TEMENOS™

TEMENOS T24

Globus Interface Tool

User Guide

Table of Contents

Introduction

This Document aims at explaining the Functionality of the GLOBUS INTERFACE TOOL. This document provides detailed information about every Field of all the Files used in the GIT. It provides information to the user to understand the different parameter files and their setup of the interface GLOBUS Interface Tools (GIT).

General Overview

The purpose of this guide is to help the user understand the various Templates and the fields in the Templates and the crucial setup of any PARAMETER Files in GIT.

In order to get the most out of this manual, it is essential that you are familiar with the standard procedures regarding navigating your way around the system. This includes items such as program access via menus, data entry and editing, mandatory and multi-level fields, committing details, and so forth.

Assumptions

It is assumed that the reader of this guide has knowledge of the following.

- Basic knowledge in UNIX operating systems.
- Fair knowledge in TEMENOS T24 TM

Abbreviations used in this Guide

S – Single Value Field

M – Multi Value Field

AMS – Associated Multi Value Starts Here

AM – Member of Associated Multi Value

AMC – Associated Multi Value Closes Here

GIT – GLOBUS Interface Tools

Process Overview

- Many of the interfaces developed require one or more of the following: selection of records, mapping of data, formatting of data, and finally, transporting the results out of T24™. All of the interfaces require that T24 initiate the process and push information out of the system.

- Regarding the Inward flow (i.e. feed from External Source), extract the data from the feed, and update the T24 application

- GLOBUS *Interface Tools*® (*GIT*) contains a set of parameter tables and APIs which will help in building an interface either entirely through the use of the *GIT* or through a combination of code and certain elements of the *GIT*.

OUTWARD PROCESS

The *GIT.INTERFACE.OUT* table contains one record for each interface with a @ID being the name of the interface (e.g. IDOM). This record can define the entire interface by specifying the table(s) to select (without the FBNK. or F.), the selection criteria in a multi-value field (without the WITH; but with the AND, OR), and the transport to use.

The routine GIT.INTERFACE.OUT.RUN is launched when a User uses the VERIFY function on a *GIT.INTERFACE.OUT* record. It is also possible to call the GIT.INTERFACE.OUT.RUN directly from an external program or during the EOD process.

This process selects the appropriate record(s) for each selection type and passes the results as a parameter to the GIT.INTERFACE.OUT.RUN routine. In addition, the @ID of the *GIT.INTERFACE.OUT* record and the list of selection types (e.g., DDA, SWAPS, MM, etc.) are passed as parameters.

The GIT.INTERFACE.OUT.RUN routine uses the mapping table(s) and formatting table(s) with an @ID of GIT.INTERFACE.OUT@ID-TYPE (e.g., IDOM-DDA).

This mapped and formatted data is then brought together as output data and transported through the specified mechanism.

INWARD PROCESS

The *GIT.INTERFACE.IN* table contains one record for each inward interface with @ID being the name of the interface, e.g., IDOM, and the field 'TYPE' being a multi-value set where the method of updating T24 files [OFS/WRITE] can be defined for each Type.

If Write is selected, the WRITE.STATUS has to be defined (IHLD/INAU). If any multi-value field in a record is defined as OFS, then the OFS.SOURCE.ID, LOGIN and PASSWORD has to be input by the user.

The field, MULTI.RECS, denotes whether an incoming message consists of multiple records, and if so, then the field REC.DELIM has to be input.

The routine GIT.INTERFACE.IN.RUN is launched when the *GIT.INTERFACE.IN* record is run under VERIFY mode. It is also possible to call this IN.RUN routine directly from an external program or during the EOD process.

The routine calls the GIT.TRANSPORT.RUN routine, which will read the corresponding *GIT.TRANSPORT* record, as the ID for *GIT.INTERFACE.IN* and *GIT.TRANSPORT* are the same.

GIT.TRANSPORT reads the corresponding transport record and picks up the incoming messages from the pre-defined location and writes into the file, F.GIT.IN.MSG, with the IDs in *F.GIT.IN.KEYLIST*.

The next routine called by IN.RUN is *GIT.PROCESS.MAPPING.IN*, which does the mapping and writes into the file *F.GIT.MAPPED.IN*.

Menu Structure

The GIT utilities Menu will be available under the Main menu "**IMPLEMENTATION UTILITIES**" as shown in the screen shot below

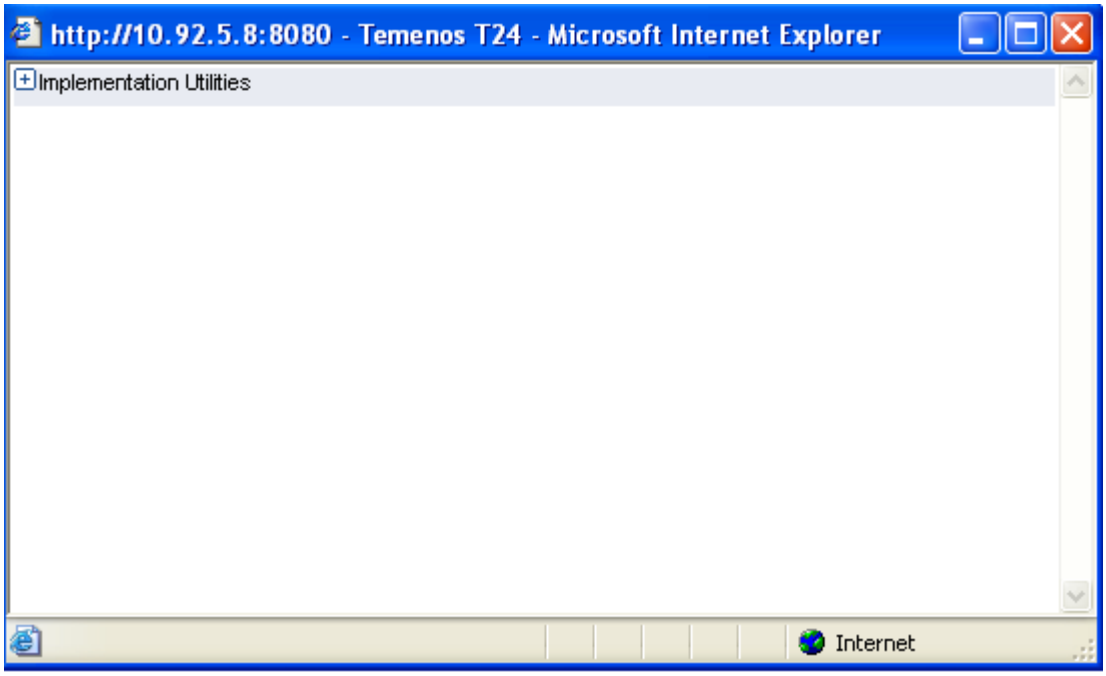


Fig 1 - **UTILITIES MENU**

The Implementation Utilities Menu when clicked will drop down to the additional submenus.

The GIT Utilities will be available under "**INTERFACE UTILITIES**" Submenu.

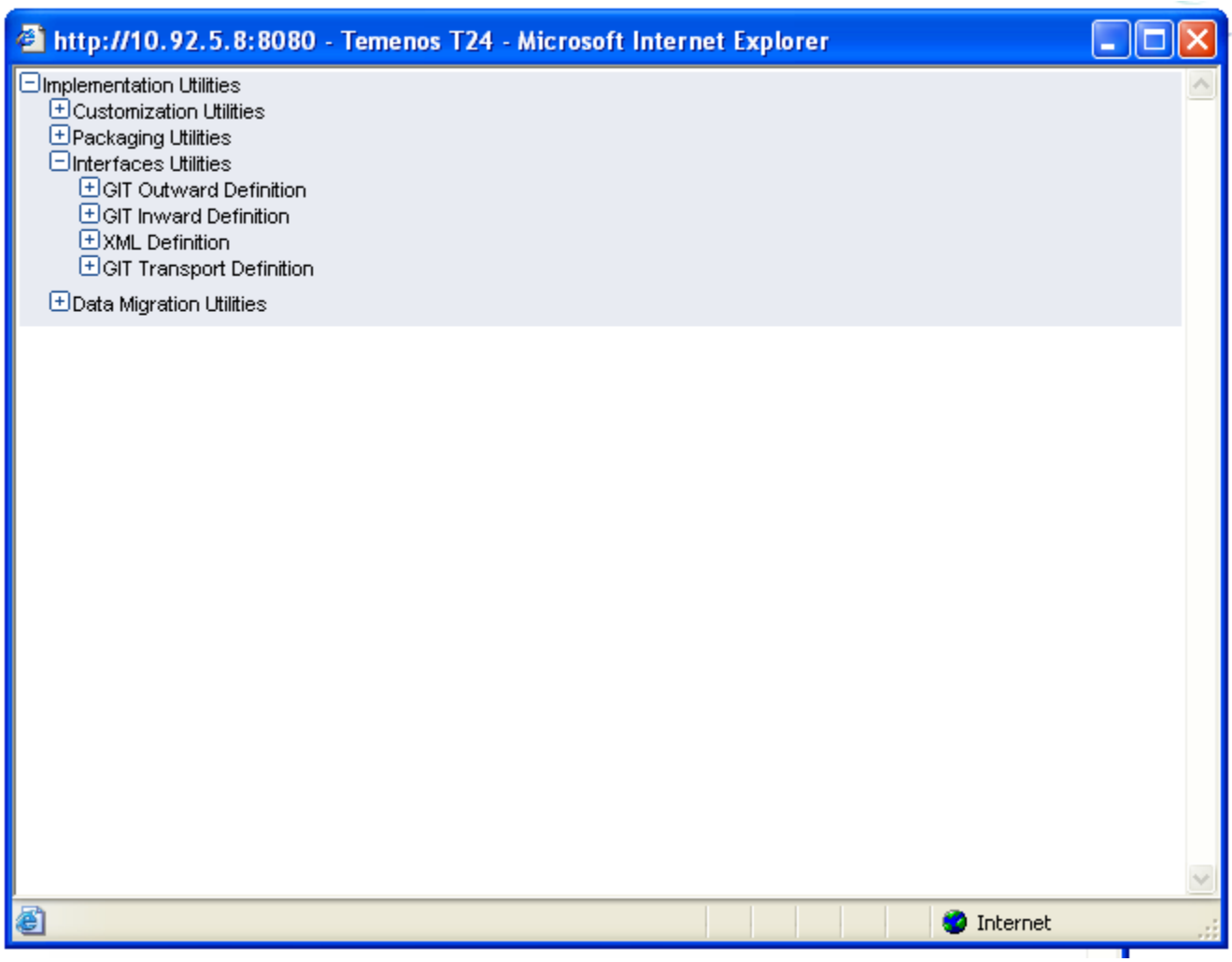


Fig 2 - INTERFACE UTILITIES MENU

The INTERFACE UTILITIES menu when clicked will drop down to the additional submenu listing

All the utilities of the GIT INTERFACE tool is shown in the next screen shot below

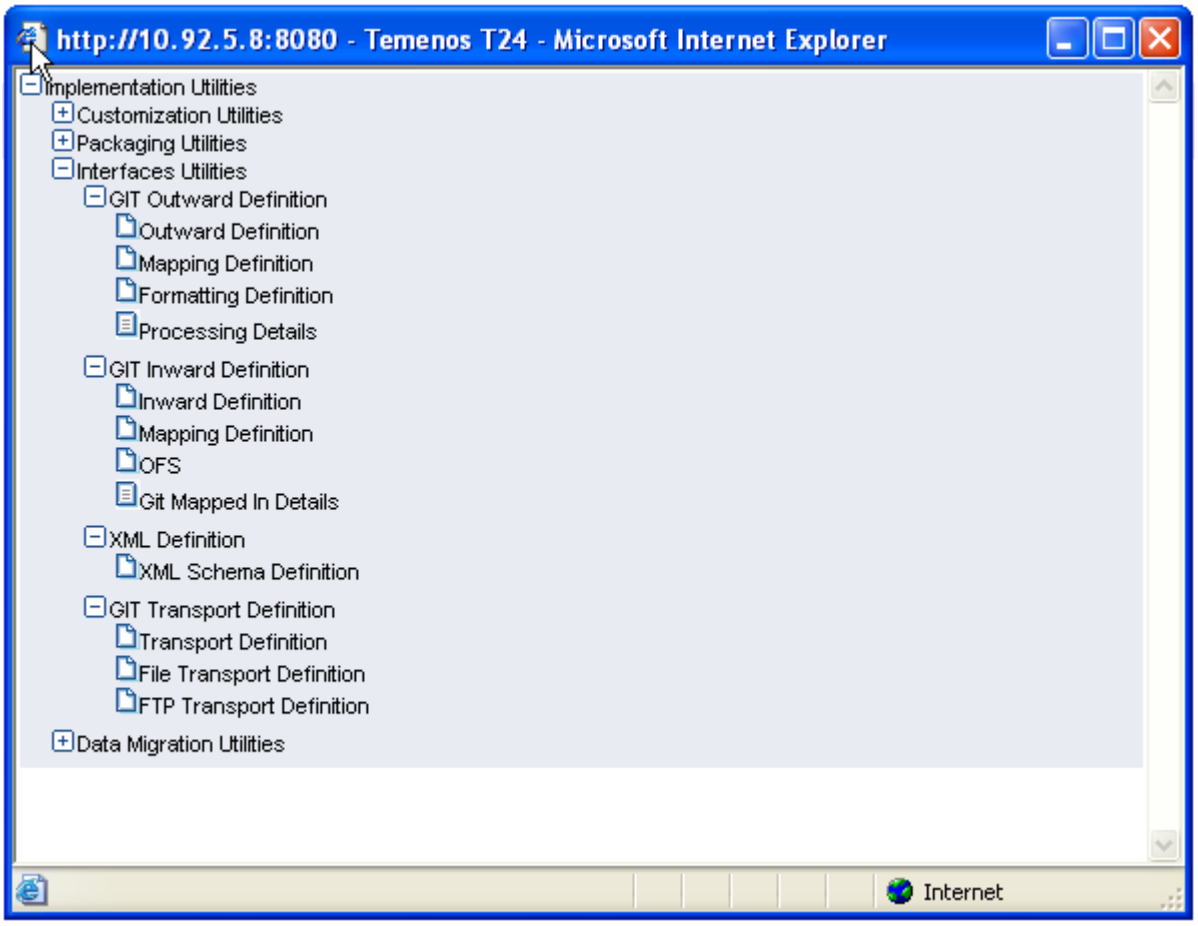


Fig 2 - GIT UTILITIES MENU

The final expanded menu of the GIT UTILITIES will appear as shown above. The first Drop down is the Outward Definition followed by the Inward Definition, XML Schema Definition and Finally the Transport definition for Transport Details.

Outward Processing:

The following flowchart depicts the phases of the GIT OUTWARD process:

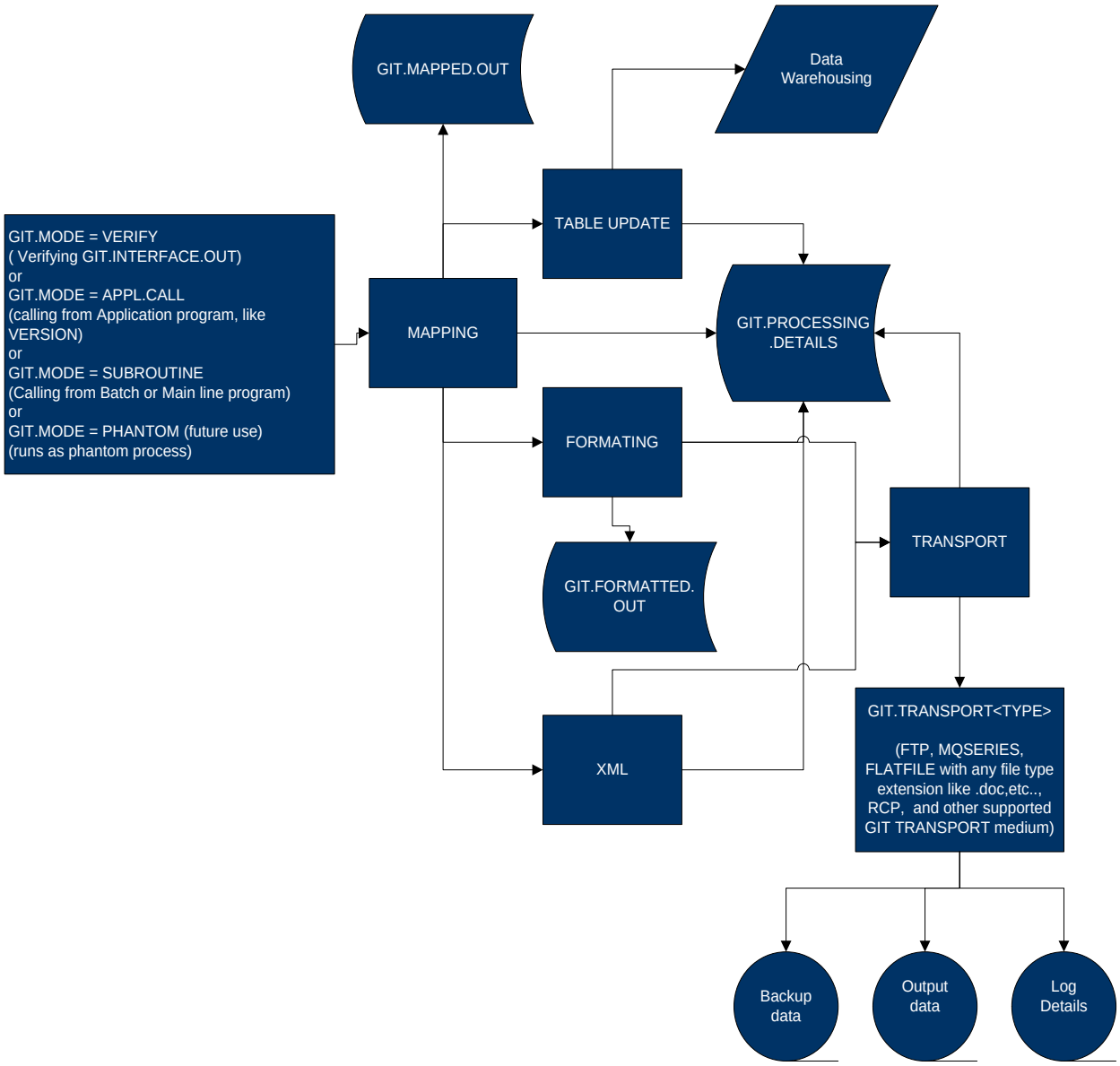


Fig 3 - GIT OUTWARD Diagrammatic Representation

The Outward Processing of GIT includes the following Parameter files:

- GIT.ITNERFACE.OUT
- GIT.MAPPING.OUT
- GIT.FORMATTING.OUT

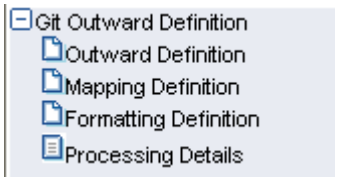


Fig 4 – GIT Outward Menu

GIT.INTERFACE.OUT,MB.DEFINE

About This File…

This is the main table containing the parameters for defining an Outward Interface. (Data selection, mapping, formatting, output and transport)

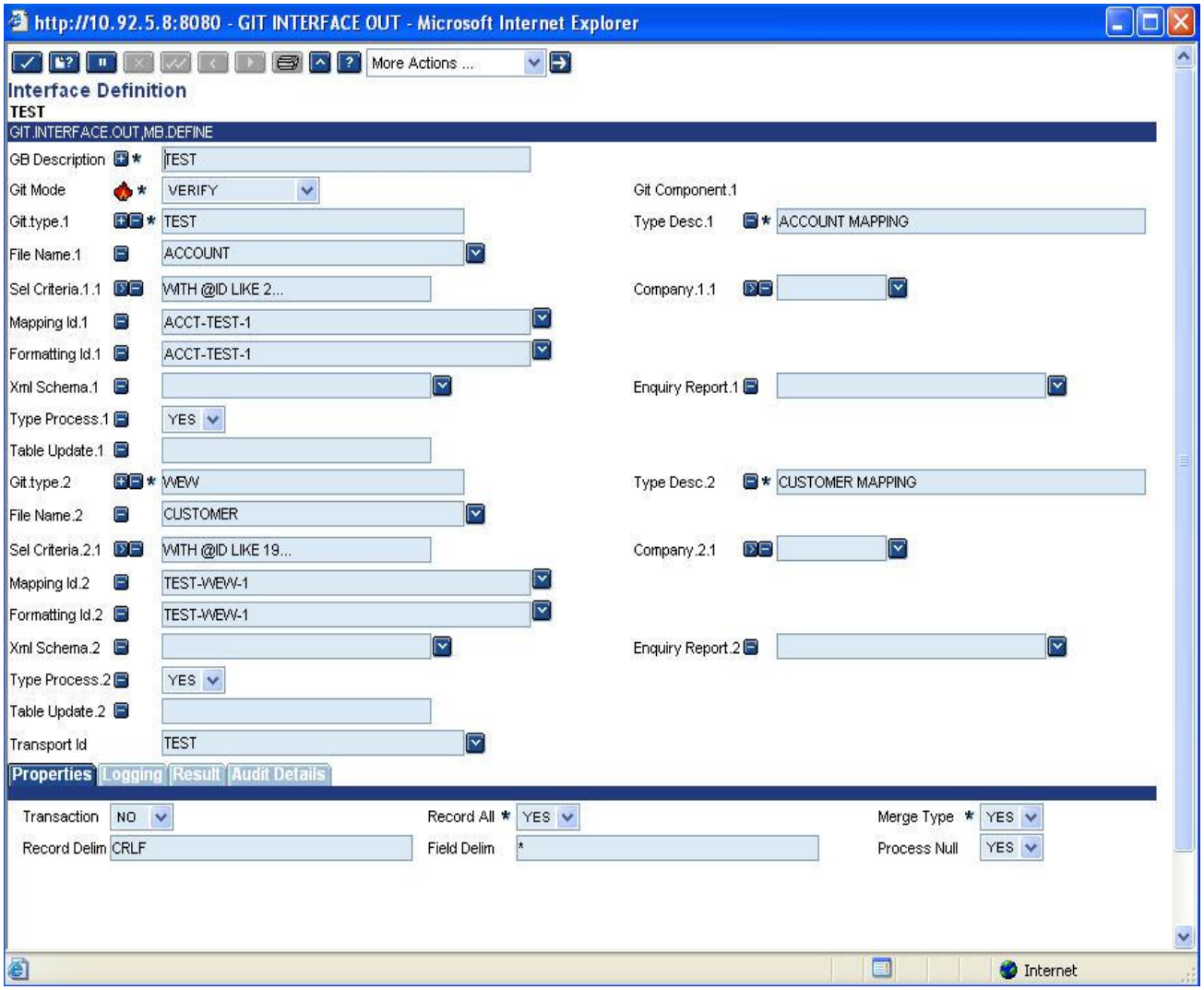


Fig 5 – GIT.INTERFACE.OUT,MB.DEFINE Screen shot

GIT.INTERFACE.OUT,MB.DEFINE - Field Descriptions.

@ID

Name of the Interface.1-20 Alphanumeric characters.

Back

GB Description

Contains the description of the Interface.1-50 Alpha numeric characters.

Mandatory field

Back

Git Mode

The mode in which the application should run.

VERIFY/SUBROUTINE/APPL.CALL/PHANTOM. Phantom for future use.

VERIFY - The application GIT.INTERFACE.OUT, when verified will do Mapping, formatting and will output the details Using the transport specified in the GIT.INTERFACE.OUT table.

SUBROUTINE - GIT.INTERFACE.OUT when called from Batch or as a Mainline Program.

APPL.CALL - This is similar to the SUBROUTINE mode but this Mode is used when GIT needs to be invoked as a Part of committing a record in an application or During the authorization stage. (i.e.) GIT can be run as Version/Version control routine.

Mandatory field.

Back

Git Component

Components of GIT being used.

MAPPING/FORMATTING/ TRANSPORT/ALL

This field is no input field if MODE is Verify.

Back

Git Type

Interface Type.1-40 Alpha numeric characters.

Mandatory field.

Back

Type Desc

Description of the Interface field TYPE .1-50 alphanumeric characters.

Mandatory Field

Back

File Name

Valid T24 application. $HIS, $NAU, $ARC can also be included

E.g. CUSTOMER, CUSTOMER$HIS, CUSTOMER$NAU etc.

Mandatory field.

If the TABLE field value is of PGM.TYPE 'T' ((i.e.) Concat file, and if the concat ids need to be split then the following format can be followed.

For example – consider the application CUSTOMER.ACCOUNT

The ID of this application is Customer.Id.Each Customer.ID in turn contains one or more Account IDs. If the Account details are to be output using GIT then in GIT.INTERFACE.OUT table

- **Set the TYPE.DESC- TABLE (value in TABLE field)". SPLIT" (E.g.: CUSTOMER.ACCOUNT.SPLIT)**
- **TABLE ------- CUSTOMER.ACCOUNT**

In GIT.MAPPING.OUT table

- **Set the SS.ID -- ACCOUNT**

Back

Sel Criteria

Selection criteria.

E.g.: 1) WITH MNEMONIC LIKE …BANK….
WITH @ID EQ 500171

2) @<Routine name> - can also be attached. The routine will be used for the Selection of records. The arguments passed is as follows

For E.g.:

@GIT.SEL.PGM(GIT.SEL.BUILD,GIT.PGM.RECORD.LIST,NEW.ID.COMPANY,GIT.FUTURE.1,GIT.FUTURE.2,GIT.FUTURE.3,GIT.ERR)

Where GIT.SEL.BUILD –	This is the Selection criteria formed Using the other sel.criteria conditions.
GIT.PGM.RECORD.LIST-	The outgoing argument that Contains the selection record list formed Using the routine. If the selection should only be by the @routine then nullify the GIT.SEL.BUILD variable in the routine, as the record list formed will contain the selection by both the routine and the other selection criteria.
NEW.ID.COMPANY-	The company Id given in the Record.

If this field is null then all the records in the application specified in field TABLE will be selected.

Back

Company

Valid COMPANY id.
E.g.: US0010001, DE0010001 etc

If this field is null then the selection of records will be for the current company

Back

Mapping Id

Consists of 3 components. Interface.id – Int.type -seq.no

Interface.id – Current GIT record id.

Int.type -- The current TYPE field value.

Seq.no -- Can be any numeric value.

If no value is Input, then GIT will create a default GIT.MAPPING.OUT record in IHLD.

Back

Formatting Id

Consists of 3 components. Interface.id-Int.type-seq.no

Interface.id - Current GIT record id.

Int.type - The current TYPE field value.

Seq.no - Can be any numeric value.

Should be a valid id in GIT.FORMATTING.OUT.

If no value is Input, then GIT will create a default GIT.MAPPING.OUT record in IHLD.

Back

XML Schema

Used to send and receive XML messages

The ID of the GIT.XML.TABLE record is specified in this field.

Back

Enquiry Report

To create a Report Based on an enquiry output, should be a valid enquiry in the ENQUIRY.REPORT application.

Back

Type Process

To process the TYPE specified, or NOT

Values allowed are either YES or NO. If YES or null then the TYPE will be processed. If NO, the TYPE will not be processed.
Default value is null.

Back

Table Update

Used to create a T24 file.

If fields are defined then a T24 file will be created with all the necessary steps.

Back

Transport Id

The GIT.TRANSPORT id using which the processed details are transported.

Valid id in GIT.TRANSPORT.

Back

Transaction

JOURNAL.UPDATE/F.WRITE/WRITE indicator

Back

Record All

Determines if all the records will create one output file.

If 'Yes' then all the records selected for a TYPE will be put onto a single Output file, if 'N' then the number of files equal to the number of the records selected will be output.

Mandatory field.

Back

Merge Type

Determines whether all the defined TYPEs will be merged as a

single output file or each record processed will be sent as individual files.

Holds value 'YES' or 'NO'.

Mandatory field

Back

Record Delim

If RECORD.ALL is Y, then this field is used to specify the delimiter Between the records.

E.g.: CRLF (for line feed), #, * etc

Any valid delimiter except invalid special characters are allowed

Back

Field Delim

Delimiter used between different fields within the same record.

E.g.: #, * etc

Any valid delimiter except invalid special characters are allowed

Back

PROCESS.NULL

Determines whether to process the null field values and return them as well

Holds value 'YES' or 'NO'.

Back

Pre Routine

Routine to be executed before GIT processing

Valid Routine name

Back

Post Routine

Routine to be executed after GIT processing

Valid Routine name

Back

Detail Log

Specifies whether and how the detail log should be populated.

Holds value FULL/ERROR/NONE. If the value is FULL, then the log is recorded in the GIT.PROCESSING DETAILS file. If ERROR, then only the error details if encountered, are logged on to the file. If NONE then no log is found.

Back

Error Stop

Determines whether an error during any part of process should halt the entire process. When set to 'Yes' will produce an override and will halt the entire process if any error is encountered.

Holds value 'YES' or 'NO'. Default value is 'YES'.

Back

Report View

Determines whether to view the Outgoing message.

If 'Y' then the outgoing message will be viewed on the screen (only for Verify mode).

Default value is 'N'

Back

Clear Files

Clear files before processing.

No input field

Back

Show Progress

Display of record being processed. Shows the Ids that are mapped

Will display the record id, which is being processed by GIT. Valid only for VERIFY mode.

Back

Process Speed

Used for speedy generation of output.

Accepts values “THROUGH” or “QUICK”.
This field helps in enhancing the speed of the interface processing.
If set as THROUGH or QUICK will produce the output. But no log details are maintained (i.e.) The MAPPED, FORMATTED record details are not maintained.

Back

Cache Off

To turn off cache .

Back

Process Info

Processed Information. Displays the number of records mapped and processed for the particular selection.

Updated by System.

No Input field.

Back

GIT.MAPPING.OUT,MB.DEFINE

MAPPING FILE

About This File…

This table contains the mapping rules for GIT to map fields to data for each record returned by the selection(s) specified in GIT.INTERFACE.OUT,MB.DEFINE. Results of mapping are saved in GIT.MAPPED.OUT table, record-by-record.

The following table gives a brief description about the ID and the fields present in GIT.MAPPING.OUT,MB.DEFINE file.

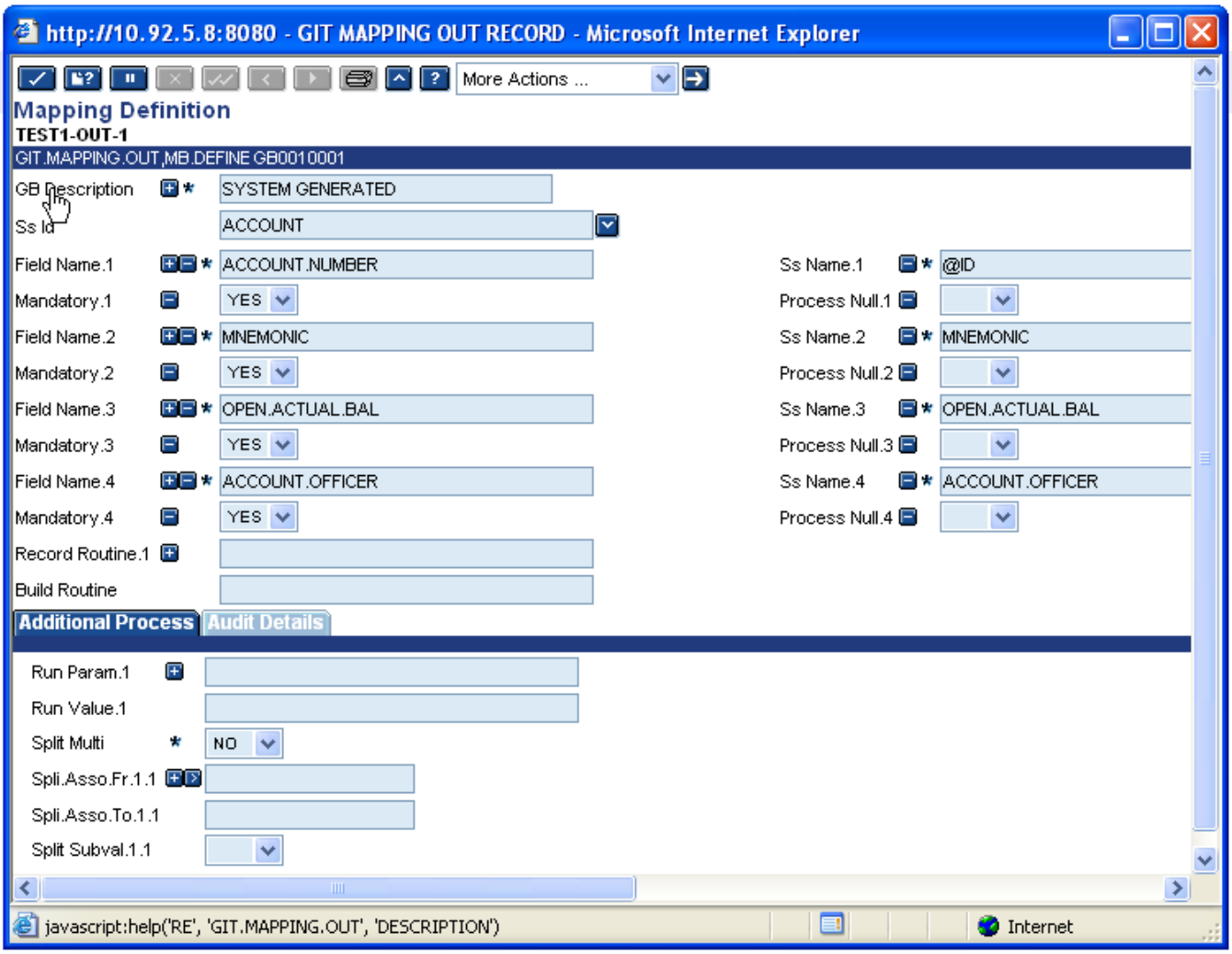


Fig 6 – GIT.MAPPING.OUT,MB.DEFINE Screen shot

GIT.MAPPING.OUT,MB.DEFINE - Field Descriptions.

@ID

The ID is made of three parts.
Name of the interface, Type and a sequence number.

First and second part will be validated against the ID of the GIT.INTERFACE.OUT record and the TYPE within the GIT.INTERFACE.OUT record. Once the GIT.INTERFACE.OUT record is created and the mapping id is supplied, then this record will be created in IHLD.

Back

GB Description

Description of the mapping record.1-35 alphanumeric characters.

Mandatory field

Back

SS ID

This field contains the Application name. The application name can be same as the one defined in TABLE field of GIT.INTERFACE.OUT table for the TYPE (Second part of the ID of GIT.MAPPING.OUT record.) or may be different.

This file forms the base file from which the fields defined in the FIELD.NAME field are extracted.

Back

Field Name

User defined name of the field to be used by the Formatting process. Must be unique in the mapping record.

Mandatory field.

Back

SS Name

Name of the field to be mapped

The field may be in the STANDARD.SELECTION record of the application defined in the SS.ID field or a constant (i.e.) a static value using double quote.eg: “static value”. The standard T24 variables can also be used.

- **The T24 common variables for e.g. TODAY, ID.COMPANY, LCCY, OPERATOR, APPLICATION, ID.NEW, LNGG, PGM.VERSION, V$DISPLAY can be given by appending a '!' in front. E.g:! <T24 common variable>**

- **! INS - If the value of the FIELD.NAME field that is defined before the current FIELD.NAME field has to be used in the current field then INS is used. E.g.! INS <PREV.DEFINED FIELD>**

- **If the field values of the COMMON records like R.SPF.SYSTEM, R.INTERCO.PARAMETER, R.ACCOUNT.PARAMETER, R.DATES, R.COMPANY are to be extracted, use the following format. "!"<COMMON variable>">"<FIELDNO>**

  E.g.: !R.SPF.SYSTEM>15 – will extract the OPERATING.SYSTEM field value of the SPF.

Back

Conversion

Used to convert the mapped field value. The following functions are available

- **A routine can be used to do more calculations or conversions and to set error condition.**
  @ <routine name>(GIT.MAP.VALUE, CONV.ERR).

  GIT.MAP.VALUE – Incoming & Outgoing argument

- **Formatting the Mapped value can be done as follows**
  FMT, <formatting condition>

  E.g.: FMT, 5R, FMT, 10%L

- **ICONV, OCONV can be used as follows. ICONV,<conversion condition>**
- **APPEND:**
  To add a String value or a previously defined field value to the existing field value APPEND can be used in the following format.

  APPEND "Sample Test"

  APPEND <previously defined field name>

- **INS – If previously defined field name's value should be used in the current field name then INS is used.**
  Format: INS <Previously defined field name>

- **ADD, SUB, MUL, DIV – If two field values (numeric) needed to be added, subtracted, multiplied, divided to get the current field's value then use the above**
  Conversion values in the following format

  <Operation> <fld1>,<fld2>

Operation – ADD/SUB/MUL/DIV

Fld1, Fld2 – Previously defined field names

- **LINK, LINKI, LINKV – If a value has to be taken from some other Application then LINK comes into play.**
  Format 1: LINK">" <Application> " > "<Previously defined field name> ">" <field name from the Application>

  Previously defined field name – This should be the Id of the Application Linked.

  E.g.; LINK>CUSTOMER>ID.VAL>MNEMONIC

  Format 2: LINK">" <Application> " > " < field name from the Application>

  Here the ID of the record in the Application will be the Current field name's value

  E.g.; LINK>CUSTOMER>MNEMONIC.

  Format 3: If the field to be fetched from the Linked Application is an IDescriptor field then use the following format.

  LINKI ">"<Application name> ">" <IDescriptor field name>

  E.g.: LINKI>CUSTOMER>CU.SAMPLE where CU.SAMPLE is the IDescriptor field.

  Format 4: LINKV is used when the value got using the link option is null then the original value used for link will be returned.

  E.g.: LINKV>CUSTOMER>NAME.2 –If the NAME.2 field is null then the Current field name's value will be returned.

- **CONCAT – When more than one field name's field values need to be concatenated then use the CONCAT in the following format**
  CONCAT <fld1>","<fld2>","<fld3>

  E.g.: CONCAT field1, field2, field3

  Field1, field2, field3 – Previously defined field names.

- **EXTRACT – Used to extract a part of field value from the mapped value.**
  Format: EXTRACT <start position>","<end position>

  E.g.: EXTRACT 1,4

- **ABS – Absolute Mapped value**
  Format: ABS

- **FIELD – used to extract the value in the n^{th} position of VM and the m^{th} SM position**
  Format: FIELD VM (or @VM)",","<nth VM position>

  E.g.: FIELD VM, 3 – will extract the field name's value's 3^{rd} VM value.

  Format 2: FIELD SM (or @SM) ",","<nth VM position>","<mth SM Position>

  E.g.: FIELD SM, 3,2 – will extract the 2^{nd} SM value from the 3^{rd} VM field value.

- **CHANGE – Used to change the occurrences of a string or a character to another character.**

Format: CHANGE <From. Val>”,” <To. Val>

E.g.: CHANGE #, VM

- **IF – IF condition can be used in the following format**
  Format: IF <fieldname1> <operator> <fieldname2> <fieldname3> <fieldname4>

  Operator – EQ, GT, LT, LE, GE, NE

  If the IF condition is successful then fieldname3 value will be used else fieldname4 will be used.

  Fieldname1, fieldname2, fieldname3, and fieldname4 – previously defined field values.
- **TRIM – Used to trim the unwanted spaces.**
  Format: TRIM.

Mandatory

Determines whether the field is Mandatory.

If this field is set to ‘Yes’ then if the FIELD.NAME to which this field is attached, is null then the error is recorded in the GIT.PROCESSING.DETAILS file and the output is written without this field value. Default value is ‘NO’

Back

Process Null

Used to specify whether the null value needs to be sent to output stage.

Holds values ‘Y’ or ‘NO’. If Mandatory is set to ‘NO’ then input to this field is

Allowed and if the value is ‘Yes’ and if the mapped value is null then

Space will be appended.

Back

Field Routine

Valid routine name. If any additional manipulation needs to be done on the mapped data then field level routine can be attached. E.g.: abc.rtn (GIT.MAP.VALUE) where GIT.MAP.VALUE contains the field value to be manipulated.

Back

Record Routine

Valid routine name. This routine will be called to do some specific Processing on the entire record.

Eg: **@REC.RTN** (GIT.ID.MAPPED, GIT.MAPPED.REC, GIT.MISN, GIT.R.PROCESS, GIT.ERR)

GIT.ID.MAPPED – CURRENT MAPPING record ID
GIT.MAPPED.REC – MAPPED Record list
GIT.MISN - Count of number of records processed.
GIT.R.PROCESS – FLAG to create the mapped record or not
Default is 1

Back

Build Routine

Valid routine name. This routine will be called with the selected record list from GIT.INTERFACE.OUT selection criteria and then this routine can be passed back with the new record list to GIT, which will be processed
build.rtn(GIT.RECORD.LIST,GIT.FUTURE.1,GIT.FUTURE.2,GIT.FUTURE.3,GIT.ERR)

GIT.RECORD.LIST – Incoming/Outgoing argument. Contains the Selected record list.

GIT.FUTURE.1,GIT.FUTURE.2 &GIT.FUTURE.3 are all for Future use

Back

Split Multi

Whether to split multi-value set to multiple records.

Multi value field:

If this field is set to YES and multi-value field name (standard selection name) is given in SPLI.ASSO.FR, SPLI.ASSO.TO and SPLIT.SUBVAL set to 'NO' then if the current selected record has more than one values for that field name, then each multi-value will be created as a separate record including the other mapped fields.

Sub value field:

If this field is set to YES and if a Sub value field needs to be split then the SUBVALUE field name (standard selection name) is given in the SPLI.ASSO.FR, SPLI.ASSO.TO and SPLIT.SUBVAL set to 'YES' will split the sub value field and

produce number of records equal to the number of sub values with all the other Mapped fields.

Associated set:

If an associated set of multi values needs to be split then the starting field name of the Associated set is given in the SPLI.ASSO.FR field and the last field name in the Associated set is given in the SPLI.ASSO.TO field, SPLIT.SUBVAL set to 'NO' will produce number of records equal to the number of Multi values of the associated set.

If the Associated set contains both Multi value and sub values then the Multi value field name is given in the SPLI.ASSO.FR, SPLI.ASSO.TO and SPIT.SUBVAL equal to 'NO' and the sub value set start and end value are given in SPLI.ASSO.FR and SPLI.ASSO.TO respectively, and SPLIT.SUBVAL is set as 'YES'.

Combination of Multi and Sub values:
If a Multi value field and a sub value field are to be split, then the number of records produced will be equal to the maximum number of multi value or sub value whichever is maximum.

Back

Spli Asso Fr

The field name from which the split has to be made is given in this field.

Valid field name from the standard selection record of the Application that is in the SS.ID field.

Back

Spli Asso To

The field name to which the split has to be made is given in this field.

Valid field name from the standard selection record of the Application that is in the SS.ID field.

Back

Split Subval

The field values given in SPLI.ASSO.FR, SPLI.ASSO.TO should be split or not is given in this field.

Holds the values 'YES' or 'NO'.

If SPLI.ASSO.FR, SPLI.ASSO.TO are sub value fields and if they have to be split then set this field to 'YES'

If SPLI.ASSO.FR, SPLI.ASSO.TO are multi value fields and if they have to be split then set this field to 'NO'

Back

GIT.FORMATTING.OUT,MB.DEFINE

FORMATTING TABLE

About This File…

The GIT.FORMATTING.OUT table contains the formatting rules to apply to each mapped record. The results of the formatting will be saved in the GIT.FORMATTED.OUT table, record by record.

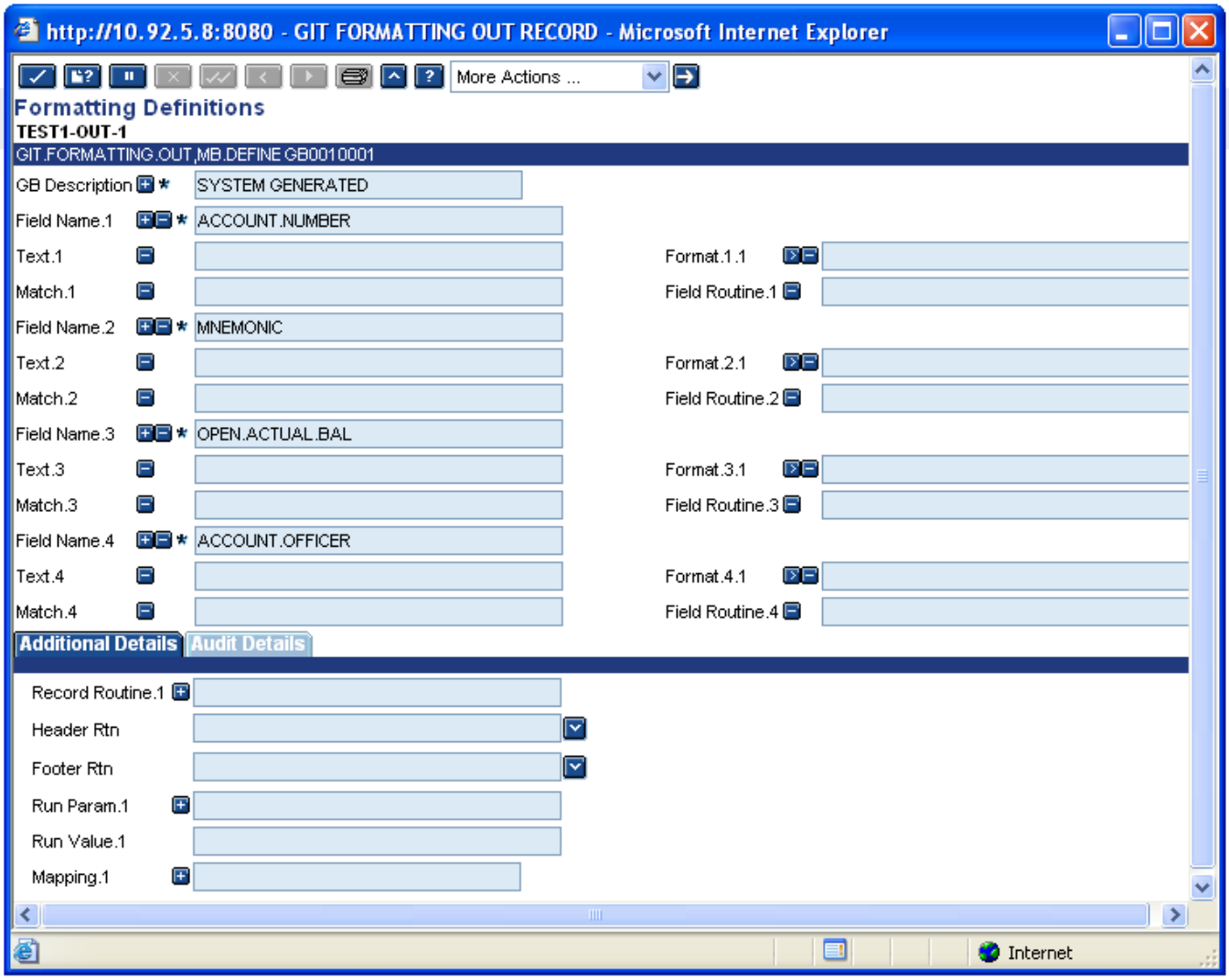


Fig 7 – GIT.FORMATTING.OUT,MB.DEFINE Screen shot

GIT.FORMATTING.OUT,MB.DEFINE - Field Descriptions.

@ID

The ID is made of three parts.
Name of the interface, Type and a number.

First and second part will be validated against the ID of the GIT.INTERFACE.OUT record and the TYPE within the GIT.INTERFACE.OUT record. Once the GIT.INTERFACE.OUT record is created and the formatting id is supplied, then this record will be created in IHLD. Header and Footer will be appended to data while sending the data out, if header and footer are defined.

@ID consists of INTERFACE-TYPE-SEQNO for data processing. INTERFACE-TYPE-HEADER for header, INTERFACE-TYPE-FOOTER for footer.

Back

GB Description

Description of the formatting record.1-35 Alphanumeric characters.

Back

Field Name

Name of the field from GIT.MAPPING.OUT table. Static fields can also be entered in the format "STATIC"

Valid field name from GIT.MAPPING.OUT table having ID same as GIT.FORMATTING.OUT record.

Back

Text

Any text or special characters to be used instead or concatenated with the field data.

The following are the functions available for the TEXT field, only if the third part of the id is HEADER/FOOTER record.

- **CRLF – Will insert a line feed.**
- **SPACE – To insert spaces.**
  Format: SPACE"("<Length>")"
  E.g.: SPACE(15) –will insert 15 spaces in the current line.
- **Common T24 variables can also be given.**
  Format: !TODAY,! LCCY,! ID.COMPANY,! LWORK.DAY
- **If the Mapped field name is given then the value of the Mapped field will be displayed.**

Back

Format

Specifies the formatting to be done on the data read from GIT.MAPPED.

In GIT.MAPPING, if PROCESS.NULL is yes, and the field has value null for the field name defined in FIELD.NAME field (if not static), if a format is defined for that field, then that format will be applied. If field has value then data will be formatted according to the defined format. The following formatting conditions can be applied.

- **CRLF – If CRLF is specified then the Mapped value is appended With the line feed.**
- **TRIM – Unwanted spaces in the mapped value is removed.**
- **EXTRACT – Part of the Mapped value can be extracted using the Command.**
  Format: EXTRACT <start value>",”<end value>
  E.g.: EXTRACT 1,5
- **The formatting of the mapped value can be done as follows**
  5%R, 10L

The following functions are available for the FORMAT field, only if the third part of the ID being HEADER/FOOTER.

- **LINK – To extract a field value from some other application**
  Format: LINK”>” <Filename>”>”<Field name>
  E.g.: LINK>CUSTOMER>MNEMONIC.
- **The format condition using the following:**
  E.g.: 10%R, 5L

Back

Match

Matches the string expression from the GIT.MAPPED to a pattern (More information on the pattern matching is available in the universe online help)

If the pattern specified here does not match with the Mapped value then error is recorded in GIT.PROCESSING.DETAILS file. If ERROR.STOP is set to 'YES'/'NO' in the GIT.INTERFACE.OUT table then the process gets stopped by displaying an override condition.

Back

Field Routine

Valid routine name. This routine name will be called to do some specific processing on the field.

E.g. @FLD.RTN (GIT.MAP.VAL.ARR.DUP,GIT.FORMAT.VAL,ERR.MSG)

GIT.MAP.VAL.ARR.DUP - Contains all the Mapped values for the TYPE.

GIT.FORMAT.VAL - Current field's mapped value.

ERR.MSG – Contains Error Message if any

Back

Record Routine

Valid routine name. This routine will be called to do some specific processing on the entire record.

E.g.: **@REC.RTN (**FORMATTED.PRE.MSG)

FORMATTED.PRE.MSG -Contains the Formatted values of all the Field names for the TYPE.

Back

Header Rtn

Routine to process the header part of the output message. This routine is used when there is a formatting record for the Header.ie INTERFACE- TYPE-HEADER.

This routine should be given only in the Header record (i.e. when the third part of the ID being HEADER).

Back

Footer Rtn

Routine to process the footer part of the output message. This routine is used when there is a formatting record for the footer.ie INTERFACE- TYPE-FOOTER

This routine should be attached in the Footer record only (i.e. when the third part of the ID being FOOTER).

Back

GIT Outward Processing Details ENQUIRY

This Enquiry will list the details of the GIT.INTERFACE.OUT uploaded process to Indicate its status.

GIT.INTERFACE.OUT ID has to be supplied as the Selection Criteria, for which the Enquiry will list all the INTERFACE ids that are processed on the current date

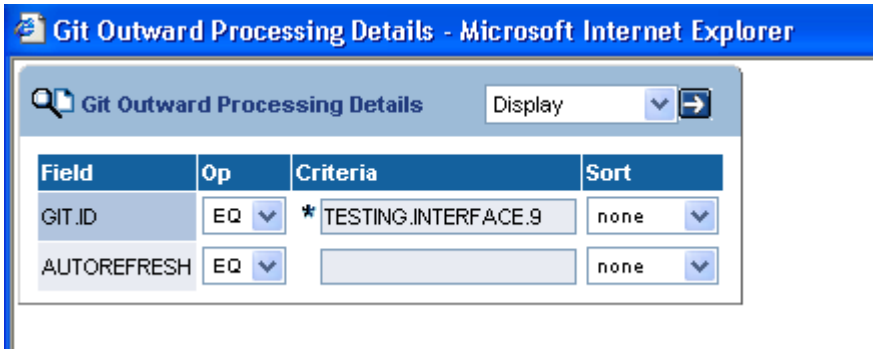
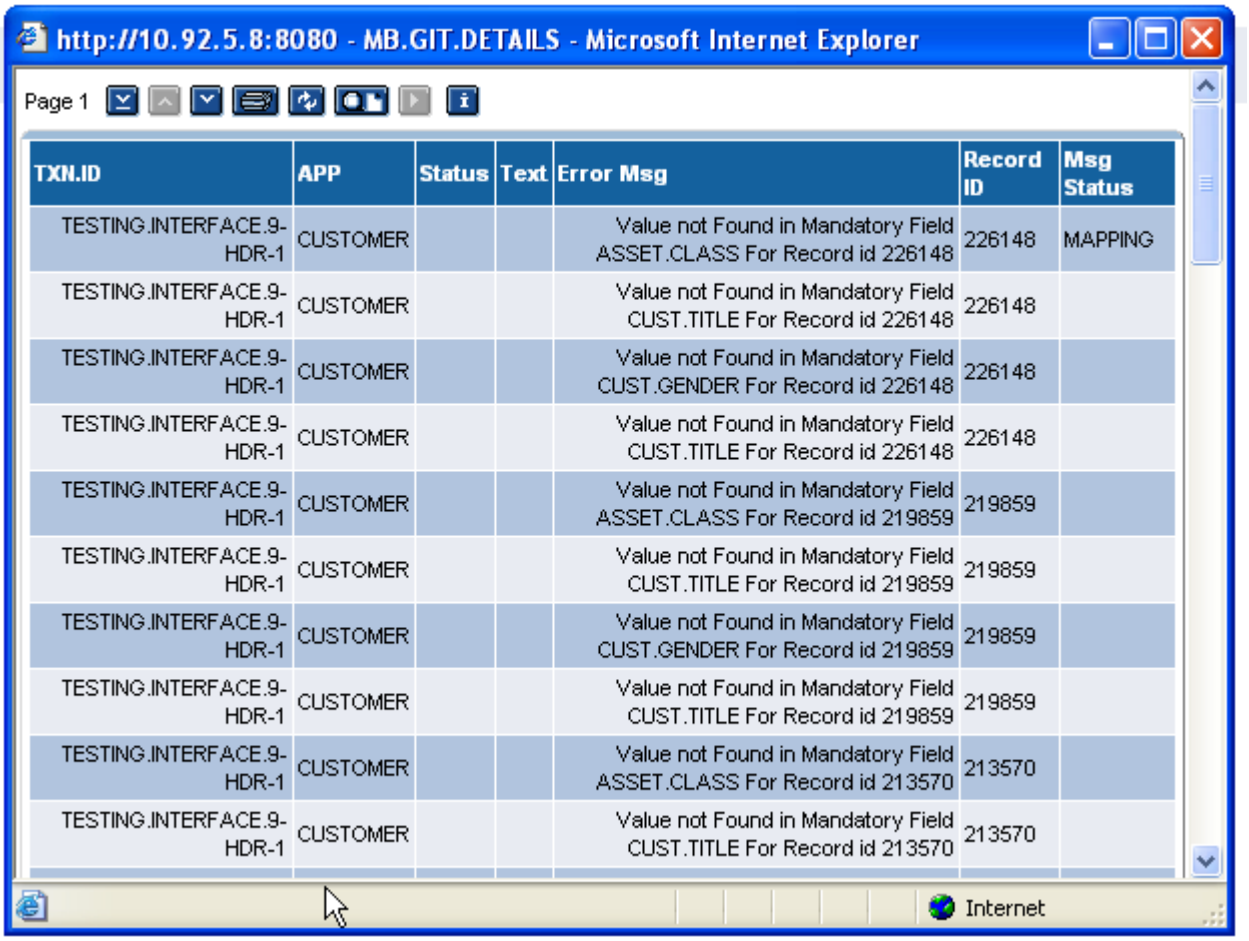


Fig 8 – GIT Outward Processing Details enquiry selection Screen shot

Enquiry displaying the results for the given GIT.INTERFACE.OUT record



TXN.ID	APP	Status	Text	Error Msg	Record ID	Msg Status
TESTING.INTERFACE.9-HDR-1	CUSTOMER			Value not Found in Mandatory Field ASSET.CLASS For Record id 226148	226148	MAPPING
TESTING.INTERFACE.9-HDR-1	CUSTOMER			Value not Found in Mandatory Field CUST.TITLE For Record id 226148	226148	
TESTING.INTERFACE.9-HDR-1	CUSTOMER			Value not Found in Mandatory Field CUST.GENDER For Record id 226148	226148	
TESTING.INTERFACE.9-HDR-1	CUSTOMER			Value not Found in Mandatory Field CUST.TITLE For Record id 226148	226148	
TESTING.INTERFACE.9-HDR-1	CUSTOMER			Value not Found in Mandatory Field ASSET.CLASS For Record id 219859	219859	
TESTING.INTERFACE.9-HDR-1	CUSTOMER			Value not Found in Mandatory Field CUST.TITLE For Record id 219859	219859	
TESTING.INTERFACE.9-HDR-1	CUSTOMER			Value not Found in Mandatory Field CUST.GENDER For Record id 219859	219859	
TESTING.INTERFACE.9-HDR-1	CUSTOMER			Value not Found in Mandatory Field CUST.TITLE For Record id 219859	219859	
TESTING.INTERFACE.9-HDR-1	CUSTOMER			Value not Found in Mandatory Field ASSET.CLASS For Record id 213570	213570	
TESTING.INTERFACE.9-HDR-1	CUSTOMER			Value not Found in Mandatory Field CUST.TITLE For Record id 213570	213570	

Fig 9 – GIT Outward Processing Details enquiry Output Screen shot

Inward Processing:

Inward processing is exact opposite to outward processing where stream of ASCII data is accepted either in the form of text files or in the form of xml files. These files are parsed and mapped by the rules defined in the mapping definition.

The following flow chart depicts the phases of the GIT INWARD process:

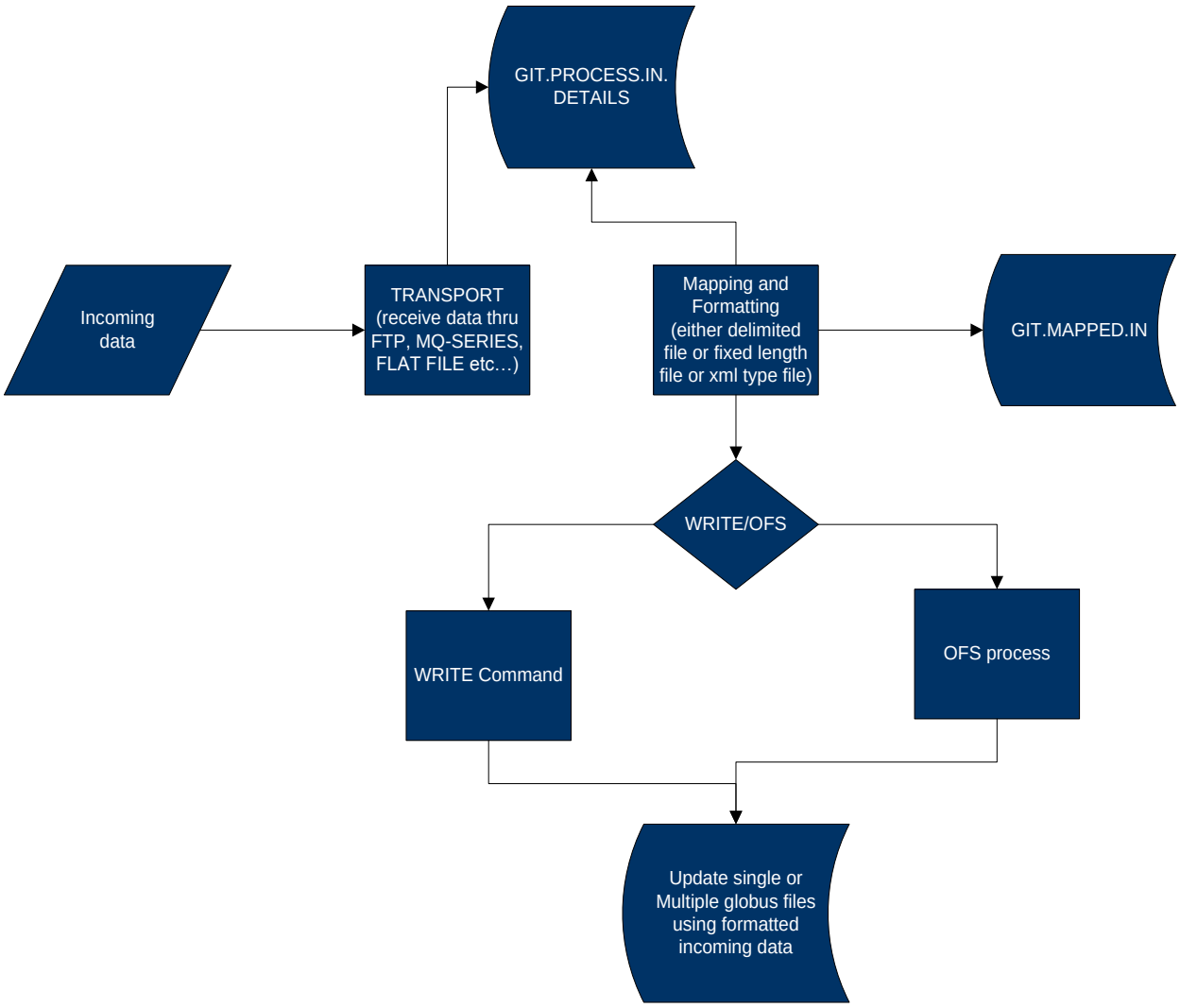


Fig 10 – GIT INWARD Diagrammatic Representation

The Inward Processing of GIT includes the following parameter files:

GIT.ITNERFACE.IN

GIT.MAPPING.IN

GIT.TRANSPORT

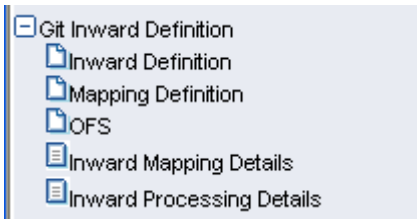


Fig 11 – GIT INWARD Menu

Description of the above files and their fields are as follows

GIT.INTERFACE.IN,MB.DEFINE

Main table for Inward Processing of GIT.

About This File…

GIT.INTERFACE.IN is the main table that contains the parameters for defining an inward interface (data selection, mapping and transport). To invoke the interface, the Interface record is verified which in turn calls the GIT.INTERFACE.IN.RUN routine.

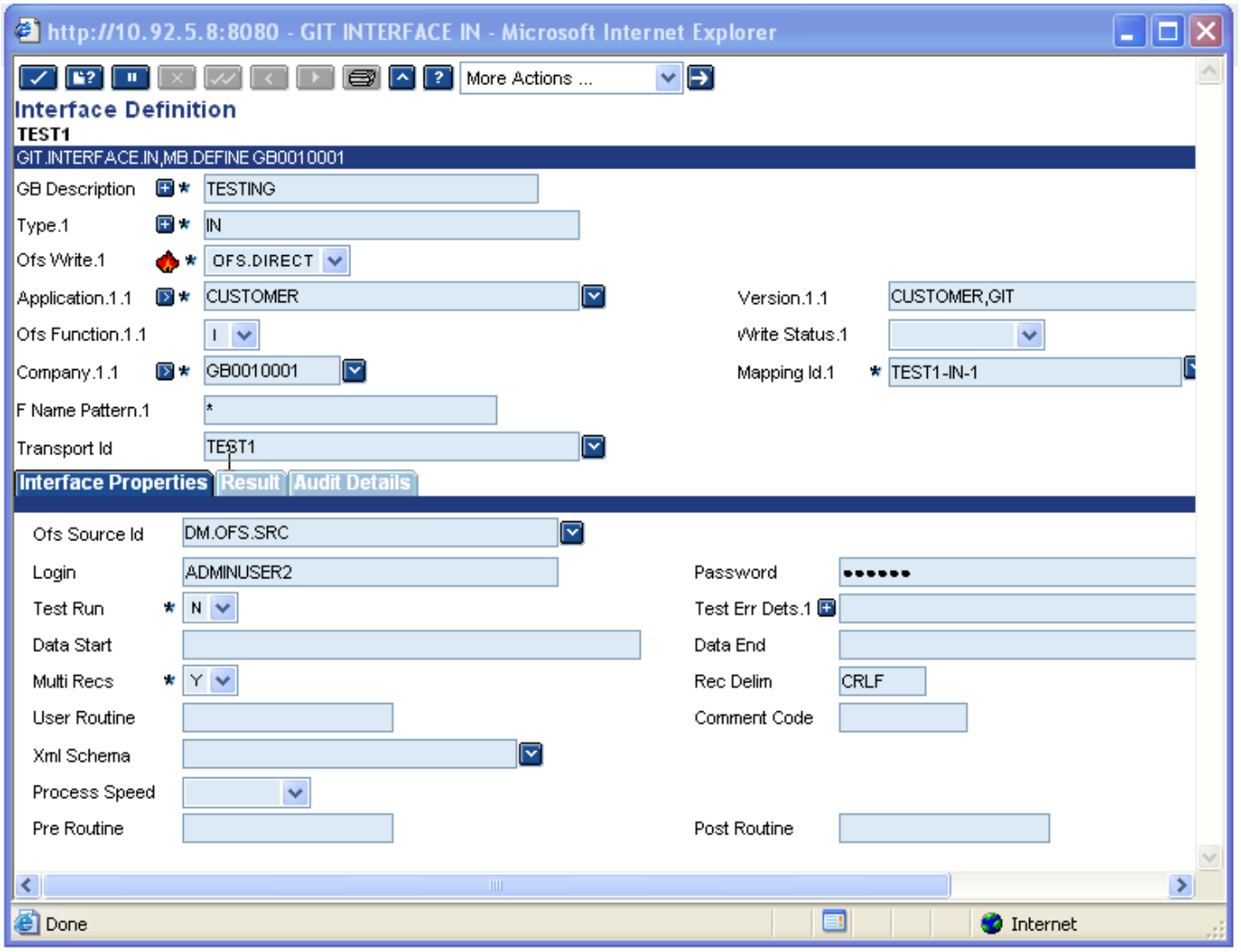


Fig 12 – GIT.INTERFACE.IN,MB.DEFINE Screen shot

GIT.INTERFACE.IN,MB.DEFINE - Field Description and Population Rules…

Additional Description for SETUP Fields…

@ID

Name of the Interface.1-20 Alphanumeric characters.

Back

GB Description

Contains the description of the Interface.1-50 Alpha numeric characters.

Mandatory field

Back

Type

Interface Type.15-40 Alpha numeric characters.

Mandatory field.

Back

OFS Write

Denotes the method of updating T24.

E.g.: OFS.DIRECT/WRTE/OFS

- **OFS.DIRECT – OFS used in Online. Set the SOURCE.TYPE field in OFS.SOURCE to T24.This is being achieved by calling OFS.GLOBUS.MANAGER to update the applications within T24.**
- **WRITE – Does a direct write of the data into T24 Application, without checking any validations like checkfields, cross validations etc.**
- **OFS – Writes the Message in OFS FORMAT into the IN.DIR Of OFS.SOURCE. Set the SOURCE.TYPE field in OFS.SOURCE to BATCH.**

Mandatory Field

Back

Application

Valid T24 application.

E.g. : CUSTOMER

Mandatory field.Valid Standard selection Id.

Back

Version

VERSION name when OFS is used.

If OFS/OFS.DIRECT is selected, then input is allowed.

Back

Ofs Function

Valid function. This function is used when OFS.WRITE is set to OFS.

Possible options are:
I/A/R/D
I – Input
A – Authorise
R – Reverse
D – Delete

Back

Write Status

Used only when the OFS.STATUS field is set as WRITE

Possible options are

IHLD/INAU/LIVE/LIVE.WRITE/LOCK.WRITE/LIVE.UPDATE

- **IHLD, INAU – The data feed is updated as an IHLD status record in the application defined in the APPLICATION field.**
- **LIVE – The data feed is updated as a LIVE record in the application defined in the APPLICATION field.**
- **LOCK.WRITE – The record onto which the data feed should be updated is locked before being written and updated as a LIVE record.**

Back

Company

Valid COMPANY id.
E.g.: US0010001, DE0010001 etc

Mandatory Field

Back

Mapping Id

Consists of 3 components. Interface.id – Int.type –Sequence number

Interface.id – Current GIT record id.

Int.type – The current TYPE field value.

Sequence number – Can be any numeric value.

If no value is input then GIT will create a default GIT.MAPPING.OUT record in IHLD.

Mandatory Field

Back

F Name Pattern

Name of the source file from which data is to be taken to update

T24.The file is taken from the IN directory that is specified in the TRANSPORT record.

Possible options are:

- **…<filename>…,<filename>…,…<filename> - Selects all the files with the occurrence of filename**
- *** - Selects the entire IN directory.**

Back

PROCESS.INFO

Processed Information. Displays the number of files mapped and processed for the particular selection. Updated by the System

No Input field.

Back

Transport Id

The id using which the processed details are transported.

Valid id in GIT.TRANSPORT file.

Back

Ofs Source Id

If OFS.WRITE is set to OFS/OFS.DIRECT then this field should be

defined and should be a valid OFS source record.

Back

Login

If OFS.WRITE is set to OFS/OFS.DIRECT then this field should be

defined and should be a valid USER id defined in the USER application.

Back

Password

Password for LOGIN. Once the value is entered it gets encrypted.

Back

Data Start

The position where the Header information ends, and the DATA part (i.e.) (the data that should be updated into T24) starts is given here.

Possible options are:

- **Data start position – Number from where the data starts.**
- **Format:"LINE.NO-"<line no> - The line no from where the data starts E.g.: LINE.NO-5 – Data starts from the fifth line.**
- **< Any text> - GIT will identify the text in the Message, and will consider that as the Start of DATA.**

Back

Data End

The position where the Footer starts and DATA part (i.e.) the data that should be updated into T24 ends is given in this field.

Possible options are:

- **Data start position – Number from where the footer starts.**
- **Format:"LINE.NO-"<line no> - The line no from where the Footer starts E.g.: LINE.NO-5 – footer starts from the fifth line.**
- **<Any text> - GIT Will identify the text in the Message and will consider that as the End of DATA or Start of Foot notes.**

Back

Multi Recs

If the Incoming data feed contains Multi records to process then set this field to 'YES'

Mandatory Field

Default Value is 'Yes'

Back

Rec Delim

If the MULTI.RECS field is set to YES then input to this field is Mandatory.
This field is used to specify the delimiter between the records.

E.g.: CRLF (for line feed), #, * etc

Any valid delimiter except invalid special characters are allowed

Back

User Routine

Valid routine name.

If a routine is given here the overall feature if GIT will be skipped. Care should be taken that all the updates will be done by the user routine.

Back

Comment Code

Special characters to tell the system that the lines in the message that are preceded by the character that is specified in this field are just comments in the message.

When a character is defined here and when the system encounters the character in the message then that line will not be processed
E.g.: #, etc

Back

Xml Schema

Used to send and receive XML messages

The ID of the GIT.XML.TABLE record will be specified in this field.

Back

Process Speed

Used for speedy generation of output.

FUTURE USE

Accepts values “THROUGH” or “QUICK”.

Back

Pre Routine

Routines can be attached before the records are processed.

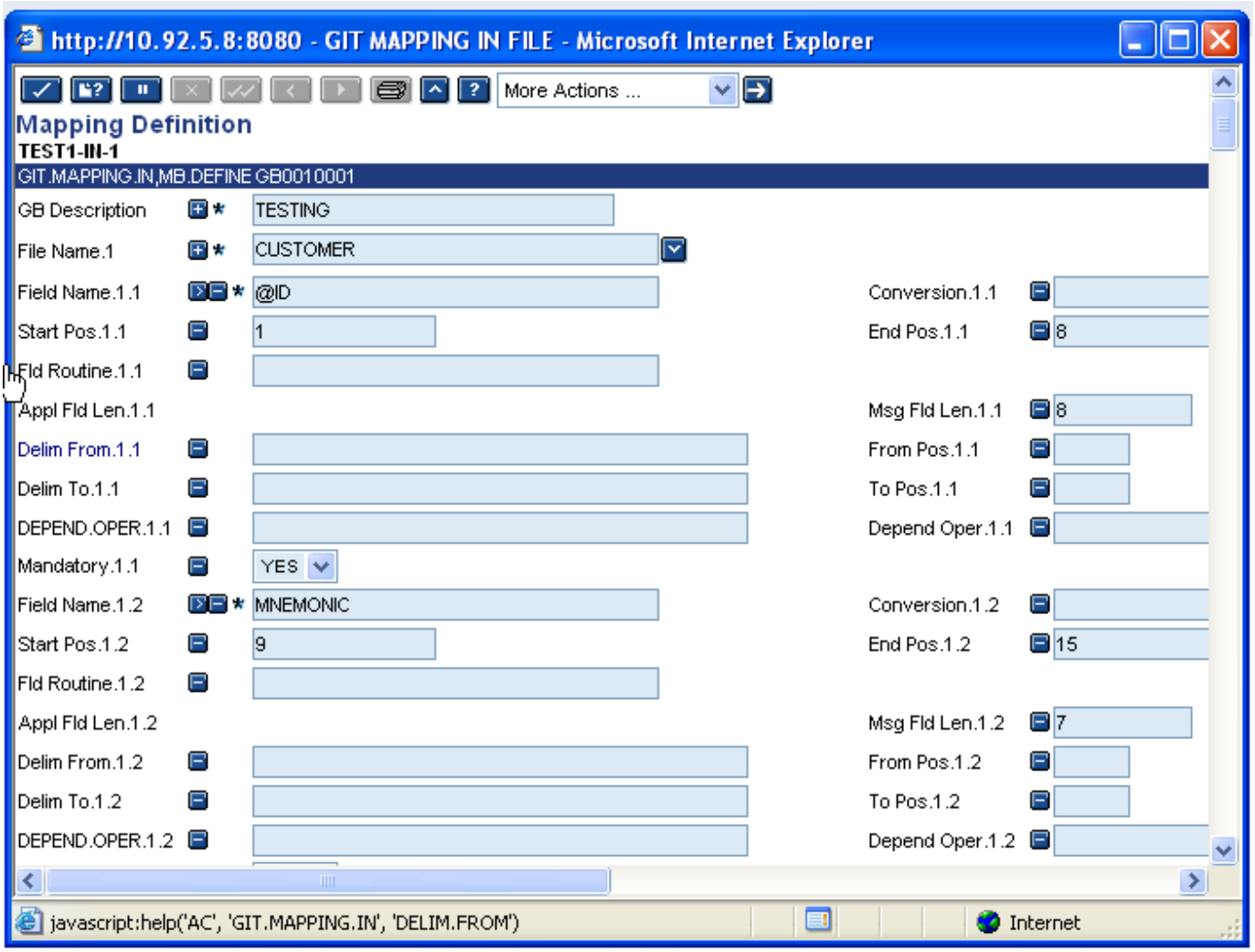
Pre Routine

Routines can be attached after the records are processed.

GIT.MAPPING.IN,MB.DEFINE

About This File…

This version is used to map the required application fields with the values in the file containing the actual data. The mapping is done based on the field position rather than any delimiters.



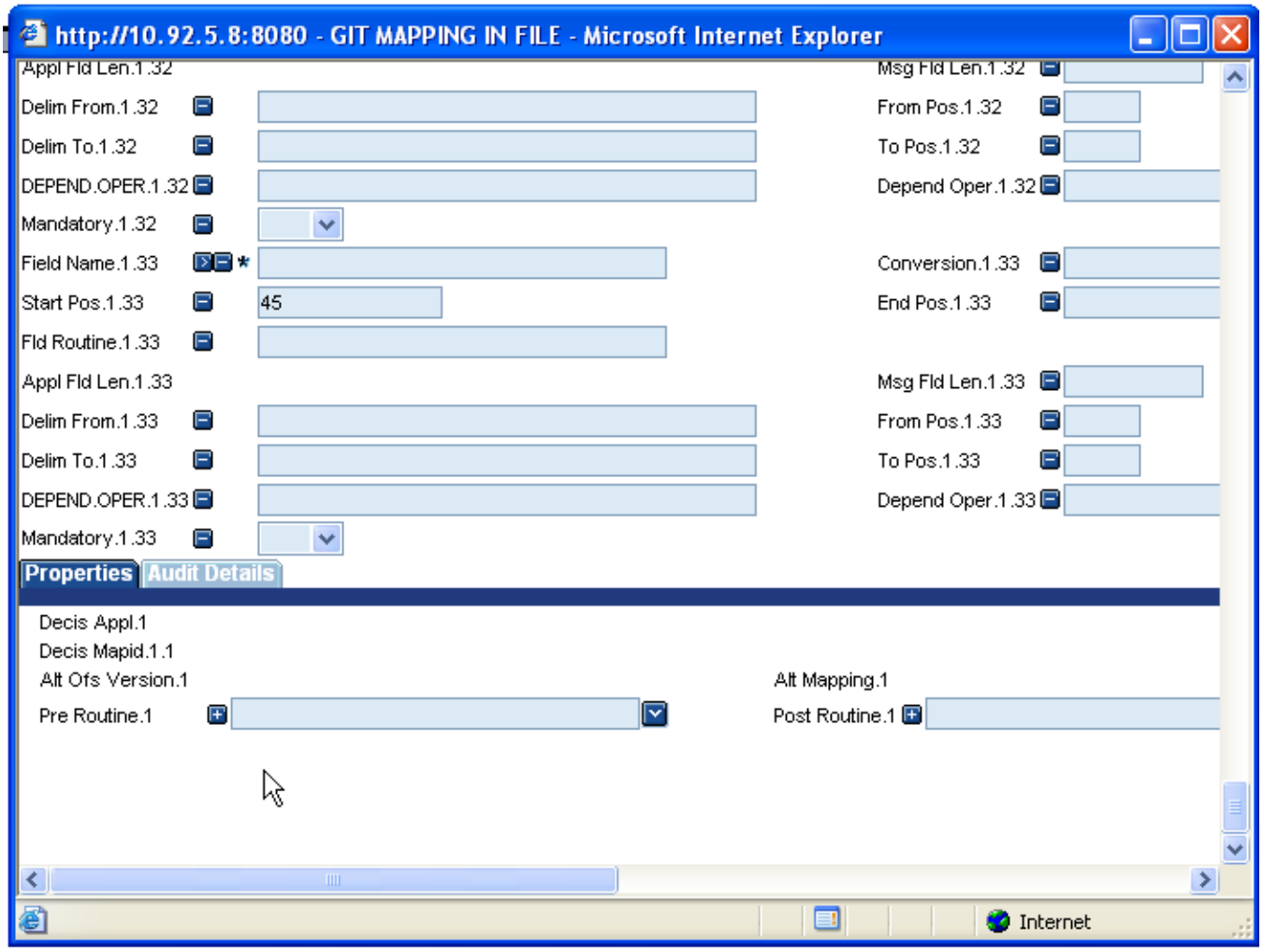


Fig 13 – GIT.MAPPING.IN,MB.DEFINE Screen shot

GIT.MAPPING.IN,MB.DEFINE - Field Description and Population Rules…

Additional Description for SETUP Fields…

@ID

The ID is made of three parts.
Name of the interface”-“Type “-“number.
Name of the Interface”-“Type “-“ “INDEX”

First and second part will be validated against the ID of the GIT.INTERFACE.IN record and the TYPE within the GIT.INTERFACE.IN record. Once the GIT.INTERFACE.IN record is created and the mapping id is supplied, then this record will be created in IHLD.
Name of the Interface”-“Type “-“ “INDEX” record will help in selecting the specific Mapping record depending on the values of one or more fields.

Back

GB Description

Contains the description of the Interface.1-50 Alpha numeric characters.

Mandatory field

Back

File Name

Valid Application name, using which the incoming data will be updated or created or deleted using OFS or WRITE.

Mandatory field.

Back

Field Name

Valid fields from the Standard Selection record of the application given in the APPLICATION field.

The first field must be @ID or ID.NEW. The other fields must be from standard selection record

Mandatory Field

Back

Conversion

Used to convert the mapped field value. The following functions are available.

- **A routine can be used to do more calculations or conversions and to set error condition.**
  @ <Routine name>(GIT.MAP.VALUE, CONV.ERR).
  GIT.MAP.VALUE – Incoming & Outgoing argument
- **Formatting the Mapped value can be done as follows**
  FMT, <formatting condition>
  E.g.: FMT,5R
  FMT,10%L
- **ICONV, OCONV can be used as follows. ICONV,<conversion condition>**
- **APPEND:**
  To add a String value or a previously defined field value to the existing field value APPEND can be used in the following format.
  APPEND "Sample Test"
  APPEND <previously defined field name>
- **INS – If previously defined field name's value should be used in the current field name then INS is used.**

Format: INS <Previously defined field name>

- **ADD, SUB, MUL, DIV – If two field values (numeric) needed to be added, subrated, multiplied, divided to get the current field's value then use the above**
  Conversion values in the following format
  <Operation> <fld1>,<fld2>
  Operation – ADD/SUB/MUL/DIV
  Fld1, Fld2 – Previously defined field names
- **LINK, LINKI, LINKV – If a value has to be taken from some other Application then LINK comes into play.**
  Format 1: LINK">" <Application> " > "<Previously defined field name> ">" <field name from the Application>
  Previously defined field name – This should be the Id of the Application Linked.
  E.g.; LINK>CUSTOMER>ID.VAL>MNEMONIC

  Format 2: LINK">" <Application> " > " < field name from the Application>
  Here the ID of the record in the Application will be the Current field name's value
  E.g.; LINK>CUSTOMER>MNEMONIC.

  Format 3: If the field fetched from the Linked Application is an IDescriptor field then use the following format.
  LINKI ">"<Application name> ">" <IDescriptor field name>
  E.g.: LINKI>CUSTOMER>CU.SAMPLE where CU.SAMPLE is the IDescriptor field.

  Format 4: If LINKV is used then when the value got using the link option is null then the original value used for link will be returned.
  E.g.: LINKV>CUSTOMER>NAME.2 –If the NAME.2 field is null then the Current field name's value will be returned.
- **CONCAT – When more than one field name's field values need to be concatenated**
  Then use the CONCAT in the following format
  CONCAT <fld1>","<fld2>","<fld3>
  E.g.: CONCAT field1,field2,field3
  Field1, field2, field3 – Previously defined field names.
- **EXTRACT – Used to extract a part of field value from the mapped value**
  Format: EXTRACT <start position>","<end position>
  E.g.: EXTRACT 1,4
- **ABS – Absolute Mapped value**
  Format: ABS
- **FIELD – used to extract the value in the VMth position and the SMth position**
  Format: FIELD VM or @VM ","<VM position>
  E.g.: FIELD VM,3 – will extract the field name's value's 3rd VM value.
  Format 2: FIELD SM or @SM ","<VM position>","<SM Position>
  E.g.: FIELD SM,3,2 – will extract the 2nd SM value from the 3rd VM field value.
- **CHANGE – Used to change the occurrences of a string or a character to another character.**
  Format: CHANGE <From. Value>"," <To. Value>
  E.g.: CHANGE #,VM
- **IF – IF condition can be used in the following format**
  Format: IF <fieldname1> <operator> <fieldname2> <fieldname3> <fieldname4>
  Operator – EQ, GT, LT, LE, GE, NE

If the IF condition is successful then fieldname3 value will be used else fieldname4 will be used.
Fieldname1, fieldname2, fieldname3, and fieldname4 – previously defined field values.

- **TRIM – Used to trim the unwanted spaces.**
  Format: TRIM.

Back

Start Pos

Indicates the position from where the data for the field defined in FIELD.NAME field should be picked.
If start position is given then the end position must be defined. The field DELIM.FROM and DELIM.TO will be NOINPUT.

Back

End Pos

Indicates the position where the data of the field defined in FIELD.NAME field ends.

If start position is given then the end position must be defined. The field DELIM.FROM and DELIM.TO will be NOINPUT.

Back

Fld Routine

Valid routine name. If any additional manipulation needs to be done then field level routine can be attached.

E.g.**: abc.rtn (GIT.MAP.VALUE)**

GIT.MAP.VALUE-contains the field value to be manipulated.

Back

Appl Fld Len

Populated from the Standard selection record**.** Updated by System

No Input field

Back

Msg Fld Len

Actual length of the field in the message.

Populated by the system depending on the details in START.POS, END.POS

Back

Multi Single

Populated from the Standard selection record

No Input field

Back

Delim From

Delimiter to denote where data starts for the field defined in FIELD.NAME.

Any valid character can be given, provided the character is used to split the fields in the message.

Back

From Pos

The position of the field start in the message is given here

E.g.: 123455#samplefield#123abc#23455.00
If the data is like this
Then for extracting the second value i.e. samplefield from the message the definition would be
DELIM.FROM ---#
FROM.POS ----1

Back

Delim To

Delimiter to denote where data ends for the field defined in FIELD.NAME.

Any valid character can be given, provided the character is used to split the fields in the message.

Back

To Pos

The position of the field end, in the message is given here

E.g.: 123455#samplefield#123abc#23455.00
If the data is like this
Then for extracting the second value i.e. samplefield from the message the definition would be
DELIM.FROM ---#
FROM.POS ---- 1

DELIM.TO----- #
TO.POS----- 2

Back

Depend On

If the current field value has to be checked against some previously defined fields then this check can be done using this field

Possible options are:

- **If more than one field names have to be checked then they can given using the following format:**
  <fieldname1>"#"<fieldname2>"#"<fieldname3>
  Fieldname1, fieldname2, fieldname3 – Previously defined fields.

Back

Depend Oper

The conditional operators for the fields defined in DEPEND.ON are given here
If more than one field is considered in DEPEND.ON field then the corresponding operator will also be in the same format as that of DEPEND.ON. E.g.: EQ#LT#GT

The number of operators should be equal to the number of fields defined in DEPEND.ON field.

Back

Depend Value

The values for the DEPEND.ON fields are given in this field.

- **If more than one value should be checked for each field then they can given using the following format**
  <Fld1val1>"-"<Fld1Val2>"-"<Fld1val3>'#'<Fld2val1>"-
  "<Fld2val2>'#'<Fld3val1>
- **If the DEPEND.OPER field used for a field is "RG" then supply only 2 values separated by "-"**

If the condition specified using the DEPEND.ON, DEPEND.OPER, DEPEND.VALUE fields is not met then the data for the associated field name in the FIELD.NAME field will not be updated in the file specified in the APPLICATION field.

Back

Mandatory

Holds the value YES/NO

Default value is NO.

If set as 'Yes' and if the value of the FIELD.NAME field is null then the data for the associated field name will not be updated in the file specified in the APPLICATION field.

Back

Decis Mapid

Valid mapping id from GIT.MAPPING.IN record is given here.

Back

Decis Appl

Value from the APPLICATION field should be given here.

The field can be Input only when the third part of the Mapping ID is "INDEX".

Back

Decis Field

Valid FIELD.NAME Value from the APPLICATION field should be given here.

The field can be Input only when the third part of the Mapping ID is "INDEX".

The field name given in the DECIS.FIELD should be a valid mapping FIELD.NAME (defined in the current record (i.e.) Interface. name-Type-INDEX).

Back

Decis Oper

No Input field

Back

Decis Value

The value of the DECIS FIELD is given here

For example

If the DECIS.FIELD is set to EVENT.TYPE and if the value of EVENT.TYPE is "BOND" then INTF-TYPE1-2 mapping record should be used and if EVENT.TYPE value is "CASH" then INTF-TYPE2-3 mapping record should be used then the condition is given as follows

DECIS.FIELD…. EVENT.TYPE

DECIS.OPER…...

DECIS.VALUE... BOND

DECIS.MAPID…. INTF-TYPE1-1

DECIS.FIELD…. EVENT.TYPE

DECIS.OPER..

DECIS.VALUE…. CASH

DECIS.MAPID. …INTF-TYPE2-3

The field will be an Input field, only when the third part of the ID is “INDEX”.

Back

Pre Routine

Valid routine name.

This routine can be used to do some format of the raw incoming data or for some other processing.

E.g.: @pre.rtn (RAW.MSG, ERR.MSG)

RAW.MSG – Message got from the feed

Back

Post Routine

Valid routine name.
This routine can be used to process the mapped data.

E.g.: @post.rtn (MAPPED.MSG, APPL, ERR.MSG)

MAPPED MSG – Mapped message that is to be written into T24

Back

Inward Mapping Details ENQUIRY

This Enquiry will list the field level mapping done for the inward processing.

On clicking the menu above the enquiry will be launched automatically as it is a composite screen.

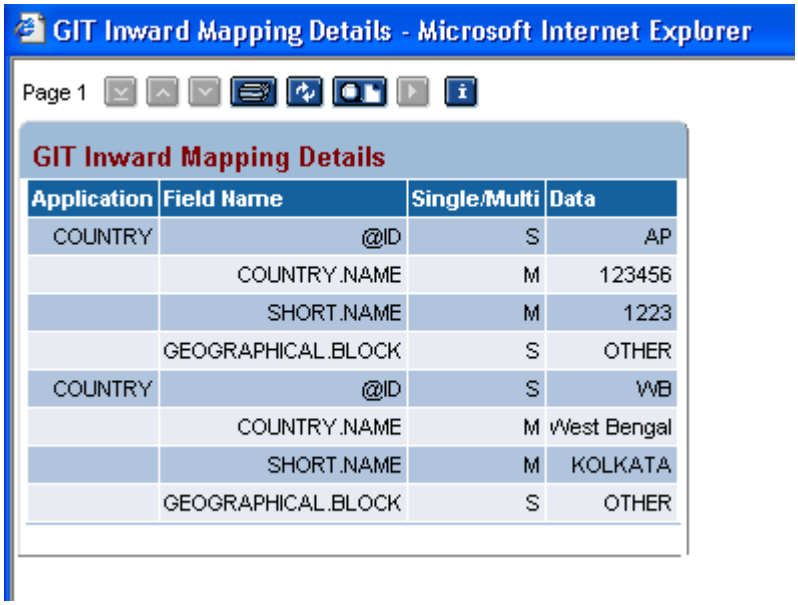
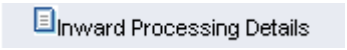


Fig 14 – GIT Inward MAPPING Details enquiry Screen shot

Inward Processing Details ENQUIRY



This Enquiry will list the processing details of the inward process done on that particular day.

GIT.INTERFACE.IN ID has to be supplied as the Selection Criteria, for which the Enquiry will list all the records that are processed on the current date

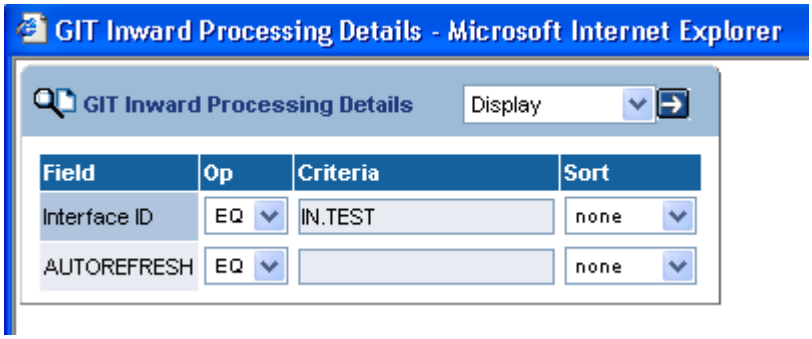


Fig 15 – GIT Inward Processing Details enquiry Selection - Screen shot

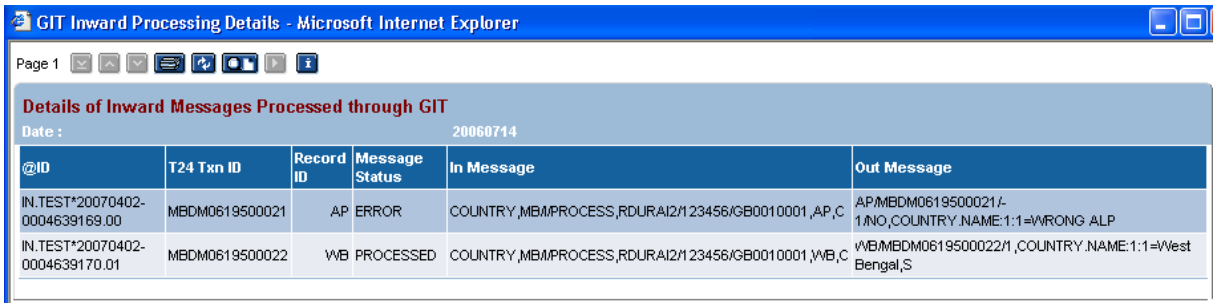


Fig 16 – GIT Inward Processing Details enquiry output Screen shot

XML Processing:

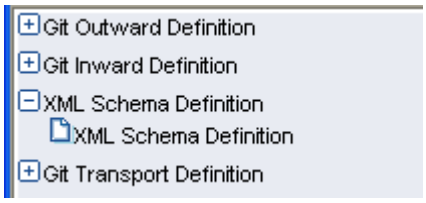


Fig 17 – XML Schema Definition Menu Screen shot

GIT.XML.SCHEMA,MB.DEFINE

About This File…

GIT.XML.SCHEMA can be used to send and receive XML messages using GIT.INTERFACE.OUT and GIT.INTERFACE.IN files respectively. XML schema will be defined in this table. GIT will use this schema to create XML message. GIT.XML.SCHEMA record @ID should be attached to GIT.INTERFACE.OUT or GIT.INTERFACE.IN.

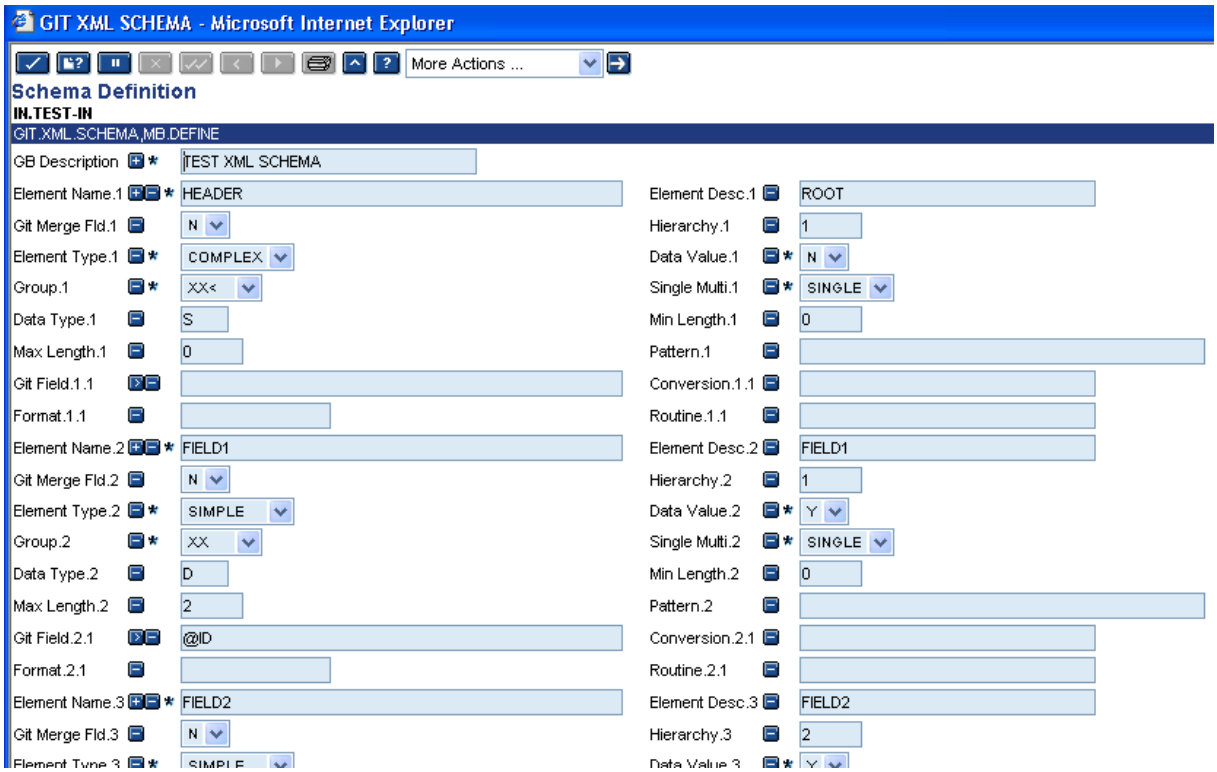


Fig 18 – GIT.XML.SCHEMA,MB.DEFINE Screen shot

GIT.XML.SCHEMA,MB.DEFINE - Field Description and Population Rules…

@ID

Valid id of GIT.XML.SCHEMA record.1-50 Alphanumeric characters.

Back

Gb Description

Description of the XML schema record.

Mandatory field

Back

Element Name

XML tag name.1-55 Alphanumeric characters.

Mandatory field

Back

Element Desc

XML tag description.1-35 Alphanumeric characters.

Back

Git Merge Fld

Flag to indicate merging of sub value fields.

If Y, then sub-value fields will be merged within the same tag<TAG>VAL1VAL2VAL3</TAG>.If N, each value will be put in a separate tag. <TAG>VAL1</TAG>

<TAG>VAL2</TAG>

<TAG>VAL3</TAG>

Back

Hierarchy

Sequence number 0…999 for tag hierarchy. This field value is used in displaying the Schema details

Back

Element Type

SIMPLE or COMPLEX

Tags like <TAG>VALUE</TAG> are simple. Tags like <TAG><TAG1>VALUE</TAG1></TAG> are complex, whereas<TAG1>inside <TAG> is simple

Mandatory field.

Back

Data Value

The data of the GIT mapped field is assigned to this field (as the GIT.FIELD contains a valid mapping field from the Mapping record id in the MAPPING field)

The following are the possible ways of supplying values to this field

- **Static values can be passed using the following format**
  Format: "<value>"
- **T24 common variables can be included:**
  Format: "!"<Common variable>
  E.g.:! ID.COMPANY
- **SPACE – If SPACE is given then the data will have one space added**
  Format: SPACE.
- **CRLF – A Line is fed in the data**
  Format: CRLF
- **FM – Field marker is inserted in the data**
  Format: FM
- **If the Mapped field name is given then the value of the field is taken for the Data value field.**

Mandatory Field

Back

Group

Branch

XX, <XX>, XX<, XX-, XX>.
Exactly the same meaning as our famous template multi value concept, except for the fact that it is applied to XML tags. XX – doesn't contain end tag. (I.e.) <TAG>VALUE. This is used in case of starting tags of XML like <TAG> XML version 1.0 <XX> - SIMPLE tag. Contains value and contain no child tags. XX< - COMPLEX tag, starting of a group XX – COMPLEX tag, within a group XX> - COMPLEX tag, end of group

Mandatory Field

Back

Single Multi

Flag to indicate if the tag should be repeated multi times or not

SINGLE, MULTI. If SINGLE, then tag is not repeated again. <TAG>

<TAG1>VALUE</TAG1> </TAG> If MULTI, then tag is repeated

multiple times <TAG> <TAG1>VALUE</TAG1>

<TAG1>VALUE1</TAG1> </TAG>

Mandatory Field

Back

Data Type

Data type of the field

Eg: A, D, S etc

Back

Min Length

Minimum length of the field.

Back

Max Length

Maximum length of the field.

Back

Pattern

Pattern Match (future Use)

Back

Git Field

GIT field name from GIT.MAPPING.OUT id in the field MAPPING of GIT.XML.SCHEMA.

Back

Conversion

The following are the conversion that can be applied on the data

- **A routine can be used to do more calculations or conversions and to set error condition.**
  @ <routine name>(GIT.MAP.VALUE, CONV.ERR).
  GIT.MAP.VALUE – Incoming & Outgoing argument
- **Formatting the Mapped value can be done as follows**
  FMT, <formatting condition>
  E.g.: FMT,5R
  FMT,10%L
- **ICONV, OCONV can be used as follows. ICONV,<conversion condition>**
- **APPEND:**
  To add a String value or a previously defined field value to the existing field value APPEND can be used in the following format.
  APPEND "Sample Test"
  APPEND <previously defined field name>
- **INS – If previously defined field name's value should be used in the current field name then INS is used.**
  Format: INS <Previously defined field name>
- **LINK, LINKI, LINKV – If a value has to be taken from some other Application then LINK comes into play.**
  Format 1: LINK">" <Application> " > "<Previously defined field name> ">"
  <field name from the Application>

Previously defined field name – This should be the Id of the Application linked.
E.g.; LINK>CUSTOMER>ID.VAL>MNEMONIC

Format 2: LINK”>” <Application> “ > “ < field name from the Application>
Here the ID of the record in the Application will be the Current field name’s value
E.g.; LINK>CUSTOMER>MNEMONIC.
Format 3: If the field fetched from the Linked Application is an IDescriptor field then use the following format.
LINKI ”>”<Application name> “>” <IDescriptor field name>
E.g.: LINKI>CUSTOMER>CU.SAMPLE where CU.SAMPLE is the IDescriptor field.
Format 4: If LINKV is used then when the value got using the link option is null then the original value used for link will be returned.
E.g.: LINKV>CUSTOMER>NAME.2 –If the NAME.2 field is null then the Current field name’s value will be returned.

- **CONCAT – When more than one field name’s field values need to be concatenated then use the CONCAT in the following format**
  CONCAT <fld1>”,”<fld2>”,”<fld3>
  E.g.: CONCAT field1, field2, field3
  field1, field2, field3 – Previously defined field names.
- **EXTRACT – Used to extract a part of field value from the mapped value**
  Format: EXTRACT <start position>”,”<end position>
  E.g.: EXTRACT 1,4
- **ABS – Absolute Mapped value**
  Format: ABS
- **FIELD – used to extract the value in the VMth position and the SMth position**
  Format: FIELD VM or @VM “,”<VM position>
  E.g.: FIELD VM, 3 – will extract the field name’s value’s 3rd VM value.
  Format 2: FIELD SM or @SM “,”<VM position>”,”<SM Position>
  E.g.: FIELD SM, 3,2 – will extract the 2nd SM value from the 3rd VM field value.
- **CHANGE – Used to change the occurrences of a string or a character to another character.**
  Format: CHANGE <From. Value>”,” <To. Value>
  E.g.: CHANGE #, VM
- **IF – IF condition can be used in the following format**
  Format: IF <fieldname1> <operator> <fieldname2> <fieldname3> <fieldname4>
  Operator – EQ, GT, LT, LE, GE, NE
  If the IF condition is successful then fieldname3 value will be used else fieldname4 will be used as the mapped value.
  Fieldname1, fieldname2, fieldname3, and fieldname4 – previously defined field values.
- **TRIM – Used to trim the unwanted spaces.**
  Format: TRIM.

Back

Format

Formatting the XML data

E.g.: 5%R, 10L etc

Back

Routine

Routine called in the field level.

Format: **@CALL.ROUTINE (XML.DATA, XML.FLD.ERR)**

XML.DATA – Current field name's data

Back

Schema Dets

The XML schema pattern of the defined fields can be seen in this field

No Input field

Back

Static Element

The XML pattern of the Static values are displayed here

No Input field

Back

Data Element

The XML pattern of the Data values are displayed here

No Input field

Back

Element Delim

Delimiter between the fields in the XML.SCHEMA record.

Back

Pre Routine

Any manipulation on the mapping record can be done using this routine

Format: @XML.PRE.ROUTINE (GIT.MAPPING.OUT.REC, XML.ERR.MSG)
GIT.MAPPING.OUT.REC – Mapped record from GIT.MAPPED.OUT record.

Back

Post Routine

Any manipulation on the Output data can be done here

Format: **@XML.POST.ROUTINE (GIT.OUTPUT.DATA, XML.ERR.MSG)**
GIT.OUTPUT.DATA – Contains the entire data that is to be written onto the Output file.

Back

In Out

Flag to specify Inward or Outward flow of data.

IN or OUT

Back

Mapping

Valid record ID from GIT.MAPPING.OUT

Back

Transport Processing

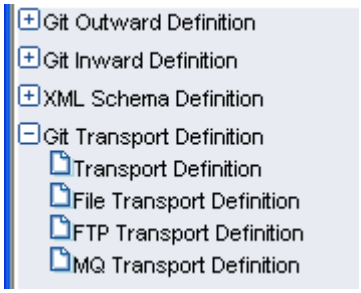


Fig 19 – GIT Transport Definition Menu

GIT.TRANSPORT,MB.GT

About This File…

The IN/OUT transport table links the interface-generated data and the proper Transport interfaces. More than one transport interface can be defined in the GIT.TRANSPORT,MB.GT version and the user can select whether all the transport Interfaces defined will be used or only the first successful transport interface.

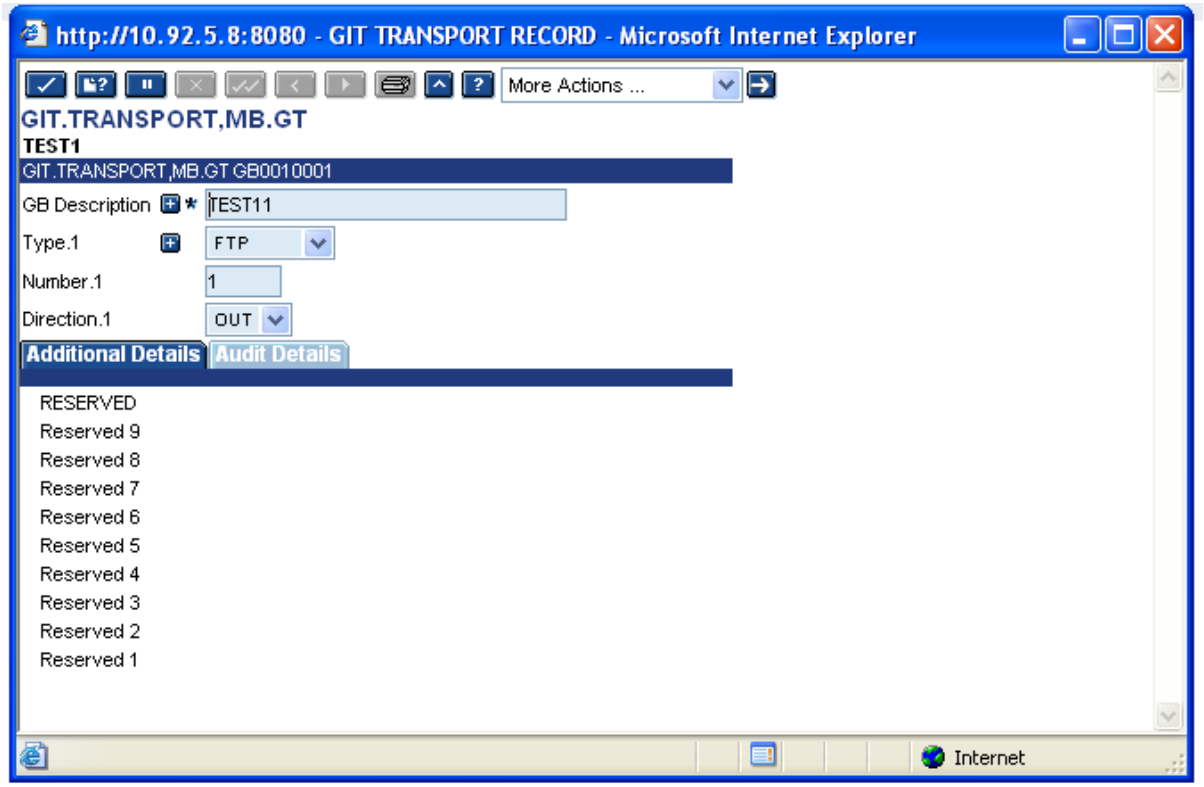


Fig 20 – GIT.TRANSPORT,MB.GT Screen shot

GIT.TRANSPORT,MB.GT - Field Description and Population Rules…

@ID

This is the identification of the GIT.TRANSPORT record.

Back

Description

Description of the transport record.1-35 Alphanumeric characters

Mandatory field.

Back

Type

This field holds the means by which the output/input is taken out/into the System

E.g.: FILE, FTP.

Back

Number

Valid sequence number of the Transport

Back

Direction

The direction of Usage of the Interface. If the data is to be taken out of T24 to external system then it is “OUT”, else if the data is coming as a feed and should be updated onto T24 then the field will hold the value “IN”

IN / OUT are allowed.

Back

GIT.TRANSPORT.FILE,MB.DEFINE

About This File…

This table enables the user to specify the parameters necessary to set up transport via a file in a directory.

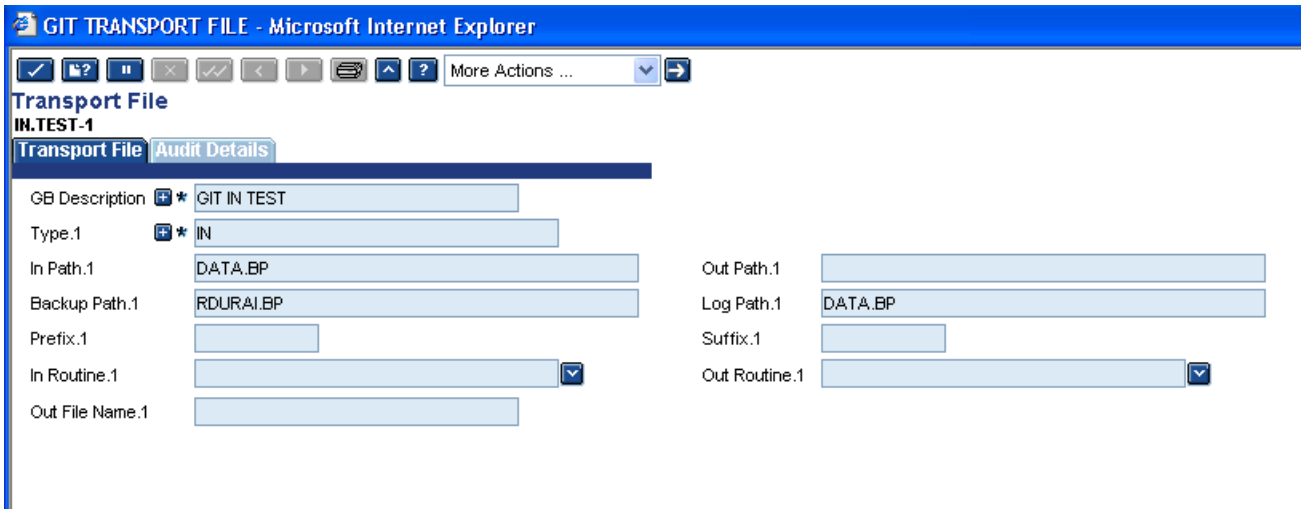


Fig 21 – GIT.TRANSPORT.FILE,MB.DEFINE Screen shot

GIT.TRANSPORT.FILE,MB.DEFINE - Field Description and Population Rules…

@ID

The Id is made up of 2 parts: Interface Id "-" Sequence Number
Transport Id - GIT.TRANSPORT record id
Sequence number – The value in the NUMBER field of GIT.TRANSPORT

Back

Description

Description of the transport.1-35 Alphanumeric characters.

Back

Type

The TYPE value defined in the GIT.INTERFACE.OUT record (having the same id as the first part of the GIT.TRANSPORT.FILE record)

If 'ALL' is specified then all the types defined in GIT.INTERFACE.OUT record will be executed with the details given under this type.

Back

In Path

This field value is used during Inward processing. The path from where GIT has to get the feed is given here. (I.e.)The path from where the data is to be read

Valid file path.

Back

Out Path

This field value is used during Outward processing. The path where GIT has to write the data is given here. (I.e.) The output data Is written onto this path.

Valid file path.

Back

Backup Path

The path where the backup is recorded.

Valid file path.

Back

Log Path

The path where the log is maintained.

Valid file path.

Back

Prefix

This is used during Outward processing. The prefix to be added to the output file name.

E.g. If the prefix is SAMP and if the output file id is 200302323113-3455 then the output file id will be SAMP200302323113-3455

Back

Suffix

This is used during Outward processing. The suffix to be added to the output file name.
E.g.: If the suffix is SAMP and if the output file id is 200302323113-3455 then the output file id will be 200302323113-3455SAMP

Back

In Routine

Any manipulations on the data that is brought into T24 can be done using this routine. This routine will be run during the inward process. (I.e.) when the data comes from the external source.

Valid routine name.
Format: **@in.rtn (output. data, err.msg)**
Output.data – Incoming data from feed

Back

Out Routine

Any manipulations on the data to be sent out can be done using this routine. This routine will be run before the data is sent out.

Valid routine name
Format: @out.rtn (output.data, err.msg)
Output.data - Outgoing data from T24

Back

Out File Routine

The default Outfile name can be overridden by using this field

The following are the options available:

- **DATE – Today's date**
- **TIME - Current time**
- **UNIQUE.NO – Unique sequence number.**
- **SPACE – The number of spaces specified is left in the Id**
  Format: SPACE"("<sequence number>")"
  E.g.: SPACE (14)
  - **GIT.ID – The interface id.**
  - **ID.COMPANY–Common variable ID.COMPANY**
  - **If any static value is required then it can also be given.**

Format: To get the combination of more than one options specified above follow the following format
DATE *UNIQUE.NO* ID.COMPANY
Ex: "SAMP"*UNIQUE.NO –where SAMP is a static value and it should be given within Quotes.

Back

GIT.TRANSPORT.FTP,MB.DEFINE

About This File…

This table enables the User to specify the parameters necessary to set up transport via FTP.
The FTP transfer is possible only for Operating system being UNIX

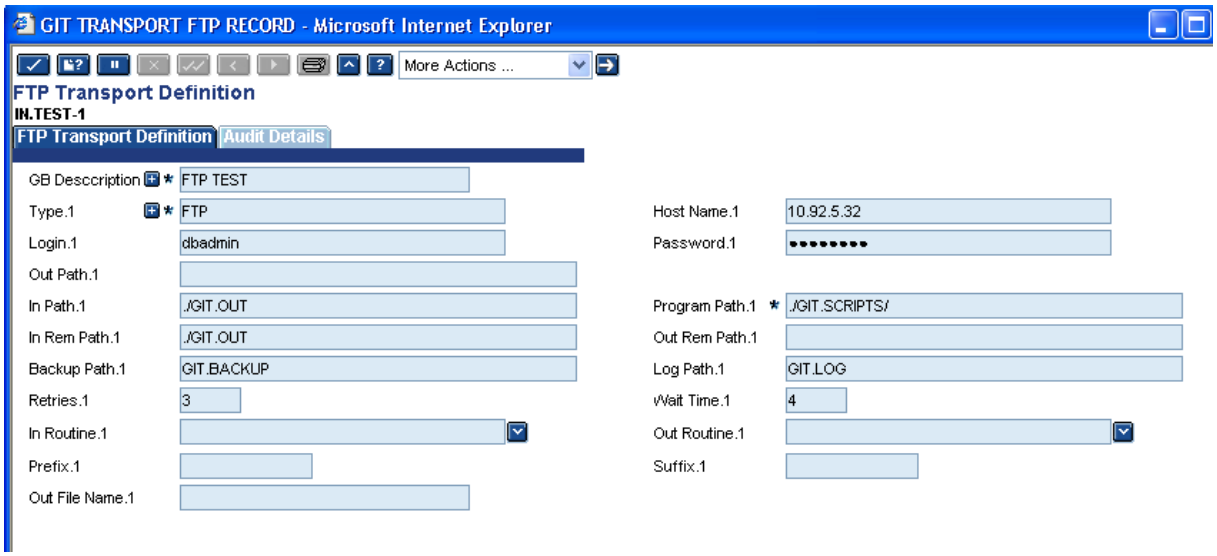


Fig 22 – GIT.TRANSPORT.FTP,MB.DEFINE Screen shot

GIT.TRANSPORT.FTP,MB.DEFINE - Field Description and Population Rules…

@ID

The Id is made up of 2 parts: Interface Id "-" Sequence Number
Interface Id - GIT.TRANSPORT record id
Sequence number – The value in the NUMBER field of GIT.TRANSPORT

Back

GB Description

Description of the transport.1-35 Alphanumeric characters.

Back

Type

The TYPE value defined in the GIT.INTERFACE.OUT record (having the same id as the first part of the GIT.TRANSPORT.FILE record)

Mandatory field

If 'ALL' is specified then all the types specified in GIT.INTERFACE.OUT table will be executed with the details given under this type.

Back

Host Name

The Host id from where the data is to be taken/sent

Back

Login

The Login id of the Host Id.

Back

Password

The valid password for the Login ID

Back

Program Path

The field's value is used both in the case of outward and inward processing. This field contains the path where the ftp.sh file is present.

The exact path of the ftp.sh file is required.
Eg:If the file resides in GIT.SCRIPTS directory, which is under .run directory Then the path will be ./GIT.SCRIPTS/

Back

In Routine

Any manipulations on the data that is brought into T24, can be done using this routine. This routine will be run during the inward process (i.e.) when the data comes from the external source.

Valid routine name.

Format: **@in.rtn (output. data, err.msg)**
Output. data – Incoming data from feed

Back

Out Routine

Any manipulations on the data to be sent out can be done using this routine. This routine will be run before the data is sent out.

Valid routine name
Format: @out.rtn (output.data, err.msg)
Output. data - Outgoing data from T24

Back

In Path

This field value is used during Inward processing. The path from where GIT has to get the feed is given here. (I.e.) The path from where the data is to be read. The path should be in the current working area.

The data is taken from the IN.REM.PATH and is put here by FTP and from here GIT processes the feed.

Valid file path.

Back

Out Path

This field value is used during Outward processing. The path where GIT has to write the data is given here. (I.e.) The output data is written onto this path.
This path should be in the current working area.

The data is taken from the OUT.PATH and is put in the OUT.REM.PATH by the script.

Valid file path.

Back

In Rem Path

The path in the Remote server from where the data has to be taken is given in this field. This field is used during Inward processing.

Valid file path

Back

Out Rem Path

The path in the Remote server where the data has to be written is given in this field. This field is used during Outward processing.

Valid file path

Back

Backup Path

The path where the backup is recorded.

Valid file path.

Back

Log Path

The path where the log is maintained.

Valid file path.

Back

Prefix

This is used during Outward processing. The prefix to be added to the output file name.

Eg:If the prefix is SAMP and if the output file id is 200302323113-3455 then the output file id will be SAMP200302323113-3455

Back

Suffix

This is used during Outward processing. The suffix to be added to the output file name.
E.g.: If the suffix is SAMP and if the output file id is 200302323113-3455 then the output file id will be 200302323113-3455SAMP

Back

Retries

Used during both inward and outward process. The number specified here gives the maximum number of retries to be made to get the input or put the output from/to the external source.

Default value is 1

Back

Wait Time

The time interval after which retry should be made is given here. This field is used during both Inward & outward process

Default value is 1

Back

Out File Name

The default Outfield name can be overridden using this field

The following are the options available:

- **DATE – Today's date**
- **TIME - Current time**
- **UNIQUE.NO – Unique sequence number.**
- **SPACE – The number of spaces to be added**
  Format: SPACE"("<sequence number>")"
  E.g.: SPACE(14)
  - **GIT.ID – The interface id can be got.**
  - **ID.COMPANY–Common variable ID.COMPANY**

- **If any static value is required then it can also be given.**
  Format: To get the combination of more than one options specified above follow the following format
  DATE *UNIQUE.NO* ID.COMPANY

Back