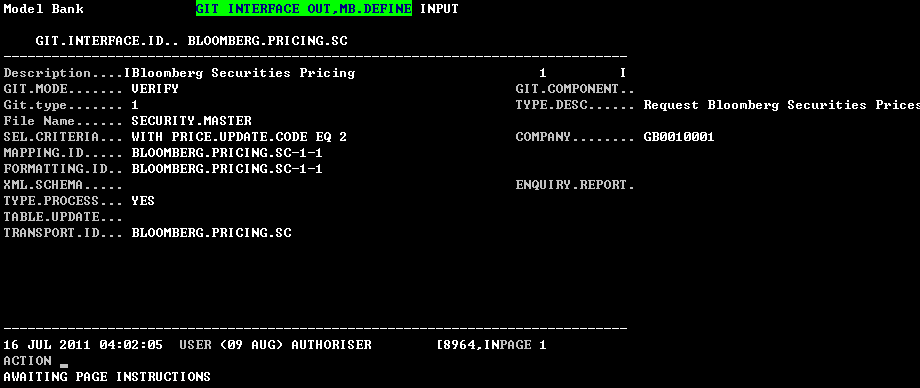
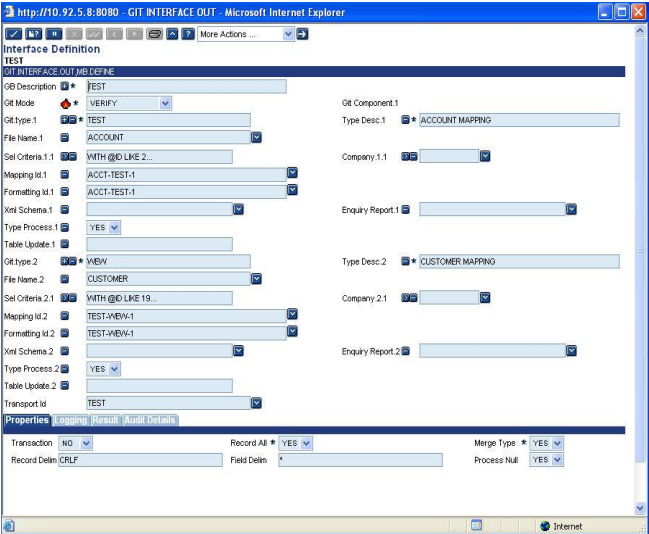
## GIT INTERFACE OUT,MB.DEFINE

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



Screen shot 1



Screen shot 2

### Fields

#### GB Description

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

#### 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.

#### GIT Component

**MAPPING**/**FORMATTING**/ **TRANSPORT/ALL**

This field is no input field if MODE is Verify.

#### GIT Type

Interface Type.1-40 Alpha numeric characters.

#### Type Desc

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

#### File Name

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

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

*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*

#### 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.

#### 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

#### 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.

#### 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.

#### XML Schema

Used to send and receive XML messages

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

#### Enquiry Report

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

#### 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.

#### Table Update

Used to create a T24 file.

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

#### Transport Id

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

Valid id in GIT.TRANSPORT.

#### Transaction

JOURNAL.UPDATE/F.WRITE/WRITE indicator

#### 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.

#### 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**.

#### 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

#### PROCESS.NULL

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

Holds value **YES** or **NO**.

#### Pre Routine

Routine to be executed before GIT processing

Valid Routine name

#### Post Routine

Routine to be executed after GIT processing

Valid Routine name

#### 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.

#### 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.**

#### 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**

#### Clear Files

Clear files before processing.

#### 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.

#### 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.

#### Cache Off

To turn off cache

#### Process Info

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

Updated by System.

## GIT MAPPING OUT RECORD,MB.DEFINE

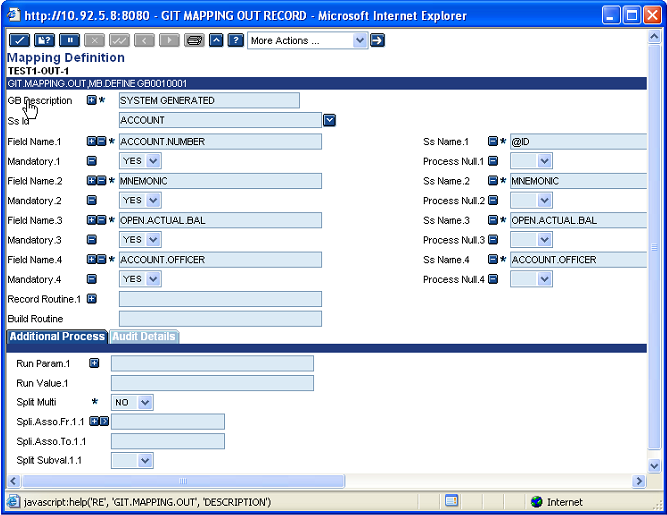
The GIT.FORMATTING.OUT table contains the formatting rules to apply to each mapped

record. The results of the formatting will be saved in the GIT.FORMATTED.OUT table, record

by record.



Screen shot 3



Screen shot 4

### Fields

#### GB Description

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

#### 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

#### Field Name

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

#### 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.

#### 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 nth

position of VM and the mth

SM

position

Format: FIELD VM (or @VM)“,”<nth VM position>

E.g.: FIELD VM, 3 – will extract the field name‟s value‟s 3rd VM value.

Format 2: FIELD SM (or @SM) “,”<nth VM position>”,”<mth 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. 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‟

#### 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.

#### 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.

#### 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

#### 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

#### 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.

#### 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.

#### 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.

#### 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‟.

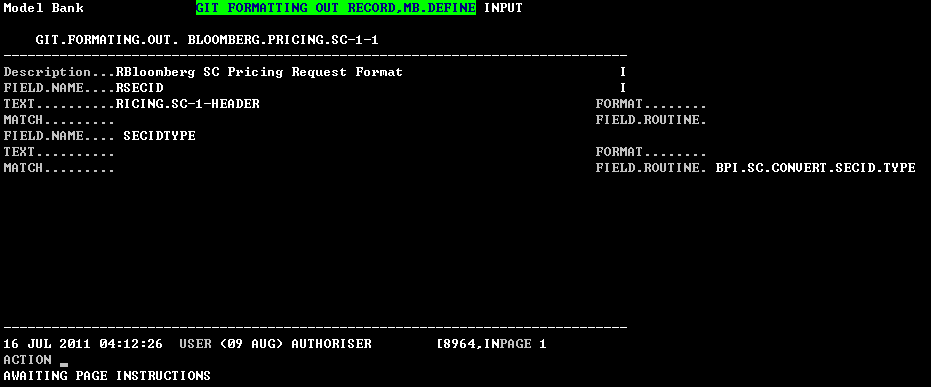
If SPLI.ASSO.FR, SPLI.ASSO.TO are sub value fields and if they have to be split then set this field to „YES‟If SPLI.ASSO.FR, SPLI.ASSO.TO are multi value fields and if they have to be split then set this field to „NO‟

## GIT FORMATTING OUT RECORD, MB.DEFINE

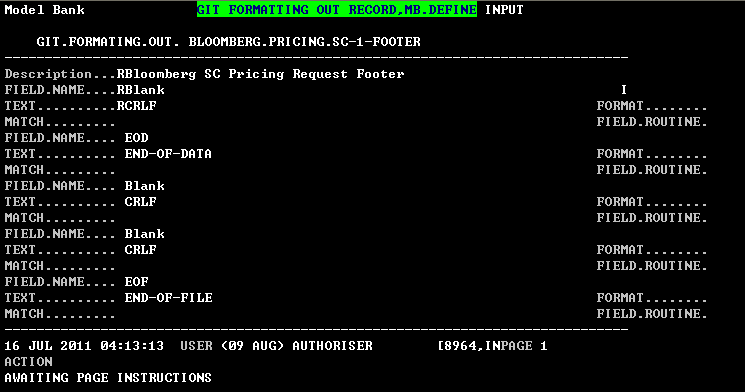
The GIT.FORMATTING.OUT table contains the formatting rules to apply to each mapped

record. The results of the formatting will be saved in the GIT.FORMATTED.OUT table, record

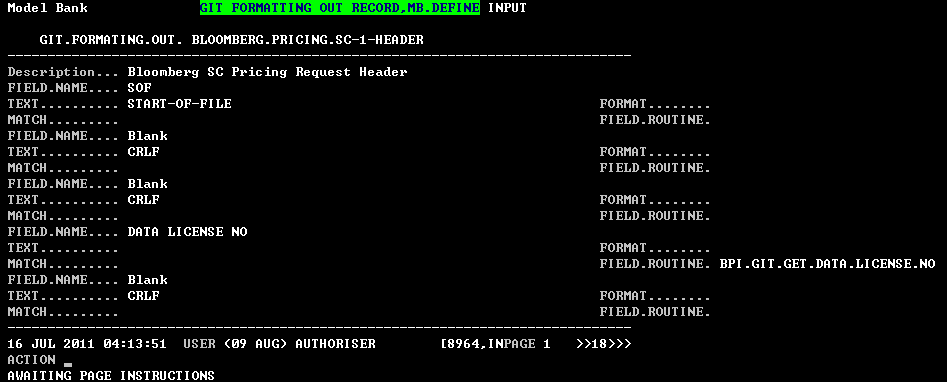
by record.



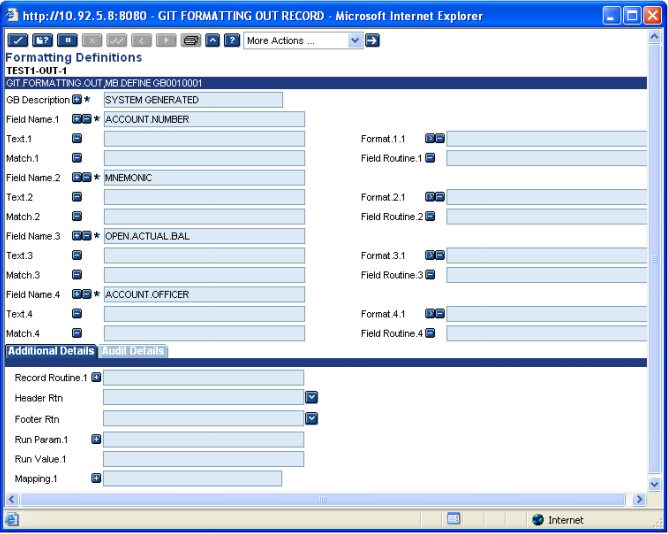
Screen shot 5



Screen shot 6



Screen shot 7



Screen shot 8

### Fields

#### GB Description

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

#### 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.

#### 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.

#### 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

#### 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.

#### Field Routine

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

on the field.

User guide pdf is wrong. Field Routine supports only **one** parameter

E.g. @FLD.RTN (GIT.FORMAT.VAL)

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

#### 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.

#### 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.

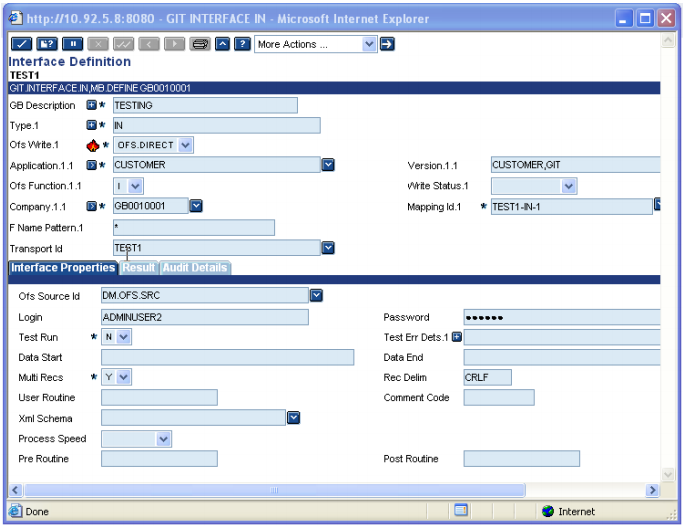
This routine should be given only in the Header record (i.e. when the third part of the ID being HEADER).

#### 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).

## GIT.INTERFACE.IN,MB.DEFINE



### Fields

#### GB Description

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

#### Type

Interface Type.15-40 Alpha numeric characters.

#### OFS Write

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.

#### Application

Valid T24 application.

E.g. : CUSTOMER

#### Version

VERSION name when OFS is used.

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

#### 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

#### 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 update as a LIVE record.

#### Company

Valid COMPANY id.

E.g.: US0010001, DE0010001 etc

#### 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.

#### 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.

#### PROCESS.INFO

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

#### Transport Id

The id using which the processed details are transported.

Valid id in GIT.TRANSPORT file.

#### 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.

#### 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.

#### Password

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

#### 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.

#### 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.

#### Multi Recs

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

Default Value is „Yes‟

#### 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

#### 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.

#### 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

#### Xml Schema

Used to send and receive XML messages

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

#### Process Speed

Used for speedy generation of output.

FUTURE USE

Accepts values “THROUGH” or “QUICK”.

#### Pre Routine

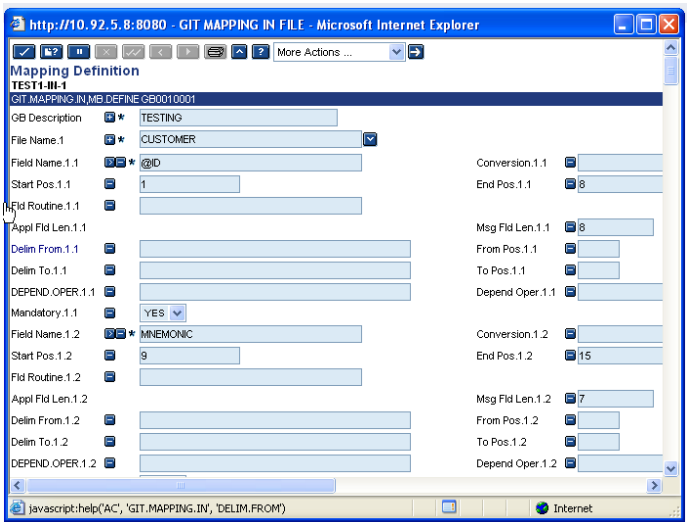
Routines can be attached before the records are processed.

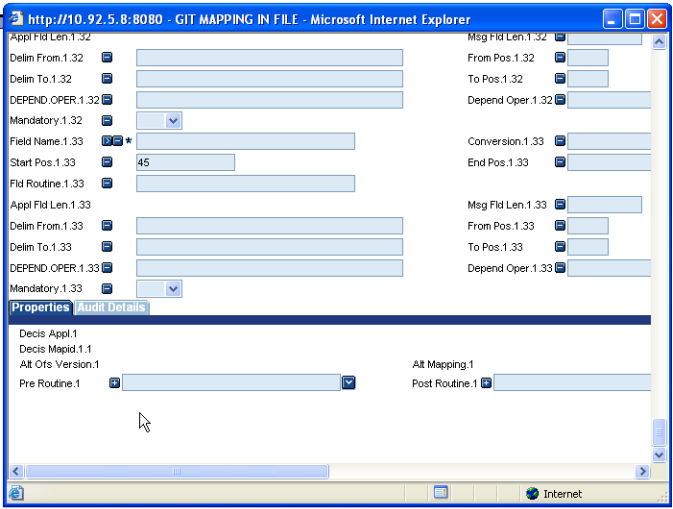
#### Post Routine

Routines can be attached after the records are processed.

## GIT.MAPPING.IN,MB.DEFINE

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.





### 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.

#### GB Description

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

#### File Name

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

#### 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

#### 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, substracted, 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.

#### 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.

#### 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.

#### 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

#### Appl Fld Len

Populated from the Standard selection record. Updated by System

No Input field

#### Msg Fld Len

Actual length of the field in the message.

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

#### Multi Single

Populated from the Standard selection record

No Input field

#### 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.

#### 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

#### 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.

#### 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

#### 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.

#### 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.

#### 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.

#### 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.

#### Decis Mapid

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

#### 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”.

#### 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).

#### Decis Oper

No Input field

#### 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”.

#### 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

#### 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