SDN Community Contribution

(This is not an official SAP document.)

Disclaimer & Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.

Applies To

Interfaces from Legacy System to SAP APO.

Summary

The use of Flat File Interfaces in the Client environment

To implement a flat file interface, the following client standards are applicable:

- each interface needs to have an interface identification
- data of the interface is stored in a customer table (interface catalogue)
- the directory location is stored in a logical file, the logical file name is stored in the customer table
- when running the interface, we record the start and end date and time, status in a history table
- there are standards for file names, and these are generated automatically by the interface start function
- the middleware system does the file transfer based upon the filename
- the security/authorization can be determined on the logical file path

Company: IBM India

Date: 04 October 2005

Table of Contents

The use of Flat File Interfaces in the Client environment

To implement a flat file interface, the following client standards are applicable:

- each interface needs to have an interface identification
- data of the interface is stored in a customer table (interface catalogue)
- the directory location is stored in a logical file, the logical file name is stored in the customer table
- when running the interface, we record the start and end date and time, status in a history table
- there are standards for file names, and these are generated automatically by the interface start function
- the middleware system does the file transfer based upon the filename
- the security/authorization can be determined on the logical file path

1.1 Interface identification

An interface key will uniquely identify each batch interface. The key consists of 3 parameters and logically describes the *source*, *destination* and *content* of any interface file.

- Organizational Unit: This represents the logical business unit within Client that is responsible for the interface. This could be the CLIENT center, a region, a market or a subsidiary within a market.

Example :	0000	Client Center – Globally managed interface
	CH11	Client Suisse SA
	GB11	Client UK

  For outbound interfaces the organizational unit indicates the origin of the data. For inbound interfaces the organizational unit identifies the destination of the data.

  For each organizational unit a separate batch schedule will be set up with the holiday and factory calendar specific to that region/market/subsidiary. For this reason we will make use of company codes to describe the organizational unit. In case the organizational unit is not a legal client entity (e.g. the AMS region as a whole) we will make use of dummy company codes and maintain these in a separate table.

- External system / partner : This represents the name of the external partner or legacy system that receives/sends the interface file. This can correspond to an actual legacy system or can be the name of an EDI partner.

The external system / partner needs to be stored in customer table /EURGLB/XEEXTPAR: CLIENT External Partner Master (see also appendix).

- Interface Identifier : This describes the type of message being sent to/from the external partner. It is effectively a unique name for a CLIENT standard plug-in.

Example :	CSORD01	EDI order interface
	CSINV01	EDI invoice interface
FICAR01	CARAT Common interface – POPL message	

The interface identifier describes *what kind of data* is in the file.

The interface identifier needs to be stored in customer table /EURGLB/XEINTMAS: CLIENT Interface ID Master.

Naming convention for this field is now PPDMMMMMNN

Where:

- PP - Process area e.g. 'FI'
- D - Direction (I - Inbound; O - Outbound)
- MMMMM - Message content (ORD - Orders, PRDPL - Product P&L, etc). Essentially 5 bytes to describe the message.
- NN - Sequence number. This is used to uniquely identify interface programs with similar content. Example - Both R/3 core and CRM have orders inbound interfaces.

For each interface the 3 parameters above answer the following questions :

- Who is the data from? (organizational unit for outbound / External system for inbound)
- Who is the data for? (organizational unit for inbound / External system for outbound)
- What data is in the file? (interface identifier)

These three identifiers will be used to save the specific information about the interface, to save the run history and it is also used in the filename, and consequently in the middleware system as routing mechanism.

When a new interface is created, you first have to check that there is not an existing interface. Therefore you need to contact *interface coordinator*. This person will check the interface catalogue and check for similar interfaces and he will give the name of the external system and the interface identifier to be used for the new interface.

1.2 CLIENT Interface Identification table

All interfaces are stored in this CLIENT Interface Identification table: see table /EURGLB/XEINTID in section 2.

For maintenance, see transaction SM30 for this table.

The interface coordinator is owner of this table but each development coordinator is for having the right entries in this table for his own interfaces.

1.3 Logical file name

The standards of the logical file name are described in a document of the basis team (see also appendix). See above: shortcut to document: Standards-SIDLogicalSystemsClients.

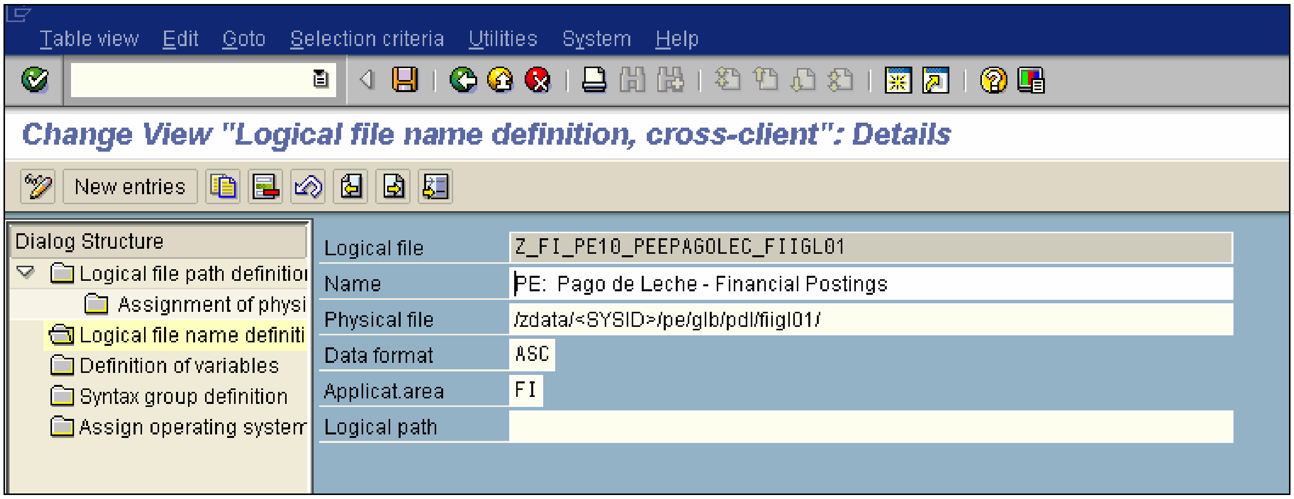
Use transaction FILE to create the logical file.

The following set-up is used in CLIENT:

-> no logical path definition is created

-> all information is stored in the logical file name definition

See picture below.



Change View "Logical file name definition, cross-client": Details

Logical file	Z_FI_PE10_PEEPAGOLEC_FIIGL01
Name	PE: Pago de Leche - Financial Postings
Physical file	/zdata/<SYSID>/pe/glb/pdl/fiigl01/
Data format	ASC
Applicat.area	FI
Logical path	

1.4 Physical file path - name

The standards for the physical file path are stored in the following document: (See also appendix).

See above: shortcut to document: Standards-SIDLogicalSystemsClients.

The concept of the interface identification key is maintained also in the file name. This will reduce complexity within SAP and will give the adapter a key with which to determine routing information for the middleware.

The file name consists of:

```
[ORGANIZATIONAL UNIT] .
        [EXTERNAL SYSTEM] .
                [INTERFACE IDENTIFIER] .
                        [SEQUENCE NUMBER] . DAT
```

For example an interface with

- Organizational unit = Client Suisse (CH11)
- External system = CHEINFO+
- Interface identifier = FIOTRPRC01
- with file number 176

will have as file name : CH11. CHEINFO+. FIOTRPRC01.000000000176.DAT

The middleware system normally uses only the organizational unit, external system and interface identifier to identify the correct route for the file.

1.5 Programming standards for file handling

When the file starts, we have to call function /EURGLB/X_INTERFACE_START_PROC.

When ending the interface, we have to call function /EURGLB/X_INTERFACE_END_PROC.

We have a specific header and footer: see structure

/EURGLB/XINTERFACE_FOOTER and structure /EURGLB/XINTERFACE_HEADER.

1.5.1 /EURGLB/X_INTERFACE_START_PROC

The function has the following importing parameters:

BUKRS	: Organizational unit
INTID	: Interface identifier

PARNUM : Partner number

INPUT_FILENAME : Local file for upload/download

PARALLEL_RUNS_ALLOWED : Parallel runs allowed = 'X', if the flag is set, the last interface run is not checked to not have status 'in process'.

NORMAL_RUN : Run mode normal = 'X', if the flag is not set, the function assumes it is a rerun and it will get validated with the last sequence number and no new entry will be created in the history table

TEST_RUN : Test run = 'X': if set, there is not updating of the history table

NO_SEQ_CHECK : No sequence check = 'X', if set, there is no check on the sequence number for inbound files

NO_HEAD_FOOTER : No header nor footer, if set, no checks are performed on the header or footer, and the header and footer are not removed

Exporting parameters:

XEINTID : CLIENT Interface Identification table

PHYSICAL_FILENAME : Logical file name

HEADER : Interface header record

Table parameters:

INPUT_FILE : EDI Interface file structure

Functionality of the Function Module.

- Based upon the organizational unit, interface identifier and partner number, table /EURGLB/XEINTID is read. This gives the specific interface information.
- All interface information for this interface is selected by the history table /EURGLB/XEINTHIS and sorted by date and time
- If the flag PARALLEL_RUNS_ALLOWED is not set, then we check that the last specific interface run is not in status 'In process'.
- The header records is now filled using the information from /EURGLB/XEINTID, the date and time is set to the current date and time. The sequence number is the current sequence number of the last interface added by 1.

- We get the physical path using function FILE_GET_NAME using the logical file name stored in table /EURGLB/XEINTID. For inbound interfaces, the physical path is concatenated with the file name. For outbound interfaces, the same is done if the filename is not empty. If the filename INPUT_FILENAME is empty, the following filename is built:
  physical path + XEINTID-BUKRS + '.' + XEINTID-PARNUM '.' + XEINTID-INTID + '.' + new sequence number + '.DAT'
- If it is inbound interface, some additional checks are done:
  - The file is opened and completely read into table parameter INPUT_FILE.
  - If the flag NO_HEAD_FOOTER is not set and NORMAL_RUN is set, the header is read and checked: record type, sequence number, partner number. The footer records is also read and checked: record type is checked and also the number of records
  - The header and the footer are removed from the table parameter INPUT_FILE.
- If the TEST_RUN parameter is empty, then the history table /EURGLB/XEINTHIS is updated with the current date and time if the status is not 'in process', otherwise the time stamp is not changed.

During Inbound interface processing the HEADER and FOOTER are provided in the file.

During Outbound interface processing

- The HEADER will be built by function module /EURGLB/X_INTERFACE_START_PROC, but will still need to be written into the file by the interface code

The FOOTER record needs to be built and written into the file by the interface code.

1.5.2 /EURGLB/X_INTERFACE_END_PROC

This function is to be called at the end of the interface processing.

The function has the following importing parameters:

HEADER		: Interface header record
STATUS		: Status
RECCT		: Total records of the file
TRANS		: Number of transactions
ERRCT		: Number of incorrect transactions
TEST_RUN		: Test run
MEMHOLD	TYPE	: Free Text (e.g. last record processed)

There are no export parameters nor table parameters.

The following statuses are valid:

'in process' : 1

'successfully finished' : 2

'Failure': 3

The function updates the specific record using the HEADER record information in the history table /EURGLB/XEINTHIS. If the status is not in process, then the current date and time is set, otherwise the time stamp is not changed.

1.6 HEADER / FOOTER RECORD

1.6.1 HEADER RECORD

Regardless of the content of the file, the first record of any interface flat file should always be the HEADER record. It consists of the following fields :

Field	Description	Type	Size	Valueset / Example
RECTYPE	Record type	CHAR	10	'HEADER'
BUKRS	Organizational Unit	CHAR	4	e.g.: '0937' (Client Suisse)
PARNUM	External partner / system	CHAR	10	e.g.: 'CARAT'
INTID	Interface Identifier	CHAR	12	e.g.: 'CSINV01'
SEQNO	File sequence number (table XEINTHIS keeps log of previous sequence numbers for this interface)	NUMC	12	e.g.: '000000000005'

BEGDA	File creation date	DATS	8	YYYYMMDD
BEGTI	File creation time	TIMS	6	HHMMSS

This record is defined as structure /EURGLB/XINTERFACE_HEADER in the data dictionary.

1.6.2 FOOTER RECORD

Regardless of the content of the file, the last record of any interface flat file should always be the FOOTER record. It consists of the following fields :

Field	Description	Type	Size	Valueset / Example
RECTYPE	Record type	CHAR	10	'FOOTER'
RECORDS	Total number of records in file	NUMC	12	e.g.: '000000000100'
TRANS	Total number of LUWs in file	NUMC	12	e.g.: '000000000025'
CHKSUM	Control checksum value (is based on a control algorithm)	NUMC	25	e.g.: '000000000012586'

This record is defined as structure /EURGLB/XINTERFACE_FOOTER in the data dictionary.

1.7 Error handling

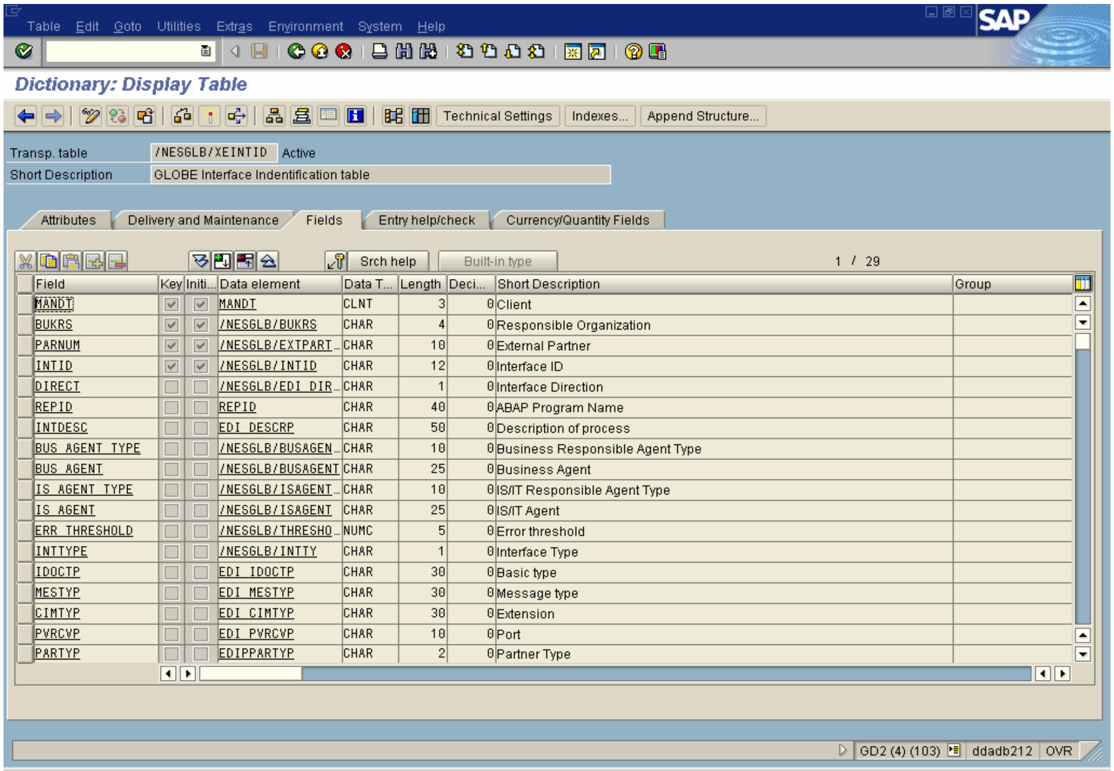
The /EURGLB/X_INTERFACE_START_PROC function and the /EURGLB/X_INTERFACE_END_PROC function return specific exceptions back to the calling program whenever errors for these functions.

The calling program can decide how to handle this exceptions. However, if an entry is already made in the history table, the purpose is to set the status to 3:failure. If there were no errors, the status should be set to 2: success. This change is done when you call the /EURGLB/X_INTERFACE_END_PROC function with the proper import parameters.

If an interface has the status in process and a new interface run is requested with the option PARALLEL_RUNS_ALLOWED = space, then an the interface start function will return an exception. In this situation, the entry in the history table with the status in process has to be checked and if there is not interface running anymore and all reconciliation activities have taken place, the status should be changed in the table.

It is possible to code (within the interface program itself) sending the output report via SAPMail to designated users in the case of error. The fields /EURGLB/XEINTID-BUS_AGENT_TYPE, /EURGLB/XEINTID-BUS_AGENT, /EURGLB/XEINTID-IS_AGENT_TYPE, /EURGLB/XEINTID-IS_AGENT are used to determine where the output should be directed. The EDI Orders interface program /EURGLB/VGTRB_EDI_ORDERS_INT is a good example of this functionality.

2 Screen Shots of Interface tables and parameters of Function modules:



Dictionary: Display Table

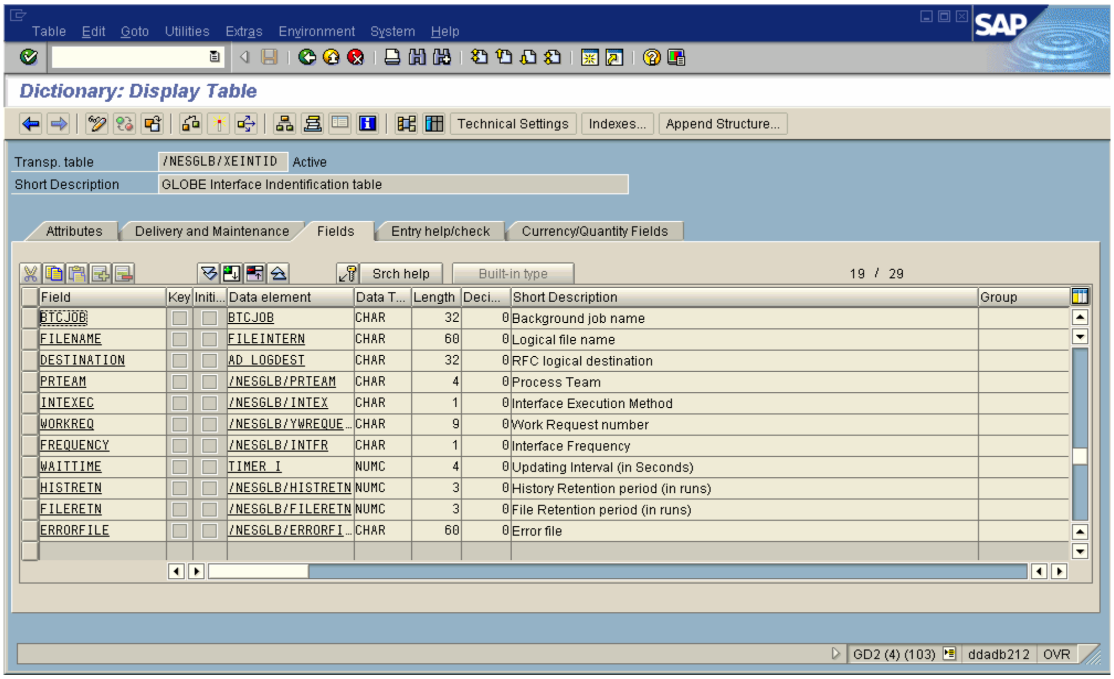
Transp. table /NESGLB/XEINTID Active

Short Description GLOBE Interface Indentification table

1 / 29

Field	Key	Initi...	Data element	Data T...	Length	Deci...	Short Description	Group
MANDT	☑	☑	MANDT	CLNT	3	0	Client	
BUKRS	☑	☑	/NESGLB/BUKRS	CHAR	4	0	Responsible Organization	
PARNUM	☑	☑	/NESGLB/EXTPART...	CHAR	10	0	External Partner	
INTID	☑	☑	/NESGLB/INTID	CHAR	12	0	Interface ID	
DIRECT	□	□	/NESGLB/EDI_DIR...	CHAR	1	0	Interface Direction	
REPID	□	□	REPID	CHAR	40	0	ABAP Program Name	
INTDESC	□	□	EDI_DESCRP	CHAR	50	0	Description of process	
BUS_AGENT_TYPE	□	□	/NESGLB/BUSAGEN...	CHAR	10	0	Business Responsible Agent Type	
BUS_AGENT	□	□	/NESGLB/BUSAGENT	CHAR	25	0	Business Agent	
IS_AGENT_TYPE	□	□	/NESGLB/ISAGENT...	CHAR	10	0	IS/IT Responsible Agent Type	
IS_AGENT	□	□	/NESGLB/ISAGENT	CHAR	25	0	IS/IT Agent	
ERR_THRESHOLD	□	□	/NESGLB/THRESHO...	NUMC	5	0	Error threshold	
INTTYPE	□	□	/NESGLB/INTTY	CHAR	1	0	Interface Type	
IDOCTP	□	□	EDI_IDOCTP	CHAR	30	0	Basic type	
MESTYP	□	□	EDI_MESTYP	CHAR	30	0	Message type	
CIMTYP	□	□	EDI_CIMTYP	CHAR	30	0	Extension	
PVRCVP	□	□	EDI_PVRCVP	CHAR	10	0	Port	
PARTYP	□	□	EDIPPARTYP	CHAR	2	0	Partner Type	

GD2 (4) (103) ddadb212 OVR



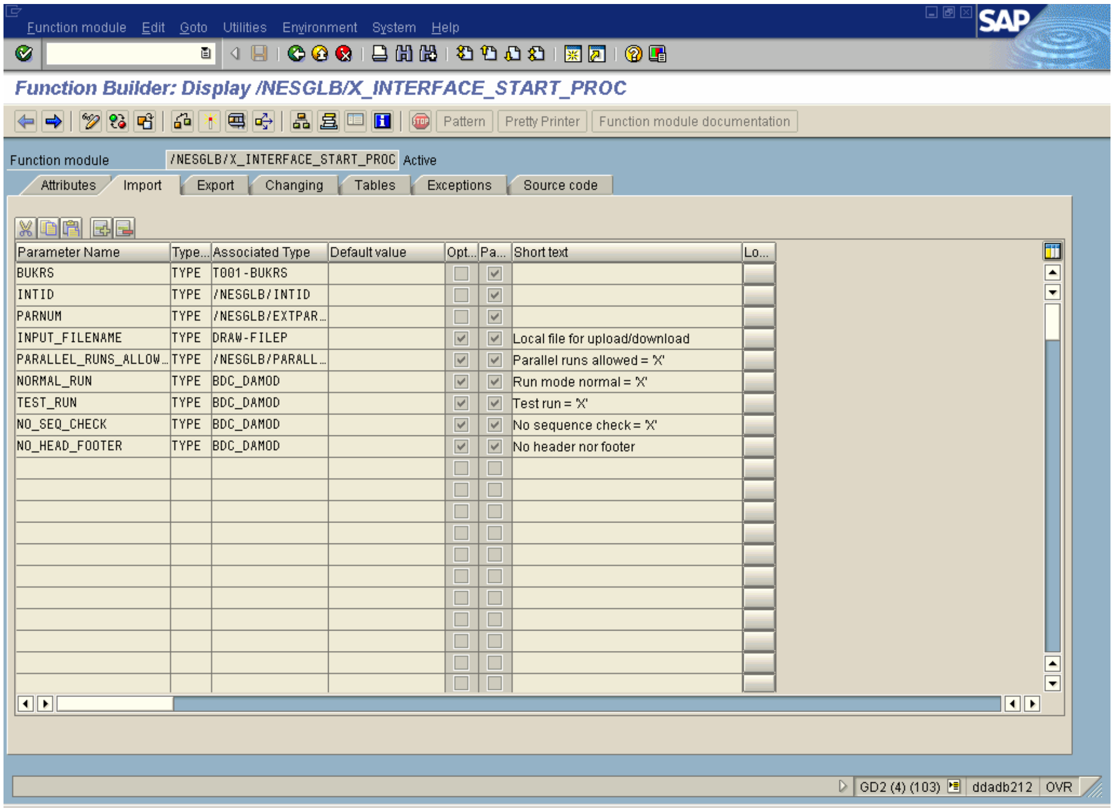
Dictionary: Display Table

Transp. table /NESGLB/XEINTID Active

Short Description GLOBE Interface Indentification table

19 / 29

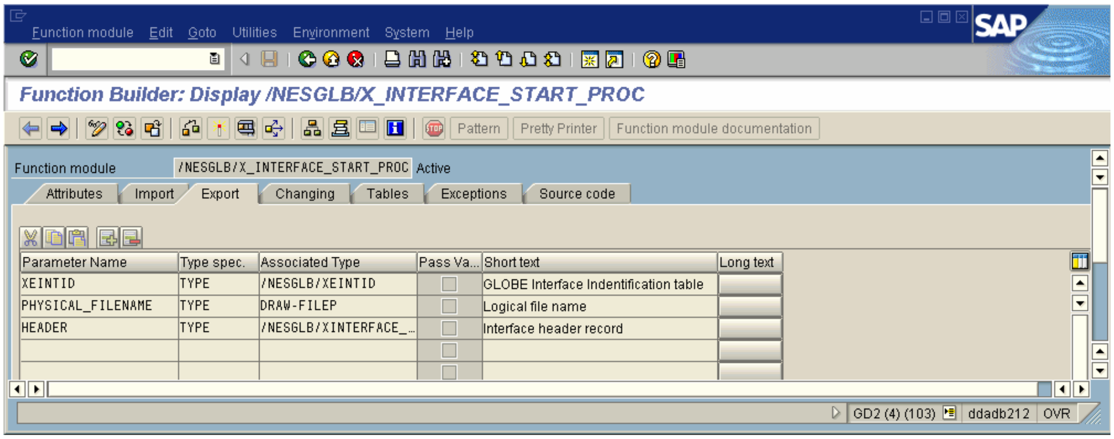
Field	Key	Initi...	Data element	Data T...	Length	Deci...	Short Description	Group
BTCJOB	□	□	BTCJOB	CHAR	32	0	Background job name	
FILENAME	□	□	FILEINTERN	CHAR	60	0	Logical file name	
DESTINATION	□	□	AD_LOGDEST	CHAR	32	0	RFC logical destination	
PRTEAM	□	□	/NESGLB/PRTEAM	CHAR	4	0	Process Team	
INTEXEC	□	□	/NESGLB/INTEX	CHAR	1	0	Interface Execution Method	
WORKREQ	□	□	/NESGLB/YWREQUE...	CHAR	9	0	Work Request number	
FREQUENCY	□	□	/NESGLB/INTFR	CHAR	1	0	Interface Frequency	
WAITTIME	□	□	TIMER_I	NUMC	4	0	Updating Interval (in Seconds)	
HISTRETN	□	□	/NESGLB/HISTRETN	NUMC	3	0	History Retention period (in runs)	
FILERETN	□	□	/NESGLB/FILERETN	NUMC	3	0	File Retention period (in runs)	
ERRORFILE	□	□	/NESGLB/ERRORFI...	CHAR	60	0	Error file	



Function Builder: Display /NESGLB/X_INTERFACE_START_PROC

Function module /NESGLB/X_INTERFACE_START_PROC Active

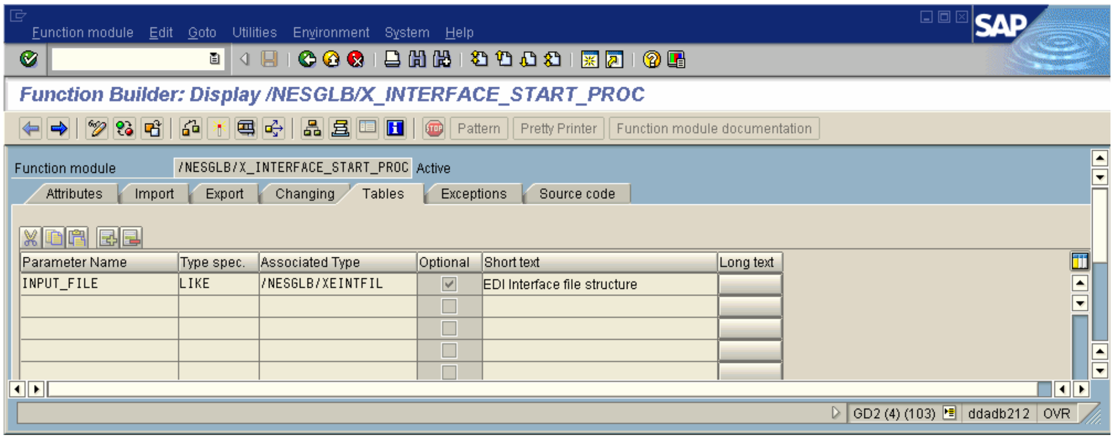
Parameter Name	Type...	Associated Type	Default value	Opt...	Pa...	Short text	Lo...
BUKRS	TYPE	T001-BUKRS		☐	☑		
INTID	TYPE	/NESGLB/INTID		☐	☑		
PARNUM	TYPE	/NESGLB/EXTPAR...		☐	☑		
INPUT_FILENAME	TYPE	DRAW-FILEP		☑	☑	Local file for upload/download	
PARALLEL_RUNS_ALLOW...	TYPE	/NESGLB/PARALL...		☑	☑	Parallel runs allowed = 'X'	
NORMAL_RUN	TYPE	BDC_DAMOD		☑	☑	Run mode normal = 'X'	
TEST_RUN	TYPE	BDC_DAMOD		☑	☑	Test run = 'X'	
NO_SEQ_CHECK	TYPE	BDC_DAMOD		☑	☑	No sequence check = 'X'	
NO_HEAD_FOOTER	TYPE	BDC_DAMOD		☑	☑	No header nor footer	



Function Builder: Display /NESGLB/X_INTERFACE_START_PROC

Function module /NESGLB/X_INTERFACE_START_PROC Active

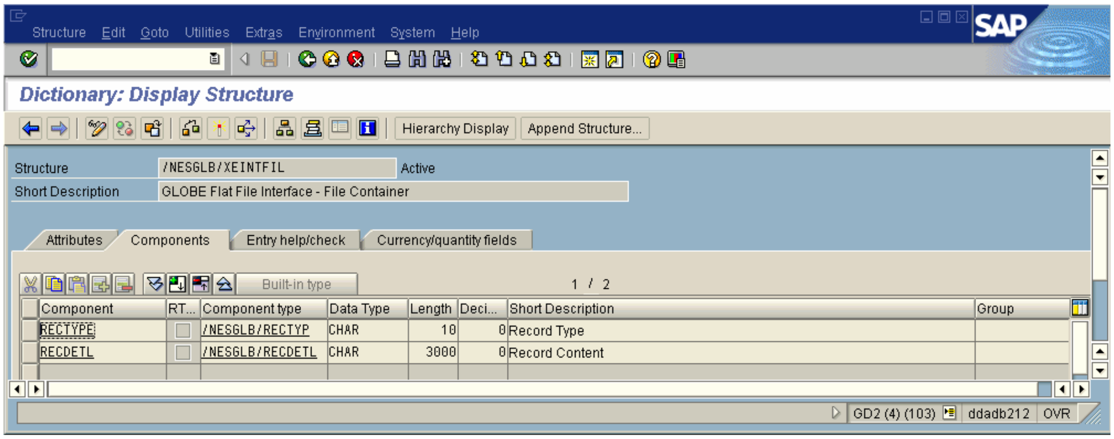
Parameter Name	Type spec.	Associated Type	Pass Va...	Short text	Long text
XEINTID	TYPE	/NESGLB/XEINTID	☐	GLOBE Interface Indentification table	
PHYSICAL_FILENAME	TYPE	DRAW-FILEP	☐	Logical file name	
HEADER	TYPE	/NESGLB/XINTERFACE_...	☐	Interface header record	



Function Builder: Display /NESGLB/X_INTERFACE_START_PROC

Function module /NESGLB/X_INTERFACE_START_PROC Active

Parameter Name	Type spec.	Associated Type	Optional	Short text	Long text
INPUT_FILE	LIKE	/NESGLB/XEINTFIL	☑	EDI Interface file structure	
			☐		
			☐		
			☐		
			☐		



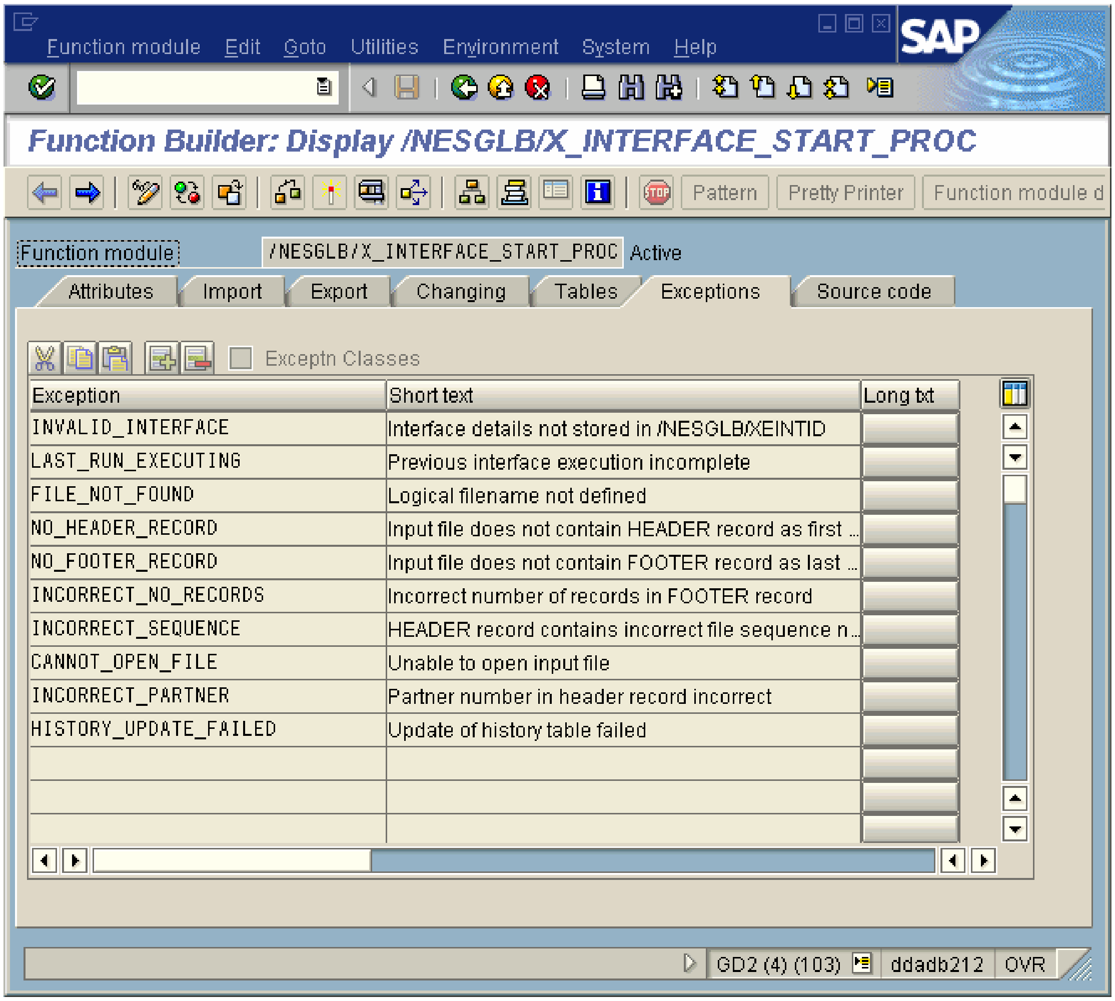
Dictionary: Display Structure

Structure /NESGLB/XEINTFIL Active

Short Description GLOBE Flat File Interface - File Container

1 / 2

Component	RT...	Component type	Data Type	Length	Deci...	Short Description	Group
RECTYPE	☐	/NESGLB/RECTYP	CHAR	10	0	Record Type	
RECDETL	☐	/NESGLB/RECDETL	CHAR	3000	0	Record Content	



Function Builder: Display /NESGLB/X_INTERFACE_START_PROC

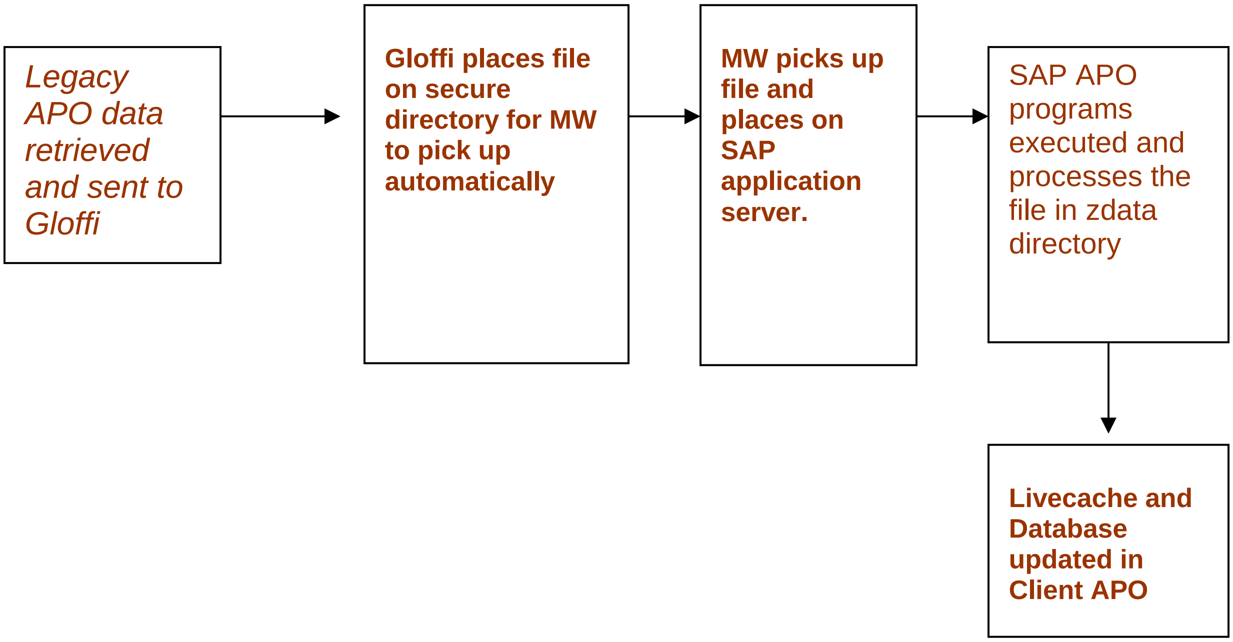
Function module /NESGLB/X_INTERFACE_START_PROC Active

Exception	Short text	Long txt
INVALID_INTERFACE	Interface details not stored in /NESGLB/XEINTID	
LAST_RUN_EXECUTING	Previous interface execution incomplete	
FILE_NOT_FOUND	Logical filename not defined	
NO_HEADER_RECORD	Input file does not contain HEADER record as first ...	
NO_FOOTER_RECORD	Input file does not contain FOOTER record as last ...	
INCORRECT_NO_RECORDS	Incorrect number of records in FOOTER record	
INCORRECT_SEQUENCE	HEADER record contains incorrect file sequence n...	
CANNOT_OPEN_FILE	Unable to open input file	
INCORRECT_PARTNER	Partner number in header record incorrect	
HISTORY_UPDATE_FAILED	Update of history table failed	

3 Specific Interfaces using Interface Controls

3.1 Inbound interfaces

3.1.1 Interface for Transferring stock from Legacy to Client AP0.


Legacy APO data retrieved and sent to Gloffi
Gloffi places file on secure directory for MW to pick up automatically
MW picks up file and places on SAP application server.
SAP APO programs executed and processes the file in zdata directory
Livecache and Database updated in Client APO

1. Data is extracted from the Legacy APO and put in a secured directory in Gloffi.
2. GLOFFI polls directory for any new files
3. GLOFFI wrapper program adds header and footer
4. GLOFFI places file on secure directory for MW to pick up automatically
5. MW picks up file and performs mapping exercise and places on designated secure directory on SAP application server (zdata directory)
6. The SAP custom developed wrapper program /EUR/AEUPDO_STOCK_FORECAST is executed and processes the file from the application server.
7. In Client APO Livecache and Database is updated.

Local support team – needs to ensure that the relevant people have access to the secured directories

GLOFFI access - The polling job needs to have a user id set up

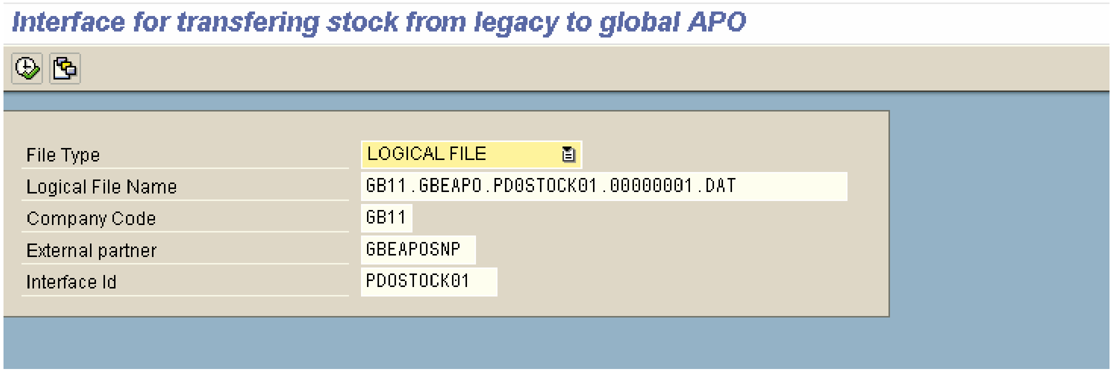
1. For Test – which will be used for test and integration testing (MIT)
2. For Production – which will be used for pre-production (MAT) and Production
3. With a non-expiring password – this will only be used by the application to access the network directory.
4. The User ids needs access to Read, Write access to Local server where the software runs and write access to the GLOFFI server.

3.1.1.1 SAP Components

Example below shows how the Stock Upload works on the SAP side. The process will be the same for UK and Ireland, but the interface controls will vary.

File will arrive in SAP Directory /zdata/EBJ/gb/apo/glb/pdostock01

Name: gb11.gbeapo.pd0stock01.00000001.dat



Interface for transfering stock from legacy to global APO

File Type	LOGICAL FILE
Logical File Name	GB11.GBEAPO.PD0STOCK01.00000001.DAT
Company Code	GB11
External partner	GBEAPOSNP
Interface Id	PDOSTOCK01

The report /EUR/AEUPDO_STOCK_FORECAST is executed with a variant.

```
* Geting the current logical system name.
 call function '/EURGLB/X_INTERFACE_START_PROC'
    exporting
       bukrs          = fp_bukrs
       intid         = fp_intid
       parnum          = fp_rcvprn
       input_filename    = l_file
       parallel_runs_allowed = l_c_key
       normal_run        = l_c_key
    importing
       physical_filename   = fp_v_phy_file
       header          = l_rec_header
    exceptions
       invalid_interface   = 1
       last_run_executing   = 2
       file_not_found     = 3
       no_header_record    = 4
       no_footer_record    = 5
       incorrect_no_records  = 6
       incorrect_sequence   = 7
       cannot_open_file    = 8
       incorrect_partner   = 9
       history_update_failed = 10
       others         = 11.
```

```
* Get forecast data from file
  PERFORM get_forecast_info      USING      p_file
                                            p_bukrs
                                            p_intid
                                            p_rcvprn
                                 CHANGING  v_phy_file
                                            i_input.
* Getting the physical file path name
    call function '/EURGLB/X_INTERFACE_START_PROC'
      exporting
        bukrs                 = fp_bukrs
        intid                 = fp_intid
        parnum                = fp_rcvprn
        input_filename        = l_file
        parallel_runs_allowed = l_c_key
        normal_run            = l_c_key
      importing
        physical_filename     = fp_v_phy_file
        header                = l_rec_header
      exceptions
        invalid_interface     = 1
        last_run_executing    = 2
        file_not_found        = 3
        no_header_record      = 4
        no_footer_record      = 5
        incorrect_no_records  = 6
```

```
          incorrect_sequence    = 7
          cannot_open_file      = 8
          incorrect_partner     = 9
          history_update_failed = 10
          others                = 11.
   open dataset fp_v_phy_file for input in text mode.

* To update total number of records and status
* In interface history table

           call function '/EURGLB/X_INTERFACE_END_PROC'
        exporting
          header           = l_rec_header
          status           = l_status
          recct            = l_record
          trans            = l_trans
          errct            = l_errct
        exceptions
          invalid_interface  = 1
          last_run_executing = 2
          file_not_found     = 3
          status_invalid     = 4
          others             = 5.

* Checking if the material-location combination is valid or not.
```

```
    call function '/SAPAPO/DM_MATERIAL_GET_PEGID'
  exporting
     iv_matnr                   = wa_input-glob_prod_code
     iv_locno                   = wa_input-glob_loc_code
  exceptions
     material_not_supplied        = 1
     location_not_supplied        = 2
     account_not_supplied         = 3
     matid_not_found              = 4
     location_not_found           = 5
     simversion_not_found         = 6
     material_not_in_location     = 7
     dont_use_sobkz_plsec_together = 8
     lc_connect_failed            = 9
     lc_com_error                 = 10
     lc_appl_error                = 11
     version_not_found            = 12
     account_not_found            = 13
     others                       = 14.

* Converting the matnr and locto internal type
    call function '/SAPAPO/DM_MAT_GET_EXTMAT'
  exporting
     iv_matnr      = wa_input-glob_prod_code
     iv_locno      = wa_input-glob_loc_code
  changing
```

```
          iv_ext_matnr  = wa_stock-product
          iv_ext_locno  = wa_stock-location
     exceptions
          not_qualified = 1
          no_material   = 2
          no_location   = 3
          not_unique    = 4
          others        = 5.

        call function 'BAPI_STSRVAPS_SAVEMULTI'
     exporting
          logical_system      = v_own_logsys
          commit_control      = c_commit_control
          planning_mode_usage = c_planning_mode
     tables
          stock               = l_i_stock
          return              = fp_i_return.
```

Details of the interface are selected from the interface control table /EURGLB/XEINTID:

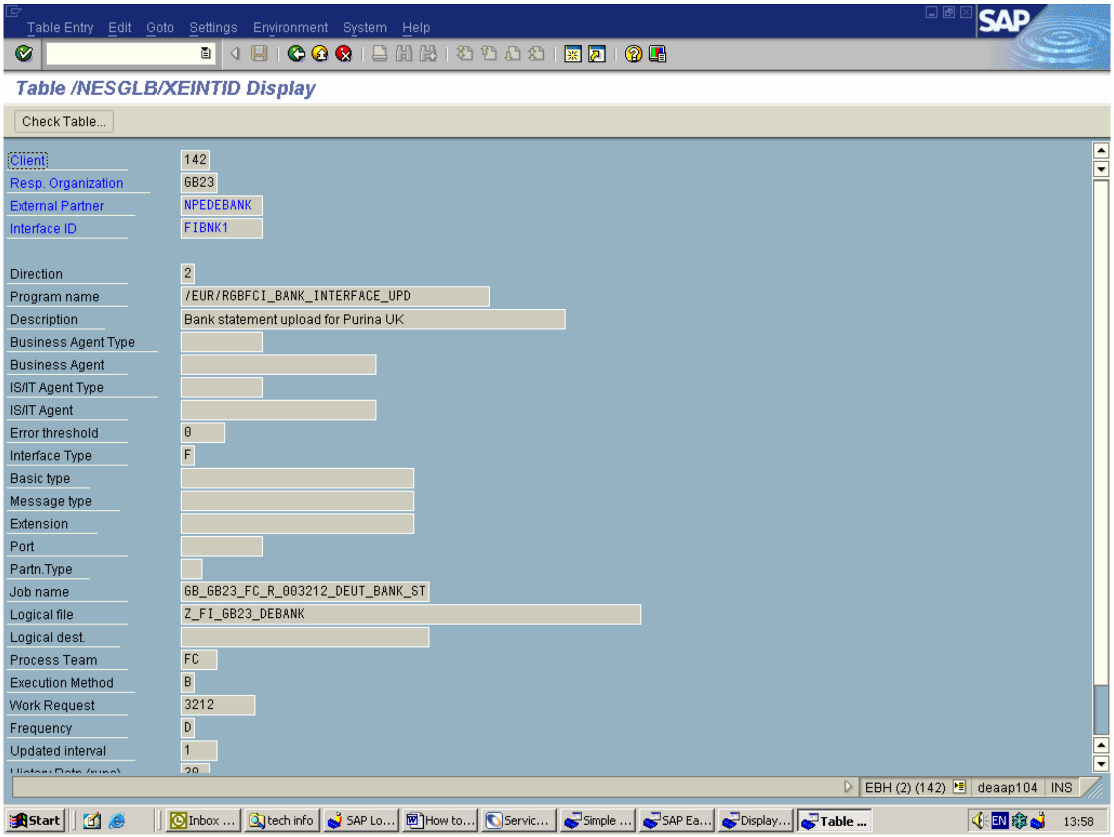


Table /NESGLB/XEINTID Display

Client	142
Resp. Organization	GB23
External Partner	NPEDEBANK
Interface ID	FIBNK1
Direction	2
Program name	/EUR/RGBFCI_BANK_INTERFACE_UPD
Description	Bank statement upload for Purina UK
Business Agent Type	
Business Agent	
IS/IT Agent Type	
IS/IT Agent	
Error threshold	0
Interface Type	F
Basic type	
Message type	
Extension	
Port	
Partn.Type	
Job name	GB_GB23_FC_R_003212_DEUT_BANK_ST
Logical file	Z_FI_GB23_DEBANK
Logical dest.	
Process Team	FC
Execution Method	B
Work Request	3212
Frequency	D
Updated interval	1
History Retn (runs)	30

When the job is complete, there are two places where the results can be seen:

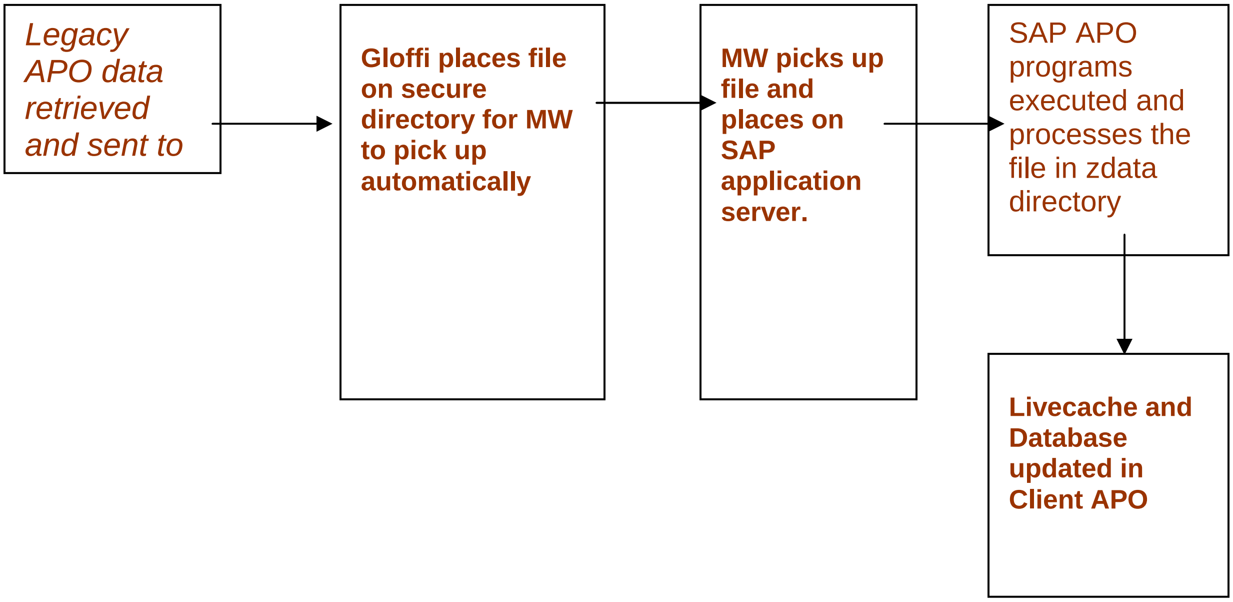
- The first place is via SM36, where a job log can be found for job created by the RFC. The job name can be found in the interface id job The job logs will give details of any errors encountered and are important if the RFC fails and the queue is blocked.
- The second place is via the interface controls history table /EURGLB/XEINTHIS. Use the interface controls to look up the last running of the interface (use the last run date and time). This will give details of the last sequence number, the success/failure code, number of records processed and the date/time the interface was run.

If there is an error with the sequence number, the interface controls table should be checked to identify which file was last processed and which sequence number is expected.   This could either mean a missing file, a duplicate file or an error with the incrementing of the sequence number.

- Missing file: find the missing file and reason for failure and if relevant, resend and then resend the next one in correct order. If no file to resend (missing file is an error), reset the sequence number to the correct one on GLOFFI and resend correct file.
- Duplicate files: investigate if same file sent twice – if so, ignore second file. If error with sequence number, fix on GLOFFI and resend files with correct sequence numbers.

If any errors are encountered with the interface, the job logs and history table should be checked first. If there was an error with the loading and the file needs to be resent, the sequence number must be the next sequence number that SAP is expecting. Local apps can change the sequence number manually if this is required. The file can also be resent if needed.

3.1.2 Interface for Transferring Forecast from Legacy to Client AP0.


Legacy APO data retrieved and sent to
Gloffi places file on secure directory for MW to pick up automatically
MW picks up file and places on SAP application server.
SAP APO programs executed and processes the file in zdata directory
Livecache and Database updated in Client APO

1. Data is extracted from the Legacy APO and put in a secured directory in Gloffi.
2. GLOFFI polls directory for any new files
3. GLOFFI wrapper program adds header and footer
4. GLOFFI places file on secure directory for MW to pick up automatically
5. MW picks up file and performs mapping exercise and places on designated secure directory on SAP application server (zdata directory)
6. The SAP custom developed wrapper program /EUR/AEUPDR_DEMAND_FORECASTis executed and processes the file from the application server.
7. In Client APO Livecache and Database is updated.

Local support team – needs to ensure that the relevant people have access to the secured directories

GLOFFI access - The polling job needs to have a user id set up

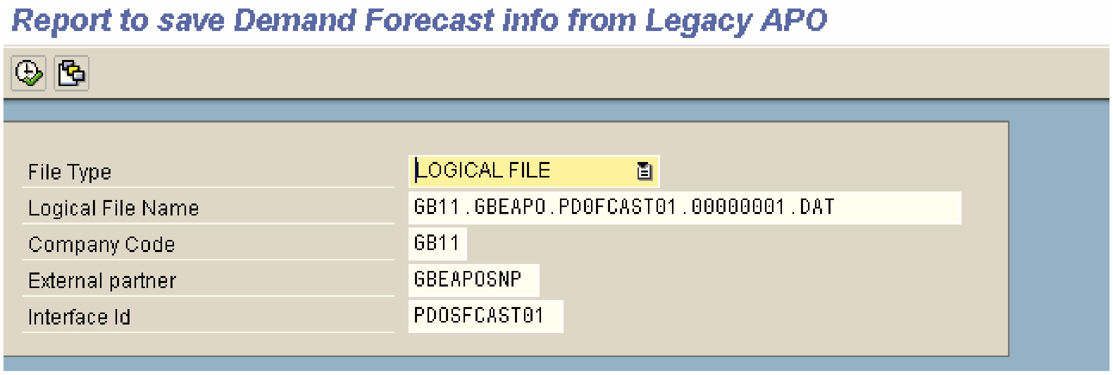
5. For Test – which will be used for test and integration testing (MIT)
6. For Production – which will be used for pre-production (MAT) and Production
7. With a non-expiring password – this will only be used by the application to access the network directory.
8. The User ids needs access to Read, Write access to Local server where the software runs and write access to the GLOFFI server.

3.1.3.1 SAP Components

Example below shows how the Stock Upload works on the SAP side. The process will be the same for UK and Ireland, but the interface controls will vary.

File will arrive in SAP Directory /zdata/EBJ/gb/apo/glb/pdofcast01

Name: gb11.gbeapo.pd0fcast01.00000001.dat



Report to save Demand Forecast info from Legacy APO

File Type	LOGICAL FILE
Logical File Name	GB11.GBEAPO.PD0FCAST01.00000001.DAT
Company Code	GB11
External partner	GBEAPOSNP
Interface Id	PDOSFCAST01

The report /EUR/AEUPDO_DEMAND_FORECAST is executed with a variant.

```
* Geting the current logical system name.
  CALL FUNCTION 'OWN_LOGICAL_SYSTEM_GET'
       IMPORTING
            own_logical_system          = v_own_logsys
       EXCEPTIONS
            own_logical_system_not_defined = 1
            OTHERS                     = 2.
```

```
* Read the forecast info from application server to an internal table.
  PERFORM get_forecast_info      USING     p_file
                                           p_bukrs
                                           p_intid
                                           p_rcvprn
                                 CHANGING  v_phy_file
                                           i_input.



* Getting the physical file path name
  call function '/EURGLB/X_INTERFACE_START_PROC'
       exporting
            bukrs                 = fp_bukrs
            intid                 = fp_intid
            parnum                = fp_rcvprn
            input_filename        = l_file
            parallel_runs_allowed = l_c_key
            normal_run            = l_c_key
       importing
            physical_filename     = fp_v_phy_file
            header                = l_rec_header
       exceptions
            invalid_interface     = 1
            last_run_executing    = 2
            file_not_found        = 3
```

```
          no_header_record      = 4
          no_footer_record      = 5
          incorrect_no_records  = 6
          incorrect_sequence    = 7
          cannot_open_file      = 8
          incorrect_partner     = 9
          history_update_failed = 10
          others                = 11.

open dataset fp_v_phy_file for input in text mode.



* To update total number of records and status
* in interface history table

  call function '/EURGLB/X_INTERFACE_END_PROC'
       exporting
          header            = l_rec_header
          status            = l_status
          recct             = l_record
          trans             = l_trans
          errct             = l_errct
       exceptions
          invalid_interface  = 1
          last_run_executing = 2
          file_not_found     = 3
```

```
status_invalid     = 4
others             = 5.
```

```
* Finding the MATID for corr. Client product code
   call function '/SAPAPO/LOC_LOCNO_GET_LOCID'
      exporting
         iv_locno          = wa_input-glb_loc_no
      importing
         ev_locid          = l_v_locid
      exceptions
         location_not_found = 1
         not_qualified      = 2
         others             = 3.

* Getting the Product no.
      call function '/SAPAPO/W_MATID_GET_MATERIAL'
         exporting
            iv_matid = wa_forecast-matid
         importing
            ev_matnr = l_v_matno.

* Getting the Location no.
      call function '/SAPAPO/W_LOCID_GET_LOCATION'
         exporting
            iv_locid = wa_forecast-locid
         importing
            ev_locno = l_v_locno.
* Saving data
```

```
call function '/SAPAPO/DM_FCST_PUT_BAPI'
      exporting
        i_logsys   = v_own_logsys
      tables
        i_t_dp_fcs = i_forecast2
      exceptions
        error      = 1
        others     = 2.
```

Details of the interface are selected from the interface control table /EURGLB/XEINTID (details vary by box/interface id):

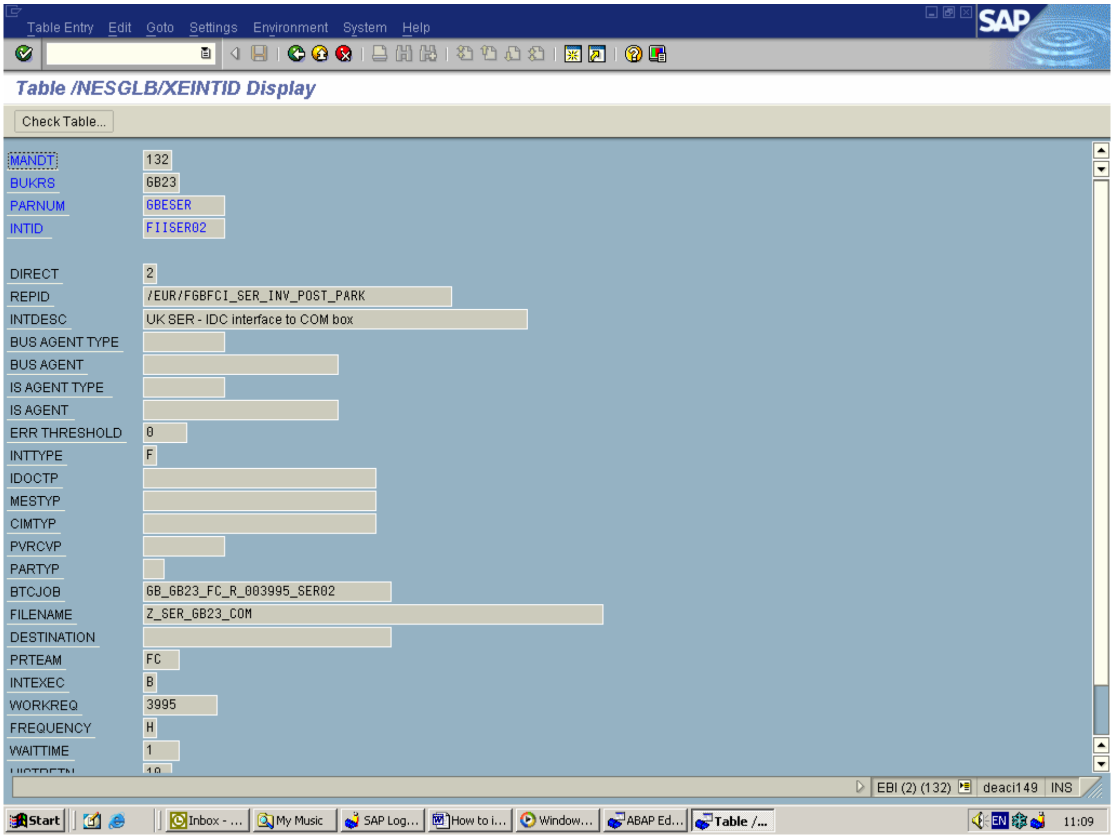


Table /NESGLB/XEINTID Display

MANDT	132
BUKRS	GB23
PARNUM	GBESER
INTID	FIISER02
DIRECT	2
REPID	/EUR/FGBFCI_SER_INV_POST_PARK
INTDESC	UK SER - IDC interface to COM box
BUS AGENT TYPE	
BUS AGENT	
IS AGENT TYPE	
IS AGENT	
ERR THRESHOLD	0
INTTYPE	F
IDOCTP	
MESTYP	
CIMTYP	
PVRCVP	
PARTYP	
BTCJOB	GB_GB23_FC_R_003995_SER02
FILENAME	Z_SER_GB23_COM
DESTINATION	
PRTEAM	FC
INTEXEC	B
WORKREQ	3995
FREQUENCY	H
WAITTIME	1

When the job is complete, there are two places where the results can be seen:

- The first place is via SM36, where a job log can be found for job created by the RFC. The job name can be found in the interface id job above – for the SER interface the name is GB_GB23_FC_R_003995_SER01, 2 or 3.
- The job logs will give details of any errors encountered and are important if the RFC fails and the queue is blocked.
- The second place is via the interface controls history table /EURGLB/XEINTHIS. Use the interface controls to look up the last running of the interface (use the last run date and time). This will give details of the last sequence number, the success/failure code, number of records processed and the date/time the interface was run.

If there is an error with the sequence number, the interface controls table should be checked to identify which file was last processed and which sequence number is expected. This could either mean a missing file, a duplicate file or an error with the incrementing of the sequence number.

- Missing file: find the missing file and reason for failure and if relevant, resend and then resend the next one in correct order. If no file to resend (missing file is an error), reset the sequence number to the correct one on GLOFFI and resend correct file.
- Duplicate files: investigate if same file sent twice – if so, ignore second file. If error with sequence number, fix on GLOFFI and resend files with correct sequence numbers.

Author Bio

Aveek has more than 11 years experience in software analysis and design and custom development both in India and abroad. Aveek has been educated in the US. He has acquired an MS in Economics from Virginia Tech in Blacksburg, Virginia and an MS in Information systems from George Mason University in Fairfax, Virginia. He has handled software projects U.S., Holland, UK, Switzerland and India. He has during his association with IBM, PWC & Consultancy firms in the U.S. gathered wide experience in developing leading edge technological solutions. He has been heavily involved in developing Materials Management System, Manufacturing & Material Handlings applications, Process Control System Interfaces, Warehouse Control applications and financial applications in almost all the software projects he has undertaken. For the past 6 years, Aveek has been associated with a number of SAP projects in India and abroad. Aveek has been involved in handling Process Control Interfaces in the Manufacturing area. Aveek was a core member of the Warehouse Management system team in the University of Michigan in Ann-Arbor, Michigan, USA.