.

READ TABLE lt\_uf\_dest INTO ls\_uf\_dest

WITH KEY regio = ls\_j\_1bnfdoc-dest

bukrs = ls\_j\_1bnfdoc-bukrs.

IF sy-subrc = 0.

ls\_bran\_dest-uf = ls\_uf\_dest-regio.

ls\_bran\_dest-uf\_type = 'D'."Destination.

ls\_bran\_dest-ra\_only = ls\_uf\_dest-ra\_only.

\* Append to output parameter

APPEND ls\_bran\_dest TO et\_bran\_dest.

ENDIF.

\* Read UF Branch Origin

READ TABLE lt\_uf\_branch INTO ls\_uf\_branch

WITH KEY regio = ls\_j\_1bnfdoc-orig

bukrs = ls\_j\_1bnfdoc-bukrs

branch = ls\_j\_1bnfdoc-branch.

IF sy-subrc = 0.

ls\_bran\_dest-uf = ls\_uf\_branch-regio.

ls\_bran\_dest-uf\_type = 'B'."Branch - (Origin)

ls\_bran\_dest-ra\_only = ls\_uf\_branch-ra\_only.

\* Append to output parameter

APPEND ls\_bran\_dest TO et\_bran\_dest.

ENDIF."sy-subrc READ TABLE lt\_uf\_branch

ENDIF."ls\_j\_1bnfdoc-dest = ls\_j\_1bnfdoc-orig.

\* If there are no valid destination and origin.

IF et\_bran\_dest[] IS INITIAL.

\* Set invalid to True.

lv\_invalid = abap\_true.

ENDIF.

\* If no invalid condition was found.

IF lv\_invalid IS INITIAL.

\* Set valid to true.

ev\_valid = abap\_true.

ENDIF.

ENDMETHOD.

METHOD check\_completed\_nf.

\*&---------------------------------------------------------------------\*

\*& Method CHECK\_COMPLETED\_NF

\*----------------------------------------------------------------------\*

\* Identification

\* Author : Denis Pereira - EURQK

\* Creation date : 26.11.2024

\* Owner : Daniel Golin

\* Basis Release : 755

\*-----------------------------------------------------------------------

\* Description : Check obligatory fields.

\*----------------------------------------------------------------------\*

\* Changes \*

\* Vers. Date Author Request Description \*

\* V001 EURQK S1DK932604 created \*

\*----------------------------------------------------------------------\*

\* Local Types.

TYPES: BEGIN OF ts\_field\_map,

table TYPE char20,

field\_sel TYPE edpline,

END OF ts\_field\_map,

BEGIN OF ts\_field\_list,

field\_sel TYPE edpline,

END OF ts\_field\_list,

BEGIN OF ts\_fields\_qty,

table TYPE char20,

fields\_qty TYPE i,

END OF ts\_fields\_qty.

\* Local tables.

DATA: lt\_field\_map TYPE TABLE OF ts\_field\_map,

lt\_fields\_qty TYPE TABLE OF ts\_fields\_qty,

lt\_itm TYPE TABLE OF /bay0/o2c\_defitm,

lt\_lin TYPE TABLE OF j\_1bnflin,

lt\_app TYPE TABLE OF /bay0/o2c\_defapp,

lt\_mat TYPE TABLE OF /bay0/o2c\_defmat,

lt\_field\_list TYPE TABLE OF ts\_field\_list.

\* Local Ranges.

DATA: lt\_r\_param TYPE RANGE OF /bay0/tda\_devkey-param.

\* Local Structures

DATA: ls\_r\_param LIKE LINE OF lt\_r\_param,

ls\_field\_map TYPE ts\_field\_map,

ls\_hdr TYPE /bay0/o2c\_defhdr,

ls\_itm TYPE /bay0/o2c\_defitm,

ls\_doc TYPE j\_1bnfdoc,

ls\_lin TYPE j\_1bnflin,

ls\_branch TYPE j\_1bbranch,

ls\_branch\_adrc TYPE adrc,

ls\_active TYPE j\_1bnfe\_active,

ls\_cul TYPE /bay0/o2c\_defcul,

ls\_plg TYPE /bay0/o2c\_defplg,

ls\_app TYPE /bay0/o2c\_defapp,

ls\_mat TYPE /bay0/o2c\_defmat,

ls\_aum TYPE /bay0/o2c\_defaum,

ls\_agr TYPE /bay0/o2c\_defagr,

ls\_prp TYPE /bay0/o2c\_defprp,

ls\_itm\_mat TYPE /bay0/o2c\_defmat,

ls\_fields\_qty TYPE ts\_fields\_qty,

ls\_field\_list TYPE ts\_field\_list.

\* Local variables.

DATA: lv\_total\_fields TYPE i,

lv\_index TYPE sy-tabix,

lv\_selection TYPE string,

lv\_field\_name TYPE char50,

lv\_filled TYPE flag,

lv\_not\_compl TYPE flag,

lv\_doc\_cgc TYPE j\_1bnfdoc-cgc,

lv\_doc\_cpf TYPE j\_1bnfdoc-cpf,

lv\_dic\_field\_name TYPE char100,

lv\_comp\_param TYPE /bay0/tda\_devkey-param,

lv\_docnum\_char TYPE char10.

\* Field Symbos.

FIELD-SYMBOLS: <lf\_field> TYPE any.

\* If header is not provided in parameter.

IF is\_mem\_defhdr IS INITIAL.

\* Select layout from HDR

SELECT SINGLE layout

FROM /bay0/o2c\_defhdr

WHERE uf = @iv\_uf

AND docnum = @iv\_docnum

INTO @DATA(lv\_layout).

ELSE.

lv\_layout = is\_mem\_defhdr-layout.

ENDIF.

\* Fill Ranges parameters.

ls\_r\_param-sign = 'I'.

ls\_r\_param-option = 'CP'.

\* ls\_r\_param-low = 'GO\_EMRA\_\*'.

\* Ex: GO\_EMRA\_\*

CONCATENATE iv\_uf

'\_'

lv\_layout

'\_\*'

INTO ls\_r\_param-low.

\* Append range

APPEND ls\_r\_param TO lt\_r\_param.

\* Select Development key for global paramter

SELECT param, value

FROM /bay0/tda\_devkey

INTO TABLE @DATA(lt\_param\_value)

WHERE devkey = '/BAY0/O2C\_ICBR\_DEF'

AND param IN @lt\_r\_param

AND activ = @abap\_true.

\* Mount comparison parameter Ex: "GO\_EMRA\_"

CONCATENATE iv\_uf

'\_'

lv\_layout

'\_'

INTO lv\_comp\_param.

DATA(lv\_length) = strlen( lv\_comp\_param ).

\* Process constant lines.

LOOP AT lt\_param\_value INTO DATA(ls\_param\_value).

\* Clear working structures.

CLEAR: ls\_field\_map,

ls\_fields\_qty.

CASE ls\_param\_value-param.

WHEN OTHERS.

\* Fill CDS Fields

IF ls\_param\_value-param(lv\_length) = lv\_comp\_param."'GO\_EMRA\_'.

\* Fill the name which will be called in the output table.

SPLIT ls\_param\_value-value AT '-'

INTO ls\_field\_map-table

ls\_field\_map-field\_sel.

\* Append field map.

APPEND ls\_field\_map TO lt\_field\_map.

\* Collect number of fields per table.

ls\_fields\_qty-table = ls\_field\_map-table.

ls\_fields\_qty-fields\_qty = 1.

COLLECT ls\_fields\_qty INTO lt\_fields\_qty.

ENDIF."ls\_param\_value-param(8) = 'GO\_EMRA\_'

ENDCASE."ls\_param\_value-param.

ENDLOOP." AT lt\_param\_value

\* Sort Table map

SORT lt\_field\_map BY table field\_sel.

\* Remove leading zeros from document number.

CALL FUNCTION 'CONVERSION\_EXIT\_ALPHA\_OUTPUT'

EXPORTING

input = iv\_docnum

IMPORTING

output = lv\_docnum\_char.

\* \*\*\*\*\*\*\*\*\*\*\*CHECK J\_1BNFDOC \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Look for Nota Fiscal Header Obligatory fields.

READ TABLE lt\_field\_map INTO ls\_field\_map

WITH KEY table = 'DOC'.

IF sy-subrc = 0.

lv\_index = sy-tabix.

\* Clear working Tables.

REFRESH: lt\_field\_list.

\* Clear working structures and variables.

CLEAR: ls\_fields\_qty,

lv\_selection,

lv\_total\_fields.

\* Get fields quantity from constant table.

READ TABLE lt\_fields\_qty INTO ls\_fields\_qty

WITH KEY table = 'DOC'.

\* Read DOC table list to mount selection field list command.

LOOP AT lt\_field\_map INTO ls\_field\_map FROM lv\_index.

IF ls\_field\_map-table <> 'DOC'.

\* Exit Loop

EXIT.

ENDIF.

\* Increment control variable.

ADD 1 TO lv\_total\_fields.

\* Store field list in a table.

INSERT VALUE #( field\_sel = ls\_field\_map-field\_sel )

INTO TABLE lt\_field\_list.

\* If total processed fields is lower than the amount for this table.

IF lv\_total\_fields < ls\_fields\_qty-fields\_qty.

\* Add comma after field list

CONCATENATE ls\_field\_map-field\_sel

','

INTO ls\_field\_map-field\_sel.

ENDIF.

\* If selection is empty.

IF lv\_selection IS INITIAL.

lv\_selection = ls\_field\_map-field\_sel.

ELSE.

CONCATENATE lv\_selection

ls\_field\_map-field\_sel

INTO lv\_selection

SEPARATED BY space.

ENDIF." lv\_selection IS INITIAL

ENDLOOP."LOOP AT lt\_field\_map

TRY .

SELECT SINGLE (lv\_selection)

FROM j\_1bnfdoc

WHERE docnum = @iv\_docnum

INTO CORRESPONDING FIELDS OF @ls\_doc.

CATCH cx\_root.

\* Docnum: &1 UF: &2 dynamic SQL failed check constants &3.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 007

msgv1 = lv\_docnum\_char

msgv2 = iv\_uf

msgv3 = ls\_field\_map-table

probclass = '2' ) ) ). "2=Important

ENDTRY.

\* Sort Field list to become easier to visualize.

SORT lt\_field\_list.

\* Check if fields are not empty one by one.

LOOP AT lt\_field\_list INTO ls\_field\_list.

CLEAR: lv\_filled,

lv\_dic\_field\_name.

\* Concatenate structure field name.

CONCATENATE 'LS\_DOC-' ls\_field\_list-field\_sel

INTO lv\_field\_name.

ASSIGN (lv\_field\_name) TO <lf\_field>.

IF <lf\_field> IS ASSIGNED.

IF ls\_field\_list-field\_sel = 'CGC'.

lv\_doc\_cgc = <lf\_field>.

\* Avoid check for this field separated from CPF

UNASSIGN <lf\_field>.

CONTINUE.

ELSEIF ls\_field\_list-field\_sel = 'CPF'.

lv\_doc\_cpf = <lf\_field>.

UNASSIGN <lf\_field>.

CONTINUE.

ENDIF.

IF NOT <lf\_field> IS INITIAL.

lv\_filled = abap\_true.

ENDIF.

UNASSIGN <lf\_field>.

\* If field is empty.

IF lv\_filled IS INITIAL.

CONCATENATE 'J\_1BNFDOC-' ls\_field\_list-field\_sel

INTO lv\_dic\_field\_name.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

\* Set not completed flag to true allowing a RAISE clause

\* at the end of the process.

lv\_not\_compl = abap\_true.

ENDIF.

ENDIF."<lf\_field> IS ASSIGNED

ENDLOOP."AT lt\_field\_list

\* Check if both CPF and CGC are empty.

IF lv\_doc\_cgc IS INITIAL AND

lv\_doc\_cpf IS INITIAL.

lv\_dic\_field\_name = 'J\_1BNFDOC-CGC and J\_1BNFDOC-CPF'.

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

ENDIF."lv\_doc\_cgc IS INITIAL

ENDIF."READ TABLE lt\_field\_map

\* \*\*\*\*\*\*\*\*\*\*\*CHECK /BAY0/O2C\_DEFHDR \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Look for Selected Nota Fiscal – Header Obligatory fields.

READ TABLE lt\_field\_map INTO ls\_field\_map

WITH KEY table = 'HDR'.

IF sy-subrc = 0.

lv\_index = sy-tabix.

\* Clear working Tables.

REFRESH: lt\_field\_list.

\* Clear working structures and variables.

CLEAR: ls\_fields\_qty,

lv\_selection,

lv\_total\_fields.

\* Get fields quantity from constant table.

READ TABLE lt\_fields\_qty INTO ls\_fields\_qty

WITH KEY table = 'HDR'.

\* Read DOC table list to mount selection field list command.

LOOP AT lt\_field\_map INTO ls\_field\_map FROM lv\_index.

IF ls\_field\_map-table <> 'HDR'.

\* Exit Loop

EXIT.

ENDIF.

\* Increment control variable.

ADD 1 TO lv\_total\_fields.

\* Store field list in a table.

INSERT VALUE #( field\_sel = ls\_field\_map-field\_sel )

INTO TABLE lt\_field\_list.

\* If total processed fields is lower than the amount for this table.

IF lv\_total\_fields < ls\_fields\_qty-fields\_qty.

\* Add comma after field list

CONCATENATE ls\_field\_map-field\_sel

','

INTO ls\_field\_map-field\_sel.

ENDIF.

\* If selection is empty.

IF lv\_selection IS INITIAL.

lv\_selection = ls\_field\_map-field\_sel.

ELSE.

CONCATENATE lv\_selection

ls\_field\_map-field\_sel

INTO lv\_selection

SEPARATED BY space.

ENDIF." lv\_selection IS INITIAL

ENDLOOP."LOOP AT lt\_field\_map

\* If header is not provided in parameter.

IF is\_mem\_defhdr IS INITIAL.

TRY .

SELECT SINGLE (lv\_selection)

FROM /bay0/o2c\_defhdr

WHERE docnum = @iv\_docnum

INTO CORRESPONDING FIELDS OF @ls\_hdr.

CATCH cx\_root.

\* Docnum: &1 UF: &2 dynamic SQL failed check constants &3.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 007

msgv1 = iv\_docnum

msgv2 = iv\_uf

msgv3 = ls\_field\_map-table

probclass = '2' ) ) ). "2=Important

ENDTRY.

ELSE.

\* Fill HDR from provided Input parameter.

ls\_hdr = is\_mem\_defhdr.

ENDIF.

\* Sort Field list to become easier to visualize.

SORT lt\_field\_list.

\* Check if fields are not empty one by one.

LOOP AT lt\_field\_list INTO ls\_field\_list.

CLEAR: lv\_filled,

lv\_dic\_field\_name.

\* Concatenate structure field name.

CONCATENATE 'LS\_HDR-' ls\_field\_list-field\_sel

INTO lv\_field\_name.

ASSIGN (lv\_field\_name) TO <lf\_field>.

IF <lf\_field> IS ASSIGNED.

IF NOT <lf\_field> IS INITIAL.

lv\_filled = abap\_true.

ENDIF.

UNASSIGN <lf\_field>.

\* If field is empty.

IF lv\_filled IS INITIAL.

CONCATENATE '/BAY0/O2C\_DEFHDR-' ls\_field\_list-field\_sel

INTO lv\_dic\_field\_name.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

\* Set not completed flag to true allowing a RAISE clause

\* at the end of the process.

lv\_not\_compl = abap\_true.

ENDIF.

ENDIF."<lf\_field> IS ASSIGNED

ENDLOOP."AT lt\_field\_list

ENDIF."READ TABLE lt\_field\_map

\* \*\*\*\*\*\*\*\*\*\*\*CHECK /BAY0/O2C\_DEFAGR \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Look for Agronomist Obligatory fields.

READ TABLE lt\_field\_map INTO ls\_field\_map

WITH KEY table = 'AGR'.

IF sy-subrc = 0.

lv\_index = sy-tabix.

\* Clear working Tables.

REFRESH: lt\_field\_list.

\* Clear working structures and variables.

CLEAR: ls\_fields\_qty,

lv\_selection,

lv\_total\_fields.

\* Get fields quantity from constant table.

READ TABLE lt\_fields\_qty INTO ls\_fields\_qty

WITH KEY table = 'AGR'.

\* Read DOC table list to mount selection field list command.

LOOP AT lt\_field\_map INTO ls\_field\_map FROM lv\_index.

IF ls\_field\_map-table <> 'AGR'.

\* Exit Loop

EXIT.

ENDIF.

\* Increment control variable.

ADD 1 TO lv\_total\_fields.

\* Store field list in a table.

INSERT VALUE #( field\_sel = ls\_field\_map-field\_sel )

INTO TABLE lt\_field\_list.

\* If total processed fields is lower than the amount for this table.

IF lv\_total\_fields < ls\_fields\_qty-fields\_qty.

\* Add comma after field list

CONCATENATE ls\_field\_map-field\_sel

','

INTO ls\_field\_map-field\_sel.

ENDIF.

\* If selection is empty.

IF lv\_selection IS INITIAL.

lv\_selection = ls\_field\_map-field\_sel.

ELSE.

CONCATENATE lv\_selection

ls\_field\_map-field\_sel

INTO lv\_selection

SEPARATED BY space.

ENDIF." lv\_selection IS INITIAL

ENDLOOP."LOOP AT lt\_field\_map

TRY .

SELECT SINGLE (lv\_selection)

FROM /bay0/o2c\_defagr

WHERE uf = @iv\_uf

AND agrcpf = @ls\_hdr-agrcpf

INTO CORRESPONDING FIELDS OF @ls\_agr.

CATCH cx\_root.

\* Docnum: &1 UF: &2 dynamic SQL failed check constants &3.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 007

msgv1 = iv\_docnum

msgv2 = iv\_uf

msgv3 = ls\_field\_map-table

probclass = '2' ) ) ). "2=Important

ENDTRY.

\* Sort Field list to become easier to visualize.

SORT lt\_field\_list.

\* Check if fields are not empty one by one.

LOOP AT lt\_field\_list INTO ls\_field\_list.

CLEAR: lv\_filled,

lv\_dic\_field\_name.

\* Concatenate structure field name.

CONCATENATE 'LS\_AGR-' ls\_field\_list-field\_sel

INTO lv\_field\_name.

ASSIGN (lv\_field\_name) TO <lf\_field>.

IF <lf\_field> IS ASSIGNED.

IF NOT <lf\_field> IS INITIAL.

lv\_filled = abap\_true.

ENDIF.

UNASSIGN <lf\_field>.

\* If field is empty.

IF lv\_filled IS INITIAL.

CONCATENATE '/BAY0/O2C\_DEFAGR-' ls\_field\_list-field\_sel

INTO lv\_dic\_field\_name.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defAGR-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

\* Set not completed flag to true allowing a RAISE clause

\* at the end of the process.

lv\_not\_compl = abap\_true.

ENDIF.

ENDIF."<lf\_field> IS ASSIGNED

ENDLOOP."AT lt\_field\_list

ENDIF."READ TABLE lt\_field\_map

\* \*\*\*\*\*\*\*\*\*\*\*CHECK /BAY0/O2C\_DEFPRP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Look for Property Obligatory fields.

READ TABLE lt\_field\_map INTO ls\_field\_map

WITH KEY table = 'PRP'.

IF sy-subrc = 0.

lv\_index = sy-tabix.

\* Clear working Tables.

REFRESH: lt\_field\_list.

\* Clear working structures and variables.

CLEAR: ls\_fields\_qty,

lv\_selection,

lv\_total\_fields.

\* Get fields quantity from constant table.

READ TABLE lt\_fields\_qty INTO ls\_fields\_qty

WITH KEY table = 'PRP'.

\* Read DOC table list to mount selection field list command.

LOOP AT lt\_field\_map INTO ls\_field\_map FROM lv\_index.

IF ls\_field\_map-table <> 'PRP'.

\* Exit Loop

EXIT.

ENDIF.

\* Increment control variable.

ADD 1 TO lv\_total\_fields.

\* Store field list in a table.

INSERT VALUE #( field\_sel = ls\_field\_map-field\_sel )

INTO TABLE lt\_field\_list.

\* If total processed fields is lower than the amount for this table.

IF lv\_total\_fields < ls\_fields\_qty-fields\_qty.

\* Add comma after field list

CONCATENATE ls\_field\_map-field\_sel

','

INTO ls\_field\_map-field\_sel.

ENDIF.

\* If selection is empty.

IF lv\_selection IS INITIAL.

lv\_selection = ls\_field\_map-field\_sel.

ELSE.

CONCATENATE lv\_selection

ls\_field\_map-field\_sel

INTO lv\_selection

SEPARATED BY space.

ENDIF." lv\_selection IS INITIAL

ENDLOOP."LOOP AT lt\_field\_map

TRY .

SELECT SINGLE (lv\_selection)

FROM /bay0/o2c\_defprp

WHERE uf = @iv\_uf

AND kunnr = @ls\_hdr-parid

INTO CORRESPONDING FIELDS OF @ls\_prp.

CATCH cx\_root.

\* Docnum: &1 UF: &2 dynamic SQL failed check constants &3.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 007

msgv1 = iv\_docnum

msgv2 = iv\_uf

msgv3 = ls\_field\_map-table

probclass = '2' ) ) ). "2=Important

ENDTRY.

\* Sort Field list to become easier to visualize.

SORT lt\_field\_list.

\* Check if fields are not empty one by one.

LOOP AT lt\_field\_list INTO ls\_field\_list.

CLEAR: lv\_filled,

lv\_dic\_field\_name.

\* Concatenate structure field name.

CONCATENATE 'LS\_PRP-' ls\_field\_list-field\_sel

INTO lv\_field\_name.

ASSIGN (lv\_field\_name) TO <lf\_field>.

IF <lf\_field> IS ASSIGNED.

IF NOT <lf\_field> IS INITIAL.

lv\_filled = abap\_true.

ENDIF.

UNASSIGN <lf\_field>.

\* If field is empty.

IF lv\_filled IS INITIAL.

CONCATENATE '/BAY0/O2C\_DEFPRP-' ls\_field\_list-field\_sel

INTO lv\_dic\_field\_name.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defAGR-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

\* Set not completed flag to true allowing a RAISE clause

\* at the end of the process.

lv\_not\_compl = abap\_true.

ENDIF.

ENDIF."<lf\_field> IS ASSIGNED

ENDLOOP."AT lt\_field\_list

ENDIF."READ TABLE lt\_field\_map

\* \*\*\*\*\*\*\*\*\*\*\*CHECK J\_1BNFE\_ACTIVE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Look for Electronic Nota Fiscal: Actual Status Obligatory fields.

READ TABLE lt\_field\_map INTO ls\_field\_map

WITH KEY table = 'ACTIVE'.

IF sy-subrc = 0.

lv\_index = sy-tabix.

\* Clear working Tables.

REFRESH: lt\_field\_list.

\* Clear working structures and variables.

CLEAR: ls\_fields\_qty,

lv\_selection,

lv\_total\_fields.

\* Get fields quantity from constant table.

READ TABLE lt\_fields\_qty INTO ls\_fields\_qty

WITH KEY table = 'ACTIVE'.

\* Read DOC table list to mount selection field list command.

LOOP AT lt\_field\_map INTO ls\_field\_map FROM lv\_index.

IF ls\_field\_map-table <> 'ACTIVE'.

\* Exit Loop

EXIT.

ENDIF.

\* Increment control variable.

ADD 1 TO lv\_total\_fields.

\* Store field list in a table.

INSERT VALUE #( field\_sel = ls\_field\_map-field\_sel )

INTO TABLE lt\_field\_list.

\* If total processed fields is lower than the amount for this table.

IF lv\_total\_fields < ls\_fields\_qty-fields\_qty.

\* Add comma after field list

CONCATENATE ls\_field\_map-field\_sel

','

INTO ls\_field\_map-field\_sel.

ENDIF.

\* If selection is empty.

IF lv\_selection IS INITIAL.

lv\_selection = ls\_field\_map-field\_sel.

ELSE.

CONCATENATE lv\_selection

ls\_field\_map-field\_sel

INTO lv\_selection

SEPARATED BY space.

ENDIF." lv\_selection IS INITIAL

ENDLOOP."LOOP AT lt\_field\_map

TRY .

SELECT SINGLE (lv\_selection)

FROM j\_1bnfe\_active

WHERE docnum = @iv\_docnum

INTO CORRESPONDING FIELDS OF @ls\_active.

CATCH cx\_root.

\* Docnum: &1 UF: &2 dynamic SQL failed check constants &3.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 007

msgv1 = iv\_docnum

msgv2 = iv\_uf

msgv3 = ls\_field\_map-table

probclass = '2' ) ) ). "2=Important

ENDTRY.

\* Sort Field list to become easier to visualize.

SORT lt\_field\_list.

\* Check if fields are not empty one by one.

LOOP AT lt\_field\_list INTO ls\_field\_list.

CLEAR: lv\_filled,

lv\_dic\_field\_name.

\* Concatenate structure field name.

CONCATENATE 'LS\_ACTIVE-' ls\_field\_list-field\_sel

INTO lv\_field\_name.

ASSIGN (lv\_field\_name) TO <lf\_field>.

IF <lf\_field> IS ASSIGNED.

IF NOT <lf\_field> IS INITIAL.

lv\_filled = abap\_true.

ENDIF.

UNASSIGN <lf\_field>.

\* If field is empty.

IF lv\_filled IS INITIAL.

CONCATENATE 'J\_1BNFE\_ACTIVE-' ls\_field\_list-field\_sel

INTO lv\_dic\_field\_name.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defAGR-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

\* Set not completed flag to true allowing a RAISE clause

\* at the end of the process.

lv\_not\_compl = abap\_true.

ENDIF.

ENDIF."<lf\_field> IS ASSIGNED

ENDLOOP."AT lt\_field\_list

ENDIF."READ TABLE lt\_field\_map

\* \*\*\*\*\*\*\*\*\*\*\*CHECK J\_1BBRANCH \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Look for Business Place Obligatory fields.

READ TABLE lt\_field\_map INTO ls\_field\_map

WITH KEY table = 'BRANCH'.

IF sy-subrc = 0.

lv\_index = sy-tabix.

\* Clear working Tables.

REFRESH: lt\_field\_list.

\* Clear working structures and variables.

CLEAR: ls\_fields\_qty,

lv\_selection,

lv\_total\_fields.

\* Get fields quantity from constant table.

READ TABLE lt\_fields\_qty INTO ls\_fields\_qty

WITH KEY table = 'BRANCH'.

\* Read DOC table list to mount selection field list command.

LOOP AT lt\_field\_map INTO ls\_field\_map FROM lv\_index.

IF ls\_field\_map-table <> 'BRANCH'.

\* Exit Loop

EXIT.

ENDIF.

\* Increment control variable.

ADD 1 TO lv\_total\_fields.

\* Store field list in a table.

INSERT VALUE #( field\_sel = ls\_field\_map-field\_sel )

INTO TABLE lt\_field\_list.

\* If total processed fields is lower than the amount for this table.

IF lv\_total\_fields < ls\_fields\_qty-fields\_qty.

\* Add comma after field list

CONCATENATE ls\_field\_map-field\_sel

','

INTO ls\_field\_map-field\_sel.

ENDIF.

\* If selection is empty.

IF lv\_selection IS INITIAL.

lv\_selection = ls\_field\_map-field\_sel.

ELSE.

CONCATENATE lv\_selection

ls\_field\_map-field\_sel

INTO lv\_selection

SEPARATED BY space.

ENDIF." lv\_selection IS INITIAL

ENDLOOP."LOOP AT lt\_field\_map

TRY .

SELECT SINGLE (lv\_selection)

FROM j\_1bbranch

WHERE bukrs = @ls\_hdr-bukrs

AND branch = @ls\_hdr-branch

INTO CORRESPONDING FIELDS OF @ls\_branch.

CATCH cx\_root.

\* Docnum: &1 UF: &2 dynamic SQL failed check constants &3.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 007

msgv1 = iv\_docnum

msgv2 = iv\_uf

msgv3 = ls\_field\_map-table

probclass = '2' ) ) ). "2=Important

ENDTRY.

\* Sort Field list to become easier to visualize.

SORT lt\_field\_list.

\* Check if fields are not empty one by one.

LOOP AT lt\_field\_list INTO ls\_field\_list.

CLEAR: lv\_filled,

lv\_dic\_field\_name.

\* Concatenate structure field name.

CONCATENATE 'LS\_BRANCH-' ls\_field\_list-field\_sel

INTO lv\_field\_name.

ASSIGN (lv\_field\_name) TO <lf\_field>.

IF <lf\_field> IS ASSIGNED.

IF NOT <lf\_field> IS INITIAL.

lv\_filled = abap\_true.

ENDIF.

UNASSIGN <lf\_field>.

\* If field is empty.

IF lv\_filled IS INITIAL.

CONCATENATE 'J\_1BBRANCH-' ls\_field\_list-field\_sel

INTO lv\_dic\_field\_name.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defAGR-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

\* Set not completed flag to true allowing a RAISE clause

\* at the end of the process.

lv\_not\_compl = abap\_true.

ENDIF.

ENDIF."<lf\_field> IS ASSIGNED

ENDLOOP."AT lt\_field\_list

ENDIF."READ TABLE lt\_field\_map

\* \*\*\*\*\*\*\*\*\*\*\*CHECK ADRC J\_1BBRANCH \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Look for Business Place Address Obligatory fields.

READ TABLE lt\_field\_map INTO ls\_field\_map

WITH KEY table = 'BRANCH\_ADRC'.

IF sy-subrc = 0.

lv\_index = sy-tabix.

\* Clear working Tables.

REFRESH: lt\_field\_list.

\* Clear working structures and variables.

CLEAR: ls\_fields\_qty,

lv\_selection,

lv\_total\_fields.

\* Get fields quantity from constant table.

READ TABLE lt\_fields\_qty INTO ls\_fields\_qty

WITH KEY table = 'BRANCH\_ADRC'.

\* Read DOC table list to mount selection field list command.

LOOP AT lt\_field\_map INTO ls\_field\_map FROM lv\_index.

IF ls\_field\_map-table <> 'BRANCH\_ADRC'.

\* Exit Loop

EXIT.

ENDIF.

\* Increment control variable.

ADD 1 TO lv\_total\_fields.

\* Store field list in a table.

INSERT VALUE #( field\_sel = ls\_field\_map-field\_sel )

INTO TABLE lt\_field\_list.

\* If total processed fields is lower than the amount for this table.

IF lv\_total\_fields < ls\_fields\_qty-fields\_qty.

\* Add comma after field list

CONCATENATE ls\_field\_map-field\_sel

','

INTO ls\_field\_map-field\_sel.

ENDIF.

\* If selection is empty.

IF lv\_selection IS INITIAL.

lv\_selection = ls\_field\_map-field\_sel.

ELSE.

CONCATENATE lv\_selection

ls\_field\_map-field\_sel

INTO lv\_selection

SEPARATED BY space.

ENDIF." lv\_selection IS INITIAL

ENDLOOP."LOOP AT lt\_field\_map

TRY .

SELECT SINGLE (lv\_selection)

FROM adrc

WHERE addrnumber = @ls\_branch-adrnr

AND date\_from <= @sy-datum

AND date\_to >= @sy-datum

INTO CORRESPONDING FIELDS OF @ls\_branch\_adrc.

CATCH cx\_root.

\* Docnum: &1 UF: &2 dynamic SQL failed check constants &3.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 007

msgv1 = iv\_docnum

msgv2 = iv\_uf

msgv3 = ls\_field\_map-table

probclass = '2' ) ) ). "2=Important

ENDTRY.

\* Sort Field list to become easier to visualize.

SORT lt\_field\_list.

\* Check if fields are not empty one by one.

LOOP AT lt\_field\_list INTO ls\_field\_list.

CLEAR: lv\_filled,

lv\_dic\_field\_name.

\* Concatenate structure field name.

CONCATENATE 'LS\_BRANCH\_ADRC-' ls\_field\_list-field\_sel

INTO lv\_field\_name.

ASSIGN (lv\_field\_name) TO <lf\_field>.

IF <lf\_field> IS ASSIGNED.

IF NOT <lf\_field> IS INITIAL.

lv\_filled = abap\_true.

ENDIF.

UNASSIGN <lf\_field>.

\* If field is empty.

IF lv\_filled IS INITIAL.

CONCATENATE 'BRANCH\_ADRC-' ls\_field\_list-field\_sel

INTO lv\_dic\_field\_name.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defAGR-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

\* Set not completed flag to true allowing a RAISE clause

\* at the end of the process.

lv\_not\_compl = abap\_true.

ENDIF.

ENDIF."<lf\_field> IS ASSIGNED

ENDLOOP."AT lt\_field\_list

ENDIF."READ TABLE lt\_field\_map

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ITEMS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*\*\*\*\*\*\*\*\*\*\*CHECK /BAY0/O2C\_DEFITM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Look for Selected Nota Fiscal – Item Obligatory fields.

READ TABLE lt\_field\_map INTO ls\_field\_map

WITH KEY table = 'ITM'.

IF sy-subrc = 0.

lv\_index = sy-tabix.

\* Clear working Tables.

REFRESH: lt\_field\_list.

\* Clear working structures and variables.

CLEAR: ls\_fields\_qty,

lv\_selection,

lv\_total\_fields.

\* Get fields quantity from constant table.

READ TABLE lt\_fields\_qty INTO ls\_fields\_qty

WITH KEY table = 'ITM'.

\* Read DOC table list to mount selection field list command.

LOOP AT lt\_field\_map INTO ls\_field\_map FROM lv\_index.

IF ls\_field\_map-table <> 'ITM'.

\* Exit Loop

EXIT.

ENDIF.

\* Increment control variable.

ADD 1 TO lv\_total\_fields.

\* Store field list in a table.

INSERT VALUE #( field\_sel = ls\_field\_map-field\_sel )

INTO TABLE lt\_field\_list.

\* If total processed fields is lower than the amount for this table.

IF lv\_total\_fields < ls\_fields\_qty-fields\_qty.

\* Add comma after field list

CONCATENATE ls\_field\_map-field\_sel

','

INTO ls\_field\_map-field\_sel.

ENDIF.

\* If selection is empty.

IF lv\_selection IS INITIAL.

lv\_selection = ls\_field\_map-field\_sel.

ELSE.

CONCATENATE lv\_selection

ls\_field\_map-field\_sel

INTO lv\_selection

SEPARATED BY space.

ENDIF." lv\_selection IS INITIAL

ENDLOOP."LOOP AT lt\_field\_map

\* If items are not provided in the table parameter

IF it\_mem\_defitm[] IS INITIAL.

TRY .

SELECT (lv\_selection)

FROM /bay0/o2c\_defitm

WHERE uf = @iv\_uf

AND docnum = @iv\_docnum

INTO CORRESPONDING FIELDS OF TABLE @lt\_itm.

\* Sort by key

SORT lt\_itm BY docnum itmnum.

CATCH cx\_root.

\* Docnum: &1 UF: &2 dynamic SQL failed check constants &3.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 007

msgv1 = iv\_docnum

msgv2 = iv\_uf

msgv3 = ls\_field\_map-table

probclass = '2' ) ) ). "2=Important

ENDTRY.

ELSE.

\* Get table from parameters.

lt\_itm[] = it\_mem\_defitm[].

ENDIF."

\* Sort Field list to become easier to visualize.

SORT lt\_field\_list.

\* Process each item available.

LOOP AT lt\_itm INTO ls\_itm.

\* Check if fields are not empty one by one.

LOOP AT lt\_field\_list INTO ls\_field\_list.

CLEAR: lv\_filled,

lv\_dic\_field\_name.

\* Concatenate structure field name.

CONCATENATE 'LS\_ITM-' ls\_field\_list-field\_sel

INTO lv\_field\_name.

ASSIGN (lv\_field\_name) TO <lf\_field>.

IF <lf\_field> IS ASSIGNED.

IF NOT <lf\_field> IS INITIAL.

lv\_filled = abap\_true.

ENDIF.

UNASSIGN <lf\_field>.

\* If field is empty.

IF lv\_filled IS INITIAL.

CONCATENATE '/BAY0/O2C\_DEFITM-' ls\_field\_list-field\_sel

INTO lv\_dic\_field\_name.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = ls\_itm-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

\* Set not completed flag to true allowing a RAISE clause

\* at the end of the process.

lv\_not\_compl = abap\_true.

ENDIF.

ENDIF."<lf\_field> IS ASSIGNED

ENDLOOP."AT lt\_field\_list

ENDLOOP."AT lt\_itm

ENDIF."READ TABLE lt\_field\_map

IF NOT lt\_itm[] IS INITIAL.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CHECK J\_1BNFLIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Look for Nota Fiscal line items Obligatory fields.

READ TABLE lt\_field\_map INTO ls\_field\_map

WITH KEY table = 'LIN'.

IF sy-subrc = 0.

lv\_index = sy-tabix.

\* Clear working Tables.

REFRESH: lt\_field\_list.

\* Clear working structures and variables.

CLEAR: ls\_fields\_qty,

lv\_selection,

lv\_total\_fields.

\* Get fields quantity from constant table.

READ TABLE lt\_fields\_qty INTO ls\_fields\_qty

WITH KEY table = 'LIN'.

\* Read DOC table list to mount selection field list command.

LOOP AT lt\_field\_map INTO ls\_field\_map FROM lv\_index.

IF ls\_field\_map-table <> 'LIN'.

\* Exit Loop

EXIT.

ENDIF.

\* Increment control variable.

ADD 1 TO lv\_total\_fields.

\* Store field list in a table.

INSERT VALUE #( field\_sel = ls\_field\_map-field\_sel )

INTO TABLE lt\_field\_list.

\* If total processed fields is lower than the amount for this table.

IF lv\_total\_fields < ls\_fields\_qty-fields\_qty.

\* Add comma after field list

CONCATENATE ls\_field\_map-field\_sel

','

INTO ls\_field\_map-field\_sel.

ENDIF.

\* If selection is empty.

IF lv\_selection IS INITIAL.

lv\_selection = ls\_field\_map-field\_sel.

ELSE.

CONCATENATE lv\_selection

ls\_field\_map-field\_sel

INTO lv\_selection

SEPARATED BY space.

ENDIF." lv\_selection IS INITIAL

ENDLOOP."LOOP AT lt\_field\_map

TRY .

SELECT (lv\_selection)

FROM j\_1bnflin

FOR ALL ENTRIES IN @lt\_itm[]

WHERE docnum = @lt\_itm-docnum

AND itmnum = @lt\_itm-itmnum

INTO CORRESPONDING FIELDS OF TABLE @lt\_lin.

\* Sort by key

SORT lt\_lin BY docnum itmnum.

CATCH cx\_root.

\* Docnum: &1 UF: &2 dynamic SQL failed check constants &3.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 007

msgv1 = iv\_docnum

msgv2 = iv\_uf

msgv3 = ls\_field\_map-table

probclass = '2' ) ) ). "2=Important

ENDTRY.

\* Sort Field list to become easier to visualize.

SORT lt\_field\_list.

\* Process each item available.

LOOP AT lt\_lin INTO ls\_lin.

\* Check if fields are not empty one by one.

LOOP AT lt\_field\_list INTO ls\_field\_list.

CLEAR: lv\_filled,

lv\_dic\_field\_name.

\* Concatenate structure field name.

CONCATENATE 'LS\_LIN-' ls\_field\_list-field\_sel

INTO lv\_field\_name.

ASSIGN (lv\_field\_name) TO <lf\_field>.

IF <lf\_field> IS ASSIGNED.

IF NOT <lf\_field> IS INITIAL.

lv\_filled = abap\_true.

ENDIF.

UNASSIGN <lf\_field>.

\* If field is empty.

IF lv\_filled IS INITIAL.

CONCATENATE 'J\_1BNFLIN-' ls\_field\_list-field\_sel

INTO lv\_dic\_field\_name.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = ls\_lin-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

\* Set not completed flag to true allowing a RAISE clause

\* at the end of the process.

lv\_not\_compl = abap\_true.

ENDIF.

ENDIF."<lf\_field> IS ASSIGNED

ENDLOOP."AT lt\_field\_list

ENDLOOP."AT lt\_LIN

ENDIF."READ TABLE lt\_field\_map

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CHECK /BAY0/O2C\_DEFAPP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Look for Nota Fiscal line items Obligatory fields.

READ TABLE lt\_field\_map INTO ls\_field\_map

WITH KEY table = 'APP'.

IF sy-subrc = 0.

lv\_index = sy-tabix.

\* Clear working Tables.

REFRESH: lt\_field\_list.

\* Clear working structures and variables.

CLEAR: ls\_fields\_qty,

lv\_selection,

lv\_total\_fields.

\* Get fields quantity from constant table.

READ TABLE lt\_fields\_qty INTO ls\_fields\_qty

WITH KEY table = 'APP'.

\* Start with key without including in the check procedure.

lv\_selection = 'APLCD,'.

\* Read DOC table list to mount selection field list command.

LOOP AT lt\_field\_map INTO ls\_field\_map FROM lv\_index.

IF ls\_field\_map-table <> 'APP'.

\* Exit Loop

EXIT.

ENDIF.

\* Increment control variable.

ADD 1 TO lv\_total\_fields.

\* Store field list in a table.

INSERT VALUE #( field\_sel = ls\_field\_map-field\_sel )

INTO TABLE lt\_field\_list.

\* If total processed fields is lower than the amount for this table.

IF lv\_total\_fields < ls\_fields\_qty-fields\_qty.

\* Add comma after field list

CONCATENATE ls\_field\_map-field\_sel

','

INTO ls\_field\_map-field\_sel.

ENDIF.

\* If selection is empty.

IF lv\_selection IS INITIAL.

lv\_selection = ls\_field\_map-field\_sel.

ELSE.

CONCATENATE lv\_selection

ls\_field\_map-field\_sel

INTO lv\_selection

SEPARATED BY space.

ENDIF." lv\_selection IS INITIAL

ENDLOOP."LOOP AT lt\_field\_map

TRY .

SELECT (lv\_selection)

FROM /bay0/o2c\_defapp

FOR ALL ENTRIES IN @lt\_itm[]

WHERE uf = @iv\_uf

AND aplcd = @lt\_itm-aplcd

INTO CORRESPONDING FIELDS OF TABLE @lt\_app.

\* Sort by key.

SORT lt\_app BY aplcd.

CATCH cx\_root.

\* Docnum: &1 UF: &2 dynamic SQL failed check constants &3.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 007

msgv1 = iv\_docnum

msgv2 = iv\_uf

msgv3 = ls\_field\_map-table

probclass = '2' ) ) ). "2=Important

ENDTRY.

\* Sort Field list to become easier to visualize.

SORT lt\_field\_list.

\* Process each item.

LOOP AT lt\_itm INTO ls\_itm.

\* Clear working structure

CLEAR: ls\_app.

\* Read corresponding application.

READ TABLE lt\_app INTO ls\_app

WITH KEY aplcd = ls\_itm-aplcd.

\* Check if fields are not empty one by one.

LOOP AT lt\_field\_list INTO ls\_field\_list.

CLEAR: lv\_filled,

lv\_dic\_field\_name.

\* Concatenate structure field name.

CONCATENATE 'LS\_APP-' ls\_field\_list-field\_sel

INTO lv\_field\_name.

ASSIGN (lv\_field\_name) TO <lf\_field>.

IF <lf\_field> IS ASSIGNED.

IF NOT <lf\_field> IS INITIAL.

lv\_filled = abap\_true.

ENDIF.

UNASSIGN <lf\_field>.

\* If field is empty.

IF lv\_filled IS INITIAL.

CONCATENATE '/BAY0/O2C\_DEFAPP-' ls\_field\_list-field\_sel

INTO lv\_dic\_field\_name.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = ls\_itm-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

\* Set not completed flag to true allowing a RAISE clause

\* at the end of the process.

lv\_not\_compl = abap\_true.

ENDIF.

ENDIF."<lf\_field> IS ASSIGNED

ENDLOOP."AT lt\_field\_list

ENDLOOP." AT lt\_itm

ENDIF."READ TABLE lt\_field\_map

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CHECK /BAY0/O2C\_DEFMAT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Look for Defensives Control - UF Material Conversion Obligatory fields.

READ TABLE lt\_field\_map INTO ls\_field\_map

WITH KEY table = 'MAT'.

IF sy-subrc = 0.

lv\_index = sy-tabix.

\* Clear working Tables.

REFRESH: lt\_field\_list.

\* Clear working structures and variables.

CLEAR: ls\_fields\_qty,

lv\_selection,

lv\_total\_fields.

\* Get fields quantity from constant table.

READ TABLE lt\_fields\_qty INTO ls\_fields\_qty

WITH KEY table = 'MAT'.

\* Start with key without including in the check procedure.

lv\_selection = 'MATNR,'.

\* Read DOC table list to mount selection field list command.

LOOP AT lt\_field\_map INTO ls\_field\_map FROM lv\_index.

IF ls\_field\_map-table <> 'MAT'.

\* Exit Loop

EXIT.

ENDIF.

\* Increment control variable.

ADD 1 TO lv\_total\_fields.

\* Store field list in a table.

INSERT VALUE #( field\_sel = ls\_field\_map-field\_sel )

INTO TABLE lt\_field\_list.

\* If total processed fields is lower than the amount for this table.

IF lv\_total\_fields < ls\_fields\_qty-fields\_qty.

\* Add comma after field list

CONCATENATE ls\_field\_map-field\_sel

','

INTO ls\_field\_map-field\_sel.

ENDIF.

\* If selection is empty.

IF lv\_selection IS INITIAL.

lv\_selection = ls\_field\_map-field\_sel.

ELSE.

CONCATENATE lv\_selection

ls\_field\_map-field\_sel

INTO lv\_selection

SEPARATED BY space.

ENDIF." lv\_selection IS INITIAL

ENDLOOP."LOOP AT lt\_field\_map

TRY .

SELECT (lv\_selection)

FROM /bay0/o2c\_defmat

FOR ALL ENTRIES IN @lt\_itm[]

WHERE uf = @iv\_uf

AND matnr = @lt\_itm-matnr

INTO CORRESPONDING FIELDS OF TABLE @lt\_mat.

\* Sort by key.

SORT lt\_mat BY matnr.

CATCH cx\_root.

\* Docnum: &1 UF: &2 dynamic SQL failed check constants &3.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 007

msgv1 = iv\_docnum

msgv2 = iv\_uf

msgv3 = ls\_field\_map-table

probclass = '2' ) ) ). "2=Important

ENDTRY.

\* Sort Field list to become easier to visualize.

SORT lt\_field\_list.

\* Process each item.

LOOP AT lt\_itm INTO ls\_itm.

\* Clear working structure

CLEAR: ls\_mat.

\* Read corresponding UF Material Conversion.

READ TABLE lt\_mat INTO ls\_mat

WITH KEY matnr = ls\_itm-matnr.

\* Check if fields are not empty one by one.

LOOP AT lt\_field\_list INTO ls\_field\_list.

CLEAR: lv\_filled,

lv\_dic\_field\_name.

\* Concatenate structure field name.

CONCATENATE 'LS\_MAT-' ls\_field\_list-field\_sel

INTO lv\_field\_name.

ASSIGN (lv\_field\_name) TO <lf\_field>.

IF <lf\_field> IS ASSIGNED.

IF NOT <lf\_field> IS INITIAL.

lv\_filled = abap\_true.

ENDIF.

UNASSIGN <lf\_field>.

\* If field is empty.

IF lv\_filled IS INITIAL.

CONCATENATE '/BAY0/O2C\_DEFMAT-' ls\_field\_list-field\_sel

INTO lv\_dic\_field\_name.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = ls\_itm-itmnum

iv\_status = '1' "Incomplete

iv\_save\_file = ''

\* is\_defhdr = es\_defhdr

\* it\_defitm = et\_defitm

\* Docnum: &1 UF: &2 field &3 is empty.

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 004 msgv1 = lv\_docnum\_char msgv2 = iv\_uf msgv3 = lv\_dic\_field\_name

probclass = '2' ) ) ). "2=Important

\* Set not completed flag to true allowing a RAISE clause

\* at the end of the process.

lv\_not\_compl = abap\_true.

ENDIF.

ENDIF."<lf\_field> IS ASSIGNED

ENDLOOP."AT lt\_field\_list

ENDLOOP." AT lt\_itm

ENDIF."READ TABLE lt\_field\_map

ENDIF."NOT lt\_itm[] IS INITIAL

IF lv\_not\_compl = abap\_true.

\* Raise Not Completed Exception.

RAISE nfe\_not\_completed.

ENDIF.

ENDMETHOD.

METHOD cpi\_set\_finished.

SELECT SINGLE \*

FROM /bay0/o2c\_defhdr

INTO @DATA(ls\_hdr)

WHERE docnum = @iv\_docnum

AND uf = @iv\_uf.

IF sy-subrc = 0.

ls\_hdr-status = '5'.

IF iv\_receita IS SUPPLIED AND iv\_receita IS NOT INITIAL.

ls\_hdr-nu\_receita = iv\_receita.

ENDIF.

UPDATE /bay0/o2c\_defhdr FROM ls\_hdr.

IF sy-subrc = 0.

COMMIT WORK AND WAIT.

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '5' "Finished

iv\_save\_file = abap\_true

iv\_file\_raw = iv\_file\_raw

iv\_mime\_type = iv\_mime\_type

it\_messages = it\_messages

"Docnum: &1 UF: &2 set as finished.

it\_balmsg = VALUE #( ( msgty = 'S' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 009 msgv1 = iv\_docnum msgv2 = iv\_uf

probclass = '2' ) ) ). "2=Important

ENDIF.

ENDIF.

ENDMETHOD.

METHOD cpi\_set\_rejected.

SELECT SINGLE \*

FROM /bay0/o2c\_defhdr

INTO @DATA(ls\_hdr)

WHERE docnum = @iv\_docnum

AND uf = @iv\_uf.

IF sy-subrc = 0.

ls\_hdr-status = '4'.

IF iv\_receita IS SUPPLIED AND iv\_receita IS NOT INITIAL.

ls\_hdr-nu\_receita = iv\_receita.

ENDIF.

UPDATE /bay0/o2c\_defhdr FROM ls\_hdr.

IF sy-subrc = 0.

COMMIT WORK AND WAIT.

me->save\_log(

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf = iv\_uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '4' "Rejected

iv\_save\_file = abap\_true

iv\_file\_raw = iv\_file\_raw

iv\_mime\_type = iv\_mime\_type

it\_messages = it\_messages

"Docnum: &1 UF: &2 set as rejected

it\_balmsg = VALUE #( ( msgty = 'E' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 010 msgv1 = iv\_docnum msgv2 = iv\_uf

probclass = '2' ) ) ). "2=Important

ENDIF.

ENDIF.

ENDMETHOD.

METHOD def\_nf\_complete\_process.

\*&---------------------------------------------------------------------\*

\*& Method DEF\_NF\_COMPLETE\_PROCESS

\*----------------------------------------------------------------------\*

\* Identification

\* Author : Denis Pereira - EURQK

\* Creation date : 26.11.2024

\* Owner : Daniel Golin

\* Basis Release : 755

\*-----------------------------------------------------------------------

\* Description : Execute defensive full process to feed defensive data

\* and send message to CPI.

\*----------------------------------------------------------------------\*

\* Changes \*

\* Vers. Date Author Request Description \*

\* V001 EURQK S1DK932604 created \*

\*----------------------------------------------------------------------\*

\* Local internal tables.

DATA: lt\_bran\_dest TYPE /bay0/o2c\_t\_def\_bran\_dest,

lt\_defitm TYPE /bay0/o2c\_t\_defitm.

\* Local structures.

DATA: ls\_defhdr TYPE /bay0/o2c\_defhdr.

\* Local Variables.

DATA: lv\_status TYPE status2\_br,

lv\_valid TYPE flag.

\* Get valid branch and destinations

CALL METHOD me->check\_valid\_nf

EXPORTING

iv\_docnum = iv\_docnum

\* iv\_code =

IMPORTING

et\_bran\_dest = lt\_bran\_dest

RECEIVING

ev\_valid = lv\_valid.

\* Process each available document (Branch - 'origin' / Destination )

LOOP AT lt\_bran\_dest INTO DATA(ls\_bran\_dest).

\* Clear working memory

CLEAR: ls\_defhdr.

REFRESH: lt\_defitm.

\* Feed Defensive tables

CALL METHOD me->def\_nf\_feed

EXPORTING

iv\_docnum = iv\_docnum

iv\_uf\_type = ls\_bran\_dest-uf\_type

iv\_uf = ls\_bran\_dest-uf

iv\_ra\_only = ls\_bran\_dest-ra\_only

iv\_update = abap\_true

\* iv\_commit = abap\_true

IMPORTING

es\_defhdr = ls\_defhdr

et\_defitm = lt\_defitm

EXCEPTIONS

doc\_status\_finished = 1

doc\_not\_found = 2

OTHERS = 3.

IF sy-subrc = 0.

\* Check if NF IS completed

CALL METHOD me->check\_completed\_nf

EXPORTING

iv\_docnum = ls\_defhdr-docnum

iv\_uf = ls\_defhdr-uf

is\_mem\_defhdr = ls\_defhdr

it\_mem\_defitm = lt\_defitm

EXCEPTIONS

nfe\_not\_completed = 1

dynamic\_sql\_failed = 2

OTHERS = 3.

IF sy-subrc = 0.

\* Set Header status to Complete

ls\_defhdr-status = '2'."Complete

\* Send document to Event Mesh.

CALL METHOD me->send\_nf

IMPORTING

ev\_status = lv\_status

CHANGING

cs\_defhdr = ls\_defhdr.

ELSE.

ls\_defhdr-status = '1'."Incomplete

ENDIF.

\* Update database header

MODIFY /bay0/o2c\_defhdr FROM ls\_defhdr.

\* Commit database changes.

COMMIT WORK AND WAIT.

\* Append headers to output table

APPEND ls\_defhdr TO et\_defhdr.

\* Append lines to Output table.

APPEND LINES OF lt\_defitm TO et\_defitm.

ENDIF."sy-subrc lo\_def->def\_nf\_feed

ENDLOOP."AT lt\_bran\_dest

ENDMETHOD.

METHOD def\_nf\_feed.

\*&---------------------------------------------------------------------\*

\*& Method DEF\_NF\_FEED

\*----------------------------------------------------------------------\*

\* Identification

\* Author : Denis Pereira - EURQK

\* Creation date : 26.11.2024

\* Owner : Daniel Golin

\* Basis Release : 755

\*-----------------------------------------------------------------------

\* Description : Fill Defensve data records.

\*----------------------------------------------------------------------\*

\* Changes \*

\* Vers. Date Author Request Description \*

\* V001 EURQK S1DK932604 created \*

\*----------------------------------------------------------------------\*

\* Local Types.

TYPES: BEGIN OF ts\_dest\_layout,

uf TYPE /bay0/o2c\_defhdr-uf,

nftype TYPE /bay0/o2c\_defhdr-nftype,

uf\_type TYPE /bay0/o2c\_defhdr-uf\_type,

layout TYPE /bay0/o2c\_defhdr-layout,

END OF ts\_dest\_layout,

BEGIN OF ts\_uf\_cg1\_cg,

uf TYPE /bay0/o2c\_defhdr-uf,

kvgr1 TYPE vbrp-kvgr1,

kdgrp TYPE vbrk-kdgrp,

END OF ts\_uf\_cg1\_cg.

\* Local internal tables.

DATA: lt\_itm TYPE TABLE OF /bay0/o2c\_defitm,

lt\_const TYPE /bay0/tda\_tt\_devkey,

lt\_const\_aux TYPE /bay0/tda\_tt\_devkey,

lt\_num TYPE STANDARD TABLE OF bapi1003\_alloc\_values\_num,

lt\_char TYPE STANDARD TABLE OF bapi1003\_alloc\_values\_char,

lt\_curr TYPE STANDARD TABLE OF bapi1003\_alloc\_values\_curr,

lt\_return TYPE STANDARD TABLE OF bapiret2,

lt\_lines TYPE TABLE OF tline,

lt\_dest\_layout TYPE TABLE OF ts\_dest\_layout,

lt\_uf\_cg1\_cg TYPE TABLE OF ts\_uf\_cg1\_cg.

\* Local structures.

DATA: ls\_hdr TYPE /bay0/o2c\_defhdr,

ls\_itm TYPE /bay0/o2c\_defitm,

ls\_dest\_layout TYPE ts\_dest\_layout,

ls\_uf\_cg1\_cg TYPE ts\_uf\_cg1\_cg.

\* Local Variables.

DATA: lv\_object TYPE bapi1003\_key-object,

lv\_langu TYPE thead-tdspras,

lv\_name TYPE thead-tdname,

lv\_valid TYPE flag,

lv\_index TYPE sy-tabix.

\* Get constants enteries from TDA\_DEVKEY table for the regio

CALL FUNCTION '/BAY0/TDA\_GET\_PARAM\_VALUE'

EXPORTING

iv\_devkey = '/BAY0/O2C\_ICBR\_DEF'

iv\_param = 'UF\_NF\_DEST\_LAYOUT%'

IMPORTING

et\_devkey = lt\_const\_aux.

\* Append data to the main table

APPEND LINES OF lt\_const\_aux TO lt\_const.

CALL FUNCTION '/BAY0/TDA\_GET\_PARAM\_VALUE'

EXPORTING

iv\_devkey = '/BAY0/O2C\_ICBR\_DEF'

iv\_param = 'UF\_CG1\_CG%'

IMPORTING

et\_devkey = lt\_const\_aux.

\* Append data to the main table

APPEND LINES OF lt\_const\_aux TO lt\_const.

\* Process constant data.

LOOP AT lt\_const INTO DATA(ls\_const).

\* Clear working structures.

CLEAR: ls\_dest\_layout,

ls\_uf\_cg1\_cg.

IF ls\_const-param(9) = 'UF\_CG1\_CG'.

SPLIT ls\_const-value AT '\_' INTO ls\_uf\_cg1\_cg-uf

ls\_uf\_cg1\_cg-kvgr1

ls\_uf\_cg1\_cg-kdgrp.

\* Append control table.

APPEND ls\_uf\_cg1\_cg TO lt\_uf\_cg1\_cg.

ELSEIF ls\_const-param(17) = 'UF\_NF\_DEST\_LAYOUT'.

SPLIT ls\_const-value AT '\_' INTO ls\_dest\_layout-uf

ls\_dest\_layout-nftype

ls\_dest\_layout-uf\_type

ls\_dest\_layout-layout.

\* Append Destination layout

APPEND ls\_dest\_layout TO lt\_dest\_layout.

ENDIF."ls\_const-param(9) = 'UF\_CG1\_CG'

ENDLOOP."lt\_const

\* Sort control table.

SORT lt\_uf\_cg1\_cg BY uf kvgr1 kdgrp.

\* Sor layout table.

SORT lt\_dest\_layout BY uf nftype uf\_type.

\* Check if Document already exist with finished status.

SELECT SINGLE status

FROM /bay0/o2c\_defhdr

WHERE docnum = @iv\_docnum

AND uf = @iv\_uf

INTO @DATA(lv\_status).

IF lv\_status = '5'."Finished

\* Document already exists with finished status.

RAISE doc\_status\_finished.

ENDIF.

\* Select Nota Fiscal line items

SELECT \*

FROM j\_1bnflin

WHERE docnum = @iv\_docnum

INTO TABLE @DATA(lt\_lin).

IF sy-subrc = 0.

\* Sort NF item by key.

SORT lt\_lin BY docnum itmnum.

\* Read first item available

READ TABLE lt\_lin INTO DATA(ls\_lin\_first) INDEX 1.

\* If reference type is Billing

IF ls\_lin\_first-reftyp = 'BI'.

\* Select invoice data

SELECT a~vbeln, a~posnr, a~uecha, a~vgbel, a~vgpos, a~charg,

a~kvgr1, a~kvgr4, a~mvgr1, a~mvgr3, a~mvgr4, a~konda\_auft,

b~kdgrp

FROM vbrp AS a

INNER JOIN vbrk AS b ON a~vbeln = b~vbeln

FOR ALL ENTRIES IN @lt\_lin

WHERE a~vbeln = @lt\_lin-refkey(10)

AND a~posnr = @lt\_lin-refitm

INTO TABLE @DATA(lt\_vbrp).

IF sy-subrc = 0.

\* Read invoice data from first NF item available.

READ TABLE lt\_vbrp INTO DATA(ls\_first\_vbrp)

WITH KEY vbeln = ls\_lin\_first-refkey(10)

posnr = ls\_lin\_first-refitm.

\* Select Culture Code UF

SELECT mvgr3, culcd

FROM /bay0/o2c\_defcul

FOR ALL ENTRIES IN @lt\_vbrp

WHERE uf = @iv\_uf

AND mvgr3 = @lt\_vbrp-mvgr3

INTO TABLE @DATA(lt\_cul).

IF sy-subrc = 0.

\* Sort by Material group 3

SORT lt\_cul BY mvgr3.

ENDIF.

\* Select Plague Master Data

SELECT mvgr4, plgcd

FROM /bay0/o2c\_defplg

FOR ALL ENTRIES IN @lt\_vbrp

WHERE uf = @iv\_uf

AND mvgr4 = @lt\_vbrp-mvgr4

INTO TABLE @DATA(lt\_plg).

IF sy-subrc = 0.

\* Sort by Material group 4

SORT lt\_plg BY mvgr4.

ENDIF.

ENDIF."sy-subrc SELECT vbrp

ENDIF."ls\_lin-reftyp = 'BI'

\* Remove NF items which are not agrochemicals according

\* to material characteristic value.

LOOP AT lt\_lin INTO DATA(ls\_lin\_check).

lv\_index = sy-tabix.

\* Clear working variables.

CLEAR: lv\_valid.

\* Fill compatible object key variable with material number.

lv\_object = ls\_lin\_check-matnr.

\* Read material characteristic.

CALL FUNCTION 'BAPI\_OBJCL\_GETDETAIL'

EXPORTING

objectkey = lv\_object

objecttable = 'MARA'

classnum = 'YCS\_CC'

classtype = 'YCS'

keydate = sy-datum

unvaluated\_chars = ' '

language = sy-langu

\* IMPORTING

\* STATUS =

\* STANDARDCLASS =

TABLES

allocvaluesnum = lt\_num

allocvalueschar = lt\_char

allocvaluescurr = lt\_curr

return = lt\_return.

IF lt\_char[] IS NOT INITIAL.

LOOP AT lt\_char INTO DATA(ls\_char)

WHERE charact = 'YCS\_CC\_CRQ\_BR\_AGRO\_CHEM\_PROD'. "#EC CI\_NESTED

\* Set all characters to Upper case.

TRANSLATE ls\_char-value\_char TO UPPER CASE.

\* If characteristic content is equal to (Yes) "Agrochemical"

IF ls\_char-value\_char = 'YES'.

\* Set valid flag to true.

lv\_valid = abap\_true.

ENDIF.

\* If it's a valid register.

IF lv\_valid = abap\_true.

\* Read invoice data from first NF item available.

READ TABLE lt\_vbrp INTO DATA(ls\_vbrp\_aux)

WITH KEY vbeln = ls\_lin\_check-refkey(10)

posnr = ls\_lin\_check-refitm.

IF sy-subrc = 0.

\* Read UF CG from table control constants

READ TABLE lt\_uf\_cg1\_cg INTO ls\_uf\_cg1\_cg

WITH KEY uf = iv\_uf

kvgr1 = ls\_vbrp\_aux-kvgr1

kdgrp = ls\_vbrp\_aux-kdgrp.

IF sy-subrc = 0.

\* NF has Prescription

ls\_hdr-preflg = 'X'.

ELSE.

\* If only registers with prescription is considered.

IF iv\_ra\_only = abap\_true.

\* Remove valid flag to delete current item.

CLEAR lv\_valid.

ENDIF.

ENDIF."subrc READ TABLE lt\_uf\_cg1\_cg

ENDIF."sy-subrc READ TABLE lt\_vbrp

ENDIF." lv\_valid = abap\_true.

ENDLOOP."lt\_char INTO DATA

ENDIF."lt\_char[] IS NOT INITIAL

\* If valid flag is empty

IF lv\_valid IS INITIAL.

\* Remove NF item from defensives control.

DELETE lt\_lin INDEX lv\_index.

ENDIF.

ENDLOOP."AT lt\_lin

ELSE.

\* Document Not Found.

RAISE doc\_not\_found.

ENDIF."sy-subrc SLECT FROM j\_1bnflin

\* If NF items are not empty.

IF NOT lt\_lin[] IS INITIAL.

\* Select Nota Fiscal Header

SELECT SINGLE \*

FROM j\_1bnfdoc

WHERE docnum = @iv\_docnum

INTO @DATA(ls\_doc).

\* Select Property Codes

SELECT SINGLE propcd

FROM /bay0/o2c\_defprp

WHERE uf = @iv\_uf

AND kunnr = @ls\_doc-parid

INTO @DATA(lv\_propcd).

\* Select Agronomist CPF

SELECT SINGLE agrcpf

FROM /bay0/o2c\_defagr

WHERE uf = @iv\_uf

AND defau = @abap\_true

INTO @DATA(lv\_agrcpf).

\* Select Application Code

SELECT SINGLE aplcd

FROM /bay0/o2c\_defapp

WHERE uf = @iv\_uf

AND defau = @abap\_true

INTO @DATA(lv\_aplcd).

\* Select Business place

SELECT SINGLE a~bukrs, a~branch, a~adrnr, b~region

FROM j\_1bbranch AS a

LEFT OUTER JOIN adrc AS b

ON a~adrnr = b~addrnumber

AND b~date\_from <= @sy-datum

AND b~date\_to >= @sy-datum

WHERE a~bukrs = @ls\_doc-bukrs

AND a~branch = @ls\_doc-branch

INTO @DATA(ls\_branch).

\* Select Unit of Measure Area

SELECT SINGLE arecd

FROM /bay0/o2c\_defaum

WHERE uf = @iv\_uf

AND defau = @abap\_true

INTO @DATA(lv\_arecd).

\* Select Defensives Control - UF Material Conversion.

SELECT matnr, matuf

FROM /bay0/o2c\_defmat

FOR ALL ENTRIES IN @lt\_lin

WHERE uf = @iv\_uf

AND matnr = @lt\_lin-matnr

INTO TABLE @DATA(lt\_mat).

IF sy-subrc = 0.

\* Sort by material number

SORT lt\_mat BY matnr.

ENDIF.

\* Move all corresponding header fields.

MOVE-CORRESPONDING ls\_doc TO ls\_hdr.

\* Regulatory State

ls\_hdr-uf = iv\_uf.

\* UF Type (B - "Branch - (Origin)" / D - "Destination")

ls\_hdr-uf\_type = iv\_uf\_type.

\* Property Code

ls\_hdr-propcd = lv\_propcd.

\* Agronomist CPF.

ls\_hdr-agrcpf = lv\_agrcpf.

\* Branch Region

ls\_hdr-brauf = ls\_branch-region.

\* Get layout from constant values

CLEAR ls\_dest\_layout.

READ TABLE lt\_dest\_layout INTO ls\_dest\_layout

WITH KEY uf = ls\_hdr-uf

nftype = ls\_hdr-nftype

uf\_type = ls\_hdr-uf\_type.

IF sy-subrc = 0.

\* Fill NF Layout

ls\_hdr-layout = ls\_dest\_layout-layout.

ENDIF.

\* If NF does not have Prescription

IF ls\_hdr-preflg IS INITIAL.

\* Set Resales flag to true.

ls\_hdr-resales = abap\_true.

ENDIF.

\* Status. Set value 0 – Initial

ls\_hdr-status = 0.

\* Read Prescription Number

\* Portuguese

lv\_langu = 'P'.

\* Document number of the reference document

lv\_name = ls\_first\_vbrp-vgbel.

CALL FUNCTION 'READ\_TEXT'

EXPORTING

client = sy-mandt

id = 'ZH01'

language = lv\_langu

name = lv\_name

object = 'VBBK'

TABLES

lines = lt\_lines

EXCEPTIONS

id = 1

language = 2

name = 3

not\_found = 4

object = 5

reference\_check = 6

wrong\_access\_to\_archive = 7

OTHERS = 8.

IF sy-subrc <> 0.

\* Try English

lv\_langu = 'E'.

CALL FUNCTION 'READ\_TEXT'

EXPORTING

client = sy-mandt

id = 'ZH01'

language = lv\_langu

name = lv\_name

object = 'VBBK'

TABLES

lines = lt\_lines

EXCEPTIONS

id = 1

language = 2

name = 3

not\_found = 4

object = 5

reference\_check = 6

wrong\_access\_to\_archive = 7

OTHERS = 8.

ENDIF.

IF sy-subrc EQ 0 AND lt\_lines[] IS NOT INITIAL.

\* Read the first text line available.

READ TABLE lt\_lines INTO DATA(ls\_lines) INDEX 1.

IF sy-subrc EQ 0.

\* Fill Prescription Number

ls\_hdr-prescr = ls\_lines-tdline.

ENDIF.

ENDIF."sy-subrc EQ 0 AND lt\_lines[]

\* Process NF Items

LOOP AT lt\_lin INTO DATA(ls\_lin).

\* Clear working structures.

CLEAR: ls\_itm.

\* Regulatory State

ls\_itm-uf = iv\_uf.

\* Document Number

ls\_itm-docnum = ls\_lin-docnum.

\* Document Item Number

ls\_itm-itmnum = ls\_lin-itmnum.

\* Material Number

ls\_itm-matnr = ls\_lin-matnr.

\* Read UF Material Conversion.

READ TABLE lt\_mat INTO DATA(ls\_mat)

WITH KEY matnr = ls\_lin-matnr

BINARY SEARCH.

IF sy-subrc = 0.

\* UF Material Number

ls\_itm-matuf = ls\_mat-matuf.

ENDIF.

\* Batch Number

ls\_itm-charg = ls\_lin-charg.

\* Quantity

ls\_itm-menge = ls\_lin-menge.

\* Base Unit of Measure

ls\_itm-meins = ls\_lin-meins.

\* Read invoice

READ TABLE lt\_vbrp INTO DATA(ls\_vbrp) WITH KEY vbeln = ls\_lin-refkey(10)

posnr = ls\_lin-refitm.

IF sy-subrc <> 0.

CLEAR ls\_vbrp.

ENDIF.

\* Application Code

ls\_itm-aplcd = lv\_aplcd.

\* Area UF Code

ls\_itm-arecd = lv\_arecd.

\* Read Pesticide Controls - Culture

READ TABLE lt\_cul INTO DATA(ls\_cul) WITH KEY mvgr3 = ls\_vbrp-mvgr3

BINARY SEARCH.

IF sy-subrc = 0.

\* Culture Code UF

ls\_itm-culcd = ls\_cul-culcd.

ENDIF.

\* Read Plague Master Data

READ TABLE lt\_plg INTO DATA(ls\_plg) WITH KEY mvgr4 = ls\_vbrp-mvgr4

BINARY SEARCH.

IF sy-subrc = 0.

\* Plague Code

ls\_itm-plgcd = ls\_plg-plgcd.

ENDIF.

\* Append NF item to table parmeter.

APPEND ls\_itm TO et\_defitm.

ENDLOOP."AT lt\_lin INTO DATA(ls\_lin)

\* Fill header structure parameter.

es\_defhdr = ls\_hdr.

\* If database update is true.

IF iv\_update = abap\_true.

\* Update Selected Nota Fiscal – Header

MODIFY /bay0/o2c\_defhdr FROM es\_defhdr.

\* Update Selected Nota Fiscal – item

MODIFY /bay0/o2c\_defitm FROM TABLE et\_defitm.

IF iv\_log\_upd\_md IS INITIAL. "Creation

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = es\_defhdr-docnum

iv\_uf = es\_defhdr-uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = es\_defhdr-status

iv\_save\_file = ''

is\_defhdr = es\_defhdr

it\_defitm = et\_defitm

it\_balmsg = VALUE #( ( msgty = 'S' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 002 msgv1 = es\_defhdr-docnum msgv2 = es\_defhdr-uf msgv3 = es\_defhdr-status

probclass = '2' ) ) ). "2=Important

ELSE."Update Only

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = es\_defhdr-docnum

iv\_uf = es\_defhdr-uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = es\_defhdr-status

iv\_save\_file = ''

is\_defhdr = es\_defhdr

it\_defitm = et\_defitm

it\_balmsg = VALUE #( ( msgty = 'S' msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = 011 msgv1 = es\_defhdr-docnum msgv2 = es\_defhdr-uf

probclass = '2' ) ) ). "2=Important

ENDIF.

\* If commit is internal

IF iv\_commit = abap\_true.

\* Execute BAPI commit and wait.

CALL FUNCTION 'BAPI\_TRANSACTION\_COMMIT'

EXPORTING

wait = abap\_true.

ENDIF."iv\_commit = abap\_true

ENDIF."iv\_update = abap\_true

ELSE.

\* Clear header data.

CLEAR es\_defhdr.

\* Document Not Found.

RAISE doc\_not\_found.

ENDIF."NOT lt\_lin[] IS INITIAL

ENDMETHOD.

METHOD save\_file.

DATA: ls\_nf\_file TYPE /bay0/o2c\_defjso.

DATA: lt\_json TYPE /bay0/o2c\_def\_json\_tt,

ls\_json TYPE LINE OF /bay0/o2c\_def\_json\_tt,

ls\_json\_item TYPE LINE OF /bay0/o2c\_defitm\_tt.

IF is\_defhdr IS NOT INITIAL AND it\_defitm IS NOT INITIAL.

MOVE-CORRESPONDING is\_defhdr TO ls\_json-nfheader.

LOOP AT it\_defitm ASSIGNING FIELD-SYMBOL(<fs\_defitm>) WHERE docnum = is\_defhdr-docnum

AND uf = is\_defhdr-uf.

MOVE-CORRESPONDING <fs\_defitm> TO ls\_json\_item.

APPEND ls\_json\_item TO ls\_json-nfitems.

CLEAR ls\_json\_item.

ENDLOOP.

APPEND ls\_json TO lt\_json.

CLEAR ls\_json.

ELSEIF iv\_docnum IS NOT INITIAL.

SELECT \*

FROM /bay0/o2c\_defhdr

INTO TABLE @DATA(lt\_def\_hdr)

WHERE docnum = @iv\_docnum.

IF sy-subrc = 0.

SORT lt\_def\_hdr BY docnum.

SELECT \*

FROM /bay0/o2c\_defitm

INTO TABLE @DATA(lt\_def\_itm)

FOR ALL ENTRIES IN @lt\_def\_hdr

WHERE docnum = @lt\_def\_hdr-docnum

AND uf = @lt\_def\_hdr-uf.

IF sy-subrc = 0.

SORT lt\_def\_itm BY docnum uf.

ENDIF.

LOOP AT lt\_def\_hdr ASSIGNING FIELD-SYMBOL(<fs\_def\_hdr>).

MOVE-CORRESPONDING <fs\_def\_hdr> TO ls\_json-nfheader.

LOOP AT lt\_def\_itm ASSIGNING FIELD-SYMBOL(<fs\_def\_itm>) WHERE docnum = <fs\_def\_hdr>-docnum

AND uf = <fs\_def\_hdr>-uf.

MOVE-CORRESPONDING <fs\_def\_itm> TO ls\_json\_item.

APPEND ls\_json\_item TO ls\_json-nfitems.

CLEAR ls\_json\_item.

ENDLOOP.

APPEND ls\_json TO lt\_json.

CLEAR ls\_json.

ENDLOOP.

ENDIF.

ENDIF.

IF lt\_json IS NOT INITIAL.

DATA(lv\_json\_case) = /ui2/cl\_json=>serialize(

data = lt\_json

compress = abap\_false

pretty\_name = /ui2/cl\_json=>pretty\_mode-low\_case ).

DATA(lv\_json\_raw) = /ui2/cl\_json=>string\_to\_raw( iv\_string = lv\_json\_case ).

ls\_nf\_file-uf = VALUE #( lt\_json[ 1 ]-nfheader-uf OPTIONAL ).

ls\_nf\_file-docnum = VALUE #( lt\_json[ 1 ]-nfheader-docnum OPTIONAL ).

ls\_nf\_file-credat = sy-datum.

ls\_nf\_file-cretim = sy-uzeit.

ls\_nf\_file-crenam = sy-uname.

ls\_nf\_file-file\_format = 'JSON'.

ls\_nf\_file-filex = lv\_json\_case.

ls\_nf\_file-file\_raw = lv\_json\_raw.

MODIFY /bay0/o2c\_defjso FROM ls\_nf\_file.

IF sy-subrc = 0.

COMMIT WORK AND WAIT.

ENDIF.

ELSE.

"Throw Log Warning

ENDIF.

ENDMETHOD.

METHOD send\_nf.

\*&---------------------------------------------------------------------\*

\*& Method SEND\_NF

\*----------------------------------------------------------------------\*

\* Identification

\* Author : Denis Pereira - EURQK

\* Creation date : 26.11.2024

\* Owner : Daniel Golin

\* Basis Release : 755

\*-----------------------------------------------------------------------

\* Description : Trigger defensive event mesh to CPI

\*----------------------------------------------------------------------\*

\* Changes \*

\* Vers. Date Author Request Description \*

\* V001 EURQK S1DK932604 created \*

\*----------------------------------------------------------------------\*

\* Local Types

TYPES: BEGIN OF ts\_event\_mesh,

docnum TYPE /bay0/o2c\_defhdr-docnum,

uf TYPE /bay0/o2c\_defhdr-uf,

END OF ts\_event\_mesh.

\* Local structures

DATA: ls\_event\_mesh TYPE ts\_event\_mesh.

\* Local Variables.

DATA: lv\_docnum\_char TYPE char10,

lv\_json TYPE string,

lv\_msgno TYPE symsgno,

lv\_msgty TYPE symsgty.

\* Remove leading zeros from document number.

CALL FUNCTION 'CONVERSION\_EXIT\_ALPHA\_OUTPUT'

EXPORTING

input = cs\_defhdr-docnum

IMPORTING

output = lv\_docnum\_char.

\* Fill sructure o be converted to JSON

ls\_event\_mesh-docnum = cs\_defhdr-docnum.

ls\_event\_mesh-uf = cs\_defhdr-uf.

\* Convert structure to JSON format.

lv\_json = /ui2/cl\_json=>serialize( ls\_event\_mesh ).

\* Cal event Mesh function

CALL FUNCTION '/BAY0/TDA\_EVENT\_MESH'

EXPORTING

iv\_type = 'DEFENSE'

iv\_json = lv\_json

iv\_objkey = 'RMRB'

iv\_object = '001'

iv\_value = '1'

IMPORTING

ev\_status = ev\_status.

\* If event mesh status is 'S' - "Success."

IF ev\_status = 'S'.

cs\_defhdr-status = '7'."Sent

lv\_msgno = '005'."Docnum: &1 UF: &2 successfully sent.

lv\_msgty = 'S'.

ELSE.

cs\_defhdr-status = '3'."Error

lv\_msgno = '008'."Docnum: &1 UF: &2 sending failed.

lv\_msgty = 'E'.

ENDIF.

\* Save log

me->save\_log(

EXPORTING

iv\_docnum = cs\_defhdr-docnum

iv\_uf = cs\_defhdr-uf

iv\_item = '000000'"es\_defhdr-itmnum

iv\_status = '1'

iv\_save\_file = ''

\* is\_defhdr = cs\_defhdr

\* it\_defitm = lt\_defitm

it\_balmsg = VALUE #( ( msgty = lv\_msgty msgid = /bay0/o2c\_cl\_def=>gc\_log\_msg\_class

msgno = lv\_msgno

msgv1 = lv\_docnum\_char

msgv2 = cs\_defhdr-uf

\* msgv3 =

probclass = '2' ) ) ). "2=Important

ENDMETHOD.

ENDCLASS.