Part II
Components

Chapter 1
Introduction

This part presents the syntax used for expressing in an unambiguous way the components of the messages, and the special functions used for achieving the validation of an individual component. This checking is independent of the sequence of components.

For this purpose, these chapters have been identified:

- Chapter 2 gives the precise definition of a component inside a S.W.I.F.T. message.
- Chapter 3 describes the conventions used for the notation of the components.
- Chapters 4 and 5 list the syntax and the purpose of the special functions used for the validation of the components.
- Chapter 6 documents the Message User Group Text Validation Rules.

Chapter 2
Definition

A component is a character or string of characters which is defined as a logical subdivision of a text field. A text field may consist of single or multiple components as specified here. Multiple components may be semantically gathered. In case such a group of components does not define precisely the entire field, it will be called 'subfield'.

There are three general categories of components:

- a component of variable length which consists of all numerics, all alphabetics, all alphanumerics, or any characters of the permissible S.W.I.F.T. character sets except 'CRLF', '/', '//' characters which are primarily used as component separators

  Each particular occurrence of 'CRLF', '/', '//' will be signaled.

- a component of fixed length where the order of each character is predefined or belongs to a given list

- a component of variable length which is defined by a specific validation rule mixing numeric and special characters, e.g., the amount component which consists of numerics and a unique separator.

Exceptions:

- meaningless components are not allowed: components containing only blank(s) or only 'CrLf' are not allowed, with the exception of some special fields. (refer to Part II Chapter 5 for field 77E, ~~77S,~~ 77T exceptions and examples)

  Examples of meaningless (invalid) components in field 72 of an MT200:

:72:'CrLf'	1st subfield is empty
//ABCDXYZ'CrLf'	
:72:/ACC/ee'CrLf'	2nd component contains only blanks
//ABCDXYZ'CrLf'	
:72:/ACC/'CrLf'	2nd subfield, component contains only blanks
//ee'CrLf'	
:72:/ACC/ABCDEFG'CrLf'	2nd subfield, component is empty
//'CrLf'	

A component will be validated using one of the three major functions as follows:

- The component form must be one of the elements of a predefined set (code words). e.g., the component is a 'D' or 'C' denoting a debit or credit transaction. All code words must be in upper case.

- The component form is constructed from a composition of elements from predefined set(s).

- The component form is constructed from a composition of elements from predefined set(s) and the meaning of this component corresponds to some semantic checking requirements.

The component validation requirements will be specified on a functional basis, i.e., all of the functions needed to validate all currently existing components will be defined. In order to precisely specify the validation functions, the method used to present the functions will incorporate both a narrative description of each validation function and a representation faithful to the formatting rules defined in *General Information - Standards*.

The component validation functions specified herein will subsequently be used to define the field validation requirements based on component composition.

Chapter 3
Conventions

As specified in Chapter 2, the components validation requirements will be defined on a functional basis, using standard representation conventions given in the *S.W.I.F.T. User Handbook*.

The main notation conventions used in this document are:

- Variable components values are printed between angle brackets (< >).
  For example: <AMOUNT>, <NUMBER>, <DATE1>, etc.
- Where a component or a subfield is optional, it is printed between square brackets ([ ]).
  For example: [<DATE1>], ['/' <DC>], etc.
- If a component, a group of components, or a subfield, may be repeated several times, the minimum and maximum number of repetitions is specified immediately after the definition of that entity.
  For example:
  ['CRLF'35x]0-3
  the group <carriage return _ line feed _ up-to-35-characters> is optional, and may be repeated maximum 3 times.
  Note: in this case we could have used parentheses instead of square brackets, the result would be identical.
  <DATE2>['/'<DATE1>]0-11
  the component <DATE2> is mandatory, the subfield ['/'<DATE1>] is optional, but it may be present up to 11 times.
- A mandatory choice of one component or subfield from several possibilities is indicated by a vertical bar (|, the vertical bar represents the exclusive OR relationship).
  For example:
  <HHMM> | <TIME2>
  either <HHMM> or <TIME2> must be present, not both.
  7!a<DATE2> | <HHMM>
  this subfield must be composed of a 7 alphabetic character component (fixed length), followed by either <DATE2> or <HHMM> (one and only one of the later 2 components must be present).
- Parentheses may be used to identify groups of components that belong together, they may also be required to specify an OR (|) relationship between groups of components.
  For example: ref. <FLD72> in Part II, section 4.10. ['CRLF'(<INSTR>|'//'33x)]0-5
  Where each of up to 5 optional lines is/are validated by the function <INSTR> (ref. Part II, 4.9), or the line must begin with '//' (i.e. the first 2 characters are a double slash) and must contain minimum 1 character, maximum 33 characters from the <x> character set.

The full rules for the notation of components inside messages and fields can be found in the *S.W.I.F.T. User Handbook.*

These rules can be summarized as follows:

Field Length		Field Type	
nn	: maximum length (minimum is 1)	n	: numeric digits (0 through 9) only
nn-nn	: minimum and maximum length	a	: alphabetic capital letters (A through Z), upper case only
nn!	: fixed length	x	: SWIFT X set any character of the X permitted set (General FIN application set) upper case and lower case allowed
nn*nn	: maximum number of lines times maximum line length	y	: SWIFT Y set any character of the Y permitted set (EDI service specific set), upper case only
		z	: SWIFT Z set all characters included in the X and Y sets, plus a couple of special characters
		c	: alpha-numeric capital letters (upper case), and digits only
		h	: hexadecimal letters A through F (upper case), and digits only
		s	: sign (+ or -)
		e	: blank or space
		A	: alphabetic, upper case or lower case A through Z, a through z
		B	: alphanumeric upper case or lower case A through Z, a through z and 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Based on these rules, the following sets will be used for the presentation of the validation functions:

<DC>	'D' | 'C'	the debit/credit code (Error Code T51)
<DM>	'D' | 'M'	the days/months code (Error Code T61)

The value represented by the following set is maintained in a subtable in the currency code table (CURSEP). The maximum number of decimal digits permitted for each currency code is also included in this table.

<CUR> 3!a = currency code, e.g., USD for the United States dollar. (Error Code T52)

The currency code table can be found in the *BIC Directory*. This table lists the currency codes that are used to validate field components declared as currencies (their format <CUR> is to be validated against the codes included in this table).

All code words (e.g., 'D', 'C', 'PCT', 'OUR', 'BEN', etc) and currency codes ('USD', 'BEF', etc.) must be in upper case format.

The term “BIC” is used to refer to the ISO Bank Identifier Code; depending on where it is used in a FIN message, a BIC must be either connected (referred to as a SWIFT BIC) or not connected (referred to as a non-SWIFT BIC).

FIN user-to-user messages may only be sent from and received by SWIFT connected users, therefore, only SWIFT BICs can be used in the header of a SWIFT FIN message, while SWIFT BICs and non-SWIFT BICs may be referenced in the text of the FIN messages (refer to Part III, Chapter 3, fields formats).

Additional content checks are performed. Refer to the special field exceptions table for a listing of values used for code word checking (text validation).

Chapter 4
Special functions

4.1 Date

The **Date** function checks the date component, the function is represented by:

<DATE1>, or <DATE2>, or <DATE3>, or <DATE4>

The syntax for each function is as follows:

<DATE1> MMDD

<DATE2> YYMMDD

<DATE3> YYMM

<DATE4> YYYYMMDD

Where all 'Y', 'M', and 'D' characters must be numeric characters :

i.e. figures 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

'YYYY' must not consist entirely of zeroes, e.g. '0000' is rejected "T50".

'MM' check: eg., 00<MM<13 where 'MM' cannot exceed 12 or be less than 01.

'DD' check: this check depends upon the 'YY' or 'YYYY', and 'MM' values.

eg., 00<DD<32: check for a 31 day month.

eg., 00<DD<31: check for a 30 day month

Leap years are accommodated in the validation check:

eg., 00<DD<30: for a leap year 29-day month

eg., 00<DD<29: for a year having a 28-day month.

Standard presentation:
<DATE1> 4!n
<DATE2> 6!n
<DATE3> 4!n
<DATE4> 8!n

<DATE1> validation specs.

Depending on the field where the <DATE1> is recorded, the value of the "year" is selected as follows:

- in field 61: the year value is selected from the system date at the time the validation is performed.

The code error "T50" is returned after a format error in the (date) components <DATE1>, or <DATE2>, or <DATE3>, or <DATE4>.

4.1.1 Value Date

The Value Date function allows the receiver to request some FIN messages in Value Date order. At present, only the following message types are considered for Value Date ordering:

MT910 (field 32A),

all category 1 and category 2 messages containing fields 30 or 32A, or both 30 and 32A, except the common group messages 192, 292, 195, 295, 196 and 296.

List of Message types candidates for Value Date Ordering:

Cat 1	Cat 2	Cat 9
	200	910
101	201	
102_not_STP	202	
102_STP	203	
103_not_STP	204	
103_STP	205	
104	206	
107	207	
110	210	
111	256	
112		

The valid range of value dates implemented on the SWII system is from 1980 to 2060 (limits included), therefore the <year> (”YY” in <DATE2> format) of a value date component must be included in the allowed value date range.

The ”YY” component of a Value Date <DATE2> component is validated as follows:

```
IF YY >79
THEN year = ″19″ || ″YY″  (the sign ″||″ means ″concatenate″)
ELSE year = ″20″ || ″YY″
ENDIF
IF year > 2060
THEN <error:’T50’>
ENDIF
```

Note: *This validation is in addition to the <DATE2> format validation.*

If there is more than one Value Date Field in a message, for Value Date ordering purpose, the lesser date will be selected. (ref. example below)

EX: MTxxx

...

:30:951218 Value Date = 18 December 1995

...

:32A:020103USD123000,00 Value Date = 3 January 2002

...

In this example the Field 30 Value Date (18 December 1995) is selected for Value Date ordering of the message.

The code error ”T50” is returned after an invalid value date.

4.1.2 YEAR

The function **YEAR** is represented in the fields definitions by : **<YEAR>**.

The syntax is : 4!n

<4!n> must not consist entirely of zeroes, e.g. '0000' is rejected "T50".

4.2 Time

The **Time** function checks the time component, the function is represented by:

<HHMM>, or <TIME2>.

The syntax for the <TIME2> function is: HHMMSS

The time sub-components are validated as follows:

00<=HH<=23

00<=MM<=59

00<=SS<=59

Standard presentation:	<HHMM> 4!n <TIME2> 6!n

The code error ”T38” is returned after a format error in the (time) components <HHMM>, or <TIME2>.

4.3 <SWIFTBIC> and <NON-SWIFTBIC> used in copy field(s) of MTs n92, n95, n96

In the copy fields of the Message Types n92, n95 and n96 the functions <SWIFTBIC> and <NON-SWIFTBIC> perform the following syntax checks:

Bank code : 4!a	four alphabetic characters (fixed length).
Country code : 2!a	two alphabetic characters (fixed length).
Location code 1 : 1!c	one alphanumeric character, digits ‘0’ and ‘1’ not allowed.
Location code 2 : 1!c	one alphanumeric character, letter ‘O’ not allowed.
Branch code : [3!c]	optional, three alphanumeric characters.

The code ‘BIC’ is not allowed.

The letter ‘X’ is not allowed as the first letter, unless it is used in ‘XXX’.

A testandtraining destination is not allowed if the emitter of the msg is a LIVE user.

If an error is detected, the system returns the error code T27.

T27 - BIC incorrectly formatted or invalid.

4.4 <SWIFTBIC> and <NON-SWIFTBIC> used in other than: copy field(s) of MTs n92, n95, n96

4.4.1 S.W.I.F.T. Address: <SWIFTBIC>

This function is performed because the 8th position of the BIC address does NOT contain the numeric character ”1” value.

The function <SWIFTBIC> checks if the component consists of 8 alphanumeric characters, or 8 alphanumeric characters followed by 3 optional alphanumeric characters. The first eight characters representing a SWIFT DESTINATION are in the format 6!a<LC> (ref. Standards General Information, section 3.4.2 : <LC> = Location Code = 2!c). Therefore, the component is variable-length and may consist of eight or eleven characters. The first eight characters must represent a S.W.I.F.T. FIN DESTINATION, i.e. a destination pre-defined in the S.W.I.F.T. Database, and enabled for the FIN application, and with the status ”TRUE-CUTOVER” (unpublished SWIFTBICs are not allowed in this text part of the FIN messages). The last three optional characters must be one of the BRANCH CODES defined in the db for that destination.

Note: *Since all SWIFTBICs are published in the BIC directory with an 8 character address, the branch code ”XXX” is assigned to all SWIFTBICs, however the branch code ”XXX” is not printed in the BIC directory.*

Note: *A published 8 character SWIFT BIC with an unpublished branch code is ACK, the system does not have a flag to recognize the unpublished branch codes; therefore, only the unpublished destinations (8 char SWIFTBIC) are rejected.*

Standard presentation: <SWIFTBIC>::= <SWIFTDESTINATION>[3!c]

<SWIFTDESTINATION>::= 8 char. dest. recorded in the SWII db. as either:

1. a Master destination with the statuses ”enabled for FIN” and ”officially cutover”, or
2. a Synonym destination with the status ”enabled for FIN”, and its Master with the status ”officially cutover”, or
3. a Test and Training destination with the status ”enabled for FIN”.

A testandtraining destination is not allowed if the emitter of the msg is a LIVE user.

4.4.2 NON-S.W.I.F.T. Address: <NON-SWIFTBIC>

This function is performed because the 8th position of the BIC address contains the numeric character ”1” value.

The function <NON-SWIFTBIC> checks if the component consists of 8 alphanumeric characters, or 8 alphanumeric characters followed by 3 alphanumeric characters. The first 8 characters must represent a BIC address pre-defined in the S.W.I.F.T. database as a non-connected SWIFT institution.

If the BIC is 11 char long, the last three characters must be one of the BRANCH CODES defined in the db for that 8 char. destination.

Note: *Similar to the SWIFTBICs, the branch code "XXX" is assigned (by default) to the NON-SWIFTBIC that is published as an 8 character address in the BIC directory. However all NON-SWIFTBICs are not necessarily published as an 8 char address, hence the branch code "XXX" is not necessarily valid for some NON-SWIFTBICs.*

Standard presentation:

<NON-SWIFTBIC>::= <7!c> "1" [<3!c>]

<NON-SWIFTBIC>::= 8 or 11 char. dest. recorded in the SWII db. as a not-connected (for FIN) financial institution.

Note: about NON-SWIFTBIC's branch codes

The branch code for NON-SWIFTBIC's is not always optional. A NON-SWIFTBIC is published as either: (1) an 8 char. address, or (2) one or more 11 char. addresses, or (3) an 8 char. address and one or more 11 char. addresses. It is only when the NON–SWIFTBIC is published as an 8 char. address, with or without 11 char. address(es), that the branch code "XXX" may be used, i.e. is optional (ref. 1 or 3), the published 8 char. address alone may be used (ref. 1 or 3), or any of the published 11 char. addresses may be used (ref. 3).
By the same token, if a NON-SWIFTBIC is not published as an 8 char. address, i.e. is published as one or more 11 char. addresses, then the user is not allowed to refer to the 8 char. address alone, nor to use the branch code "XXX"; but the user must necessarily refer to one of the published 11 char. addresses (ref. 2).

4.5 Message Type

The function Message Type (or <MT>) will validate a component containing a message type according to the following rule: the component must be composed of 3 numeric characters, and its value must be greater than or equal to 100, and less than or equal to 999.

The error code generated by <MT> will be T18.

4.6 <Number> <Amount>

These 2 functions check a number component which is of variable length.

An invalid <Number> or <Amount> component will be rejected with the error code T40 or T43.

The <Number> and <Amount> components consists of multiple parts defined as follows:

- An integer part which consists of one to n numerics, where n is dependent on the maximum length specified for the component and dependent on the number of characters in the decimal part (if present). The integer part is mandatory in the number component and at least one character must appear, leading zeros are allowed.
- A decimal separator part consisting of one special character which must be a decimal comma (,). The decimal separator comma is mandatory in the number component.
- A decimal part which consists of zero to n numerics, where n is dependent on the maximum length specified for the component and dependent on the number of characters in the integer part.

To determine the maximum component length, the number of numerics in the integer part, the decimal separator part, and the number of numerics in the decimal part (if present) must be totaled. The character count must be compared against the maximum length parameter which is variable, to ensure component length is not exceeded.

Spurious characters, e.g., punctuation marks or blanks, are not permitted within the integer part or decimal part (if present).

Standard presentation:

If the integer part of the number component <NIP> is defined as Nn, the decimal separator part of the number component is defined as ',' and the decimal part of the number component <NDP> is defined as [Nn], then the number component <NBR> can be defined as <NIP> ',' <NDP>.

Examples:	123,	000100,00
	456,0789	00,0
	3,	000000000000,00
	1,234567	

Maximum decimal digits check:

In some message formats, the number of numerals in the decimal part must be checked against the maximum number allowed for the corresponding currency code or related code word: for the message types where this check is required, the semantic rule 3 or 89 is specified within the message format definition. (see Part V - Table of Messages).

4.7 SUB-6

This function checks if the component in subfield 6 in field 61 is four characters in length such that:

- The first character of the component must be either 'S', 'N' or 'F'.
- If the first character is 'S', the last three characters must be validated by the <MT> function.
- If the first character is 'N' or 'F', the last three characters must consist of alphanumerics.

4.8 SB - LC

The SB-LC function checks the validity of field 22, subfield 2, and field 22C, this field or subfield contains three components, which are defined as follow:

1st component : <SB1><LC1>

2nd component : 4!n

3rd component : <SB2><LC2>

Irrespective of the Message Type being validated, the following checks are performed:

The SB-LC function will check that <SB1><LC1> and <SB2><LC2> appear in fields 22 and 22C in alphabetical order, if this control is negative, the function will issue a T96 error code (see Part IV, Chapter 3).

Depending on the Message Type where fields 22 and 22C is recorded specific checks are performed:

4.8.1 Non-common group messages types (not n92, n95, n96).

1st component : <SB1><LC1>

with	<SB1> = 4!a	representing a bank code (1st four characters of the message originator or receiver, e.g., CHAS in 'CHASUS33')
	<LC1>= 2!c	representing the location code of the message originator or receiver represented in <SB1> (7th and 8th character of the message originator or receiver, e.g., 33 in 'CHASUS33').

2nd component : 4!n

3rd component : <SB2><LC2>

with	<SB2> = 4!a	representing the bank code of the other destination involved in the message exchange
	<LC2>= 2!c	representing the location code of the other destination involved in the message exchange

The components 1 and 3 must contain the respective bank codes and location codes extracted from the originator and receiver mnemonics recorded in the message headers, otherwise the function will issue a T95 error code.

Moreover, in all MTs where the fields 22 and 22C is present, the SB-LC function will check if a common reference mismatch exists between the 2nd component and other fields or subfields . Refer to T22 error code (see Part IV, Chapter 3) for the corresponding MTs and related field where this check is applied. (ex: MT 601 field 32B, subfield 2)

In MTs 360, 361, 362, 364, 365 :

> the 2nd component must be composed of '<u>YYMM</u>' from 'YY<u>YYMM</u>DD' of field 30P.

In other MTs:

> the 4th digit of the 2nd component must be the rightmost non-zero digit of the related field; and the digits 1 through 3 must be the 3 digits to the left of it, 0-filled on the left as needed. The 4 digits must be '0000' if the related field has a value of '0'.

4.8.2 Common group message types n92, n95, n96.

1st component : 4!a 2!c

2nd component : 4!n

3rd component : 4!a 2!c

When the last character of the component 2 consists of a ‘0’, and its preceding character does not consist of a ‘1’, then the entire component 2 must consist of ‘0000’, if this control is negative, the function will issue a T63 error code (see Part IV, Chapter 3).

```
IF 2nd_component = <nnn>"0"
     THEN IF 2nd_component NOT= ("0000" | <nn>"10")
          THEN <errror_T63>
```

The values ’0’ and ’1’ are not permitted in <LC1> and <LC2>, however the value ’0’ is allowed in the second (rightmost) position if the emitter is a test and training user. The function will issue a T94 error code.

4.9 Instruction

The function Instruction (or <INSTR>) will validate the content of an instruction and additional information usually found in the field 72, according to the following syntax:

<INSTR>: (’/’1!c’/’[32x])|(’/’2!c’/’[31x])|(’/’3!c’/’[30x])|(’/’4!c’/’[29x])|
(’/’5!c’/’[28x])|(’/’6!c’/’[27x])|(’/’7!c’/’[26x])|(’/’8!c’/’[25x])

Errors found by this validation function will be reported with the usual codes T12, T17, T30, T31, T32, T33, and T34.

4.10 <FLD72> Sender to Receiver (field 72)

The function Sender to Receiver (or <FLD72>) validates the content of the field 72, according to the following syntax:

<FLD72>: <INSTR>
[’CRLF’(<INSTR>|’//’33x)]0-5

Errors found by this validation function will be reported with the usual codes T12, T17, T30, T31, T32, T33, and T34.

In addition to the field 72 structured format validation (i.e. <FLD72>), the ERI (Euro Related Info) validation must be applied, refer to section 4.18 <ERI-FLD72structured> for a complete description of this additional validation.

4.11 VAR-SEQU-1

This function checks the syntax of a sequence of 1 mandatory variable-length subfield plus 1 optional variable-length subfield, separated by two optional slashes (’//’).

This sequence can be found in field 61 (subfields 7 and 8) and field 66A (subfields 4 and 5).

The syntax is expressed as follows:

```
<VAR-SEQU-1> : 16x['//'16x]
```

However, due to the fact that slashes are not considered as component (or subfield) delimiters, since they can be part of the content of these components (or subfields), a specific validation process must be implemented, which is able to handle, with a number of assumptions, the use of these slashes.

This validation process is detailed below.

Let us define the sequence validated by <VAR-SEQU-1> as the set of characters starting from the first character encountered up to the first 'CRLF' encountered.

```
IF length of the sequence before 'CR' > 34 char
     THEN <error : 'T33'>
ENDIF
IF length of the sequence before 'CR' = 0 char
     THEN <error : 'T17'>
ENDIF

IF the sequence contains one '//'
     THEN
          BEGIN_BLOCK   (It is assumed the 1st man & 2nd opt subfields
                        are both present)
          IF     length of the sequence before '//' > 16 char OR length
                 of the sequence before '//' = 0 char
                 THEN <error : 'T24'>
          ENDIF
          IF the sequence before '//' contains all blanks
               THEN <error : 'T25'>
          ENDIF
          IF the sequence before '//' contains a spurious 'LF'
               THEN <error : 'T31'>
          ENDIF
          IF length of the sequence after '//' > 16 char OR length of
          the sequence after '//' = 0 char
               THEN <error : 'T23'>
          ENDIF
          IF the sequence after '//' contains all blanks
               THEN <error : 'T25'>
          ENDIF
          IF the sequence after '//' contains a spurious 'LF'
               THEN <error : 'T31'>
          ENDIF
          END_BLOCK
     ELSE
          BEGIN_BLOCK (Its assumed only the man subfield
          is present)
          IF length of the sequence > 16 char
               THEN <error : 'T24'>
          ENDIF
          IF the sequence contains all blanks
               THEN <error : 'T25'>
          ENDIF
          IF the sequence contains a spurious 'LF'
               THEN <error : 'T31'>
          ENDIF
          END_BLOCK
ENDIF
```

It is implied in this validation sequence that the text pointer will be adjusted if the '//' sequence is encountered during the validation process.

4.12 VAR-SEQU-2

This function checks the syntax of a sequence of 1 mandatory variable-length subfield plus 1 optional subfield, separated by one optional slash ('/').

This sequence can be found in field 26A (subfields 1 and 2).

The syntax is expressed as follows:

```
<VAR-SEQU-2> : 16x['/'4!x]
```

However, due to the fact that slashes are not considered as component (or subfield) delimiters, since they can be part of the content of these components (or subfields), a specific validation process must be implemented, which is able to handle, with a number of assumptions, the use of these slashes.

This validation process is detailed below.

Let us define the sequence validated by <VAR-SEQU-2> as the set of characters starting from the first character encountered up to the first 'CRLF' encountered.

```
IF length of the sequence > 21 char
     THEN <error : 'T33'>
ENDIF
IF length of the sequence before 'CR' = 0 char
     THEN <error : 'T17'>
ENDIF
IF the sequence contains one '/'
```

```
        THEN
              BEGIN_BLOCK (Its assumed the 1st man & 2nd opt subfields are
              both present)
              IF length of the sequence before '/' > 16 char OR length of
              the sequence before '/' = 0 char
                   THEN <error : 'T24'>
              ENDIF
              IF the sequence before '/' contains all blanks
                   THEN <error : 'T25'>
              ENDIF
              IF the sequence before '/' contains a spurious 'LF'
                   THEN <error : 'T31'>
              ENDIF
              IF length of the sequence after '/' NOT= 4 char
                   THEN <error : 'T23'>
              ENDIF
              IF the sequence after '/' contains all blanks
                   THEN <error : 'T25'>
              ENDIF
              IF the sequence after '/' contains a spurious 'LF'
                   THEN <error : 'T31'>
              ENDIF
              END_BLOCK
        ELSE
              BEGIN_BLOCK (Its assumed only the man subfield is present)
              IF length of the sequence > 16 char
                   THEN <error : 'T24'>
              ENDIF
              IF the sequence contains all blanks
                   THEN <error : 'T25'>
              ENDIF
              IF the sequence contains a spurious 'LF'
                   THEN <error : 'T31'>
              ENDIF
              END_BLOCK
ENDIF
```

It is implied in this validation sequence that the text pointer will be adjusted if the '/' char is encountered during the validation process.

4.13 VAR-SEQU-3

This function checks the syntax of a sequence of 1 mandatory variable-length subfield plus 1 optional variable-length subfield, separated by one optional slash ('/').

This sequence can be found in field 26B (subfields 1 and 2).

The syntax is expressed as follows:

```
<VAR-SEQU-3> : 16x['/'16x]
```

However, due to the fact that slashes are not considered as component (or subfield) delimiters, since they can be part of the content of these components (or subfields), a specific validation process must be implemented, which is able to handle, with a number of assumptions, the use of these slashes.

This validation process is detailed below. Let us define the sequence validated by <VAR-SEQU-3> as the set of characters starting from the first character encountered up to the first 'CRLF' encountered.

```
IF length of the sequence before 'CR' > 33 char
     THEN <error : 'T33'>
ENDIF
IF length of the sequence before 'CR' = 0 char
     THEN <error : 'T17'>
ENDIF
IF the sequence contains one '/'
     THEN
          BEGIN_BLOCK   It is assumed the 1st man & 2nd opt subfields
                        are both present)
          IF     length of the sequence before '/' > 16 char OR
                 length of the sequence before '/' = 0 char
               THEN <error : 'T24'>
          ENDIF
          IF the sequence before '/' contains all blanks
               THEN <error : 'T25'>
          ENDIF
          IF the sequence before '/' contains a spurious 'LF'
                THEN <error : 'T31'>
          ENDIF
          IF     length of the sequence after '/' > 16 char OR
                 length of the sequence after '/' = 0 char
                THEN <error : 'T23'>
          ENDIF
          IF the sequence after '/' contains all blanks
                THEN <error : 'T25'>
          ENDIF
          IF the sequence after '/' contains a spurious 'LF'
               THEN <error : 'T31'>
          ENDIF
          END_BLOCK
     ELSE

          BEGIN_BLOCK(Its assumed only the man subfield is present)
          IF length of the sequence > 16 char
               THEN <error : 'T24'>
          ENDIF
          IF the sequence contains all blanks
               THEN <error : 'T25'>
          ENDIF
          IF the sequence contains a spurious 'LF'
               THEN <error : 'T31'>
          ENDIF
          END_BLOCK
ENDIF
```

It is implied in this validation sequence that the text pointer will be adjusted if the '/' char is encountered during the validation process.

4.14 VAR-SEQU-4

This function checks the syntax of a sequence of 1 mandatory variable-length subfields and 1 optional variable-length subfield, separated by two optional slashes ('//').

This sequence can be found in field 26C (subfields 5 and 6).

The syntax is expressed as follows:

```
<VAR-SEQU-4> : [4x]['//'8x]
```

However, due to the fact that slashes are not considered as component (or subfield) delimiters, since they can be part of the content of these components (or subfields), a specific validation process must be implemented, which is able to handle, with a number of assumptions, the use of these slashes.

This validation process is detailed below.

Let us define the sequence validated by <VAR-SEQU-4> as the set of characters starting from the first character encountered up to the first 'CRLF' encountered.

(It is assumed that either the 1st, or the 2nd, or both optional subfields are present.)

```
If length of the sequence before 'CR' > 14 chars
     THEN <error : 'T33'>
ENDIF

If the 1st 2 chars in the sequence = '//'
      THEN
           BEGIN_BLOCK_1 (Its assumed that only the 2nd opt subfield
                         is present)
           If     length of the sequence after '//' > 8 chars OR
                  length of the sequence after '//' = 0 char
                THEN <error : 'T23'>
           ENDIF

           If the sequence after '//' contains all blanks
                THEN <error : 'T25'>
           ENDIF
           If the sequence after '//' contains a spurious 'LF'
                THEN <error : 'T31'>
           ENDIF
           END_BLOCK_1
```

```
        ELSE
              BEGIN_BLOCK_2 (Its assumed the 1st and possibly the 2nd
                            opt subfield(s) is/are present)
              If the sequence contains one '//'
                    THEN
                         BEGIN_BLOCK_3 (Its assumed the 1st and 2nd opt
                         subfields are present)
                         If length of the sequence before '//' > 4 chars
                              THEN <error : 'T24'>
                         ENDIF
                         If the sequence before '//' contains all blanks
                              THEN <error : 'T25'>
                         ENDIF
                         If the sequence before '//' contains a spurious 'LF'
                              THEN <error : 'T31'>
                         ENDIF
                         If  length of the sequence after the '//' > 8 chars
                            OR
                            length of the sequence after the '//' = 0 char
                              THEN <error : 'T23'>
                         ENDIF
                         If the sequence after '//' contains all blanks
                              THEN <error : 'T25'>
                         ENDIF
                         If the sequence after '//' contains a spurious 'LF'
                              THEN <error : 'T31'>
                         ENDIF
                         END_BLOCK_3
              ELSE
                         BEGIN_BLOCK_4 (Its assumed that only the 1st opt
                         subfield is present
                         If the sequence before 'CR' > 4 chars
                              THEN <error : 'T24'>
                         ENDIF
                         If the sequence before 'CR' contains all blanks
                               THEN <error : 'T25'>
                         ENDIF
                         If the sequence before 'CR' contains a spurious 'LF'
                              THEN <error : 'T31'>
                         ENDIF
                    END_BLOCK_4
              END_BLOCK_2
        ENDIF
ENDIF
```

It is implied in this validation sequence that the text pointer will be adjusted if the '//' chars are encountered during the validation process.

4.15 S.W.I.F.T. Party Field Option J. <PARTYFLD/J>

The <PARTYFLD/J> function will check the contents of S.W.I.F.T. Party Fields 53J, 56J, 57J, 58J, 82J, 83J, 84J, 85J, 86J, 87J and 88J based on the rules defined below.

<PARTYFLD/J> generic format: 5*40x

The generic rules that must be applied to all <PARTYFLD/J> fields are:

- max 5 lines,
- max of 40 character per line.

<PARTYFLD/J> specific format: <code_word><format>[<code_word> <format>]0-n

In addition to the generic format documented above, the content of the <PARTYFLD/J> fields must be validated as a list of 'pair(s)': <code_word><format>, the valid pairs are specified below and are grouped into lists: Format1, Format2, Format3 and Format4.

The pairs "code word / format" enclosed in [ ] are optional, those not enclosed in [ ] are mandatory within that format-list. (ex: /ABIC/ <...> and /NAME/ <...> are mandatory in Format1, however only /NAME/ is mandatory in Format4)

Per Party Field option J the following formats are specified:

53J	<Format1> | <Format2> | <Format3>
56J	<Format1> | <Format2> | <Format3>
57J	<Format1> | <Format2> | <Format3>
58J	<Format1>
82J	<Format1>
83J	<Format4>
84J	<Format1>
85J	<Format1>
86J	<Format1> | <Format2> | <Format3>
87J	<Format1>
88J	<Format1>

<Format1>

'/ABIC/'	<SWIFTBIC> | <NON-SWIFTBIC> | 'UKWN'
'/NAME/'	<34x>
['/ACCT/'	<34x>]
['/ADD1/'	<35x>]

['/ADD2/' <35x>]

['/CITY/' <35x>]

['/USFW/' <9!n>]

['/USCH/' <6!n>]

['/GBSC/' <6!n>]

['/CLRC/' <35x>]

<Format2>

'/NETS/'

<Format3>

'/SSIS/'

<Format4>

['/ABIC/' <SWIFTBIC> | <NON-SWIFTBIC> | 'UKWN']

'/NAME/' <34x>

['/ACCT/' <34x>]

['/ADD1/' <35x>]

['/ADD2/' <35x>]

['/CITY/' <35x>]

['/USFW/' <9!n>]

['/USCH/' <6!n>]

['/GBSC/' <6!n>]

['/CLRC/' <35x>]

Note: *In all the lists of "Floating Code Words" there is no **CrLf** characters -i.e. Carriage Return Line feed- (or special characters to force the printing device to reposition the printing-head on a new line). Floating code words are not assigned at a specific position within a text line, hence they can not be defined with reference to the physical end of a line, however the generic rules which define the maximum number of lines must be verified. In other words, the generic format of a field with floating code words determines the maximum number of lines in that field.*

*Please note that in some cases (ex: '/ADD1/' <35x>) the number of characters plus the length of the code word (6 char.) will exceed the maximum number of characters allowed in one line (ex here above: 41 versus 40), the text portion must be split on more than one line to accommodate the generic definition. In the following format definitions for floating code words, we will indicate the maximum number of "significant" characters that are allowed in that particular format, and we will rely on the generic format validation to verify the maximum number of lines (i.e. the count of **CrLf** characters) and the maximum number of characters per line (i.e. number of characters between **CrLfs**).*

Floating code words / formats rules.

The following rules apply to the usage of the <code_word><format> pairs:

1. all the pairs present in the text of a message must belong to one and only one format. (ex: in a field 53J, the combination '/NAME/' <34x> '/NETS/' is not allowed, the first code word belongs to Format1 the second belongs to Format2)
   Note: this rule is not applicable to the code words which belong to more than one format list. (ref. Format1 and Format4)
2. each code word and corresponding format must be validated as specified in the lists Format1, Format2, Format3, etc. (ex: in Format1 and Format4, '/USFW/' must be followed by a 9 numeric character fixed length field)
3. more than one code word may be used in a line.
   Note: as long as the generic rules are verified, i.e. the maximum number of characters per line and the maximum number of lines in a field, another code word may follow the previous code word format text field.
   ex:
   /ABIC/UKWN/NAME/CHASE HQ/USCH/123456(**CrLf)**
   /ADD1/7TH AVENUE/ADD2/NEW YORK CITY
4. the information relating to a code word may be split across lines. (ex: after '/NAME/', the <34x> field may contain the end_of_line **CrLf** characters)
   ex (valid code word formatting, on one line, two lines, and three lines):
   /USCH/12356(CrLf)
   /NAME/CHASE(CrLf)
   MANHATTAN, NY/ABIC/UKWN(CrLf)
   /ACCT/000-1234567-99(CrLf)
   FIFTH AVENUE, 155(CrLf)
5. a code word itself must not be split across lines. (ex: '/NA**CrLf**ME/' is invalid)
   From the beginning '/' to the ending '/', all characters must be continuous.
6. the special character '/' is not allowed in any of the 'X', 'Y' or 'Z' character sets that is permitted to follow a code word. This exception is necessary because the character '/' indicates always the beginning of a code word.
7. a code word must not be duplicated within a field.
8. in each list (i.e. Formatn.), the code words / format pairs are listed according to the (user group) recommended sequence, however their usage within a message may be in any order. (ex: in a field 83J -Format4-, the sequence /ACCT/ blablabla /NAME/ blablabla /CLRC/ blablabla is perfectly valid)

Note: *The following error codes are used to validate the content of the 'ABIC' line.* ***(Error Codes T17, T27, T28, T29, T44 or T45)****. The minimum and maximum number of lines allowed depends on the field and code list that is used. When using* ***<Format1>*** *or* ***<Format4>*** *the maximum number of lines allowed is 5, whereas when using* ***<Format2>*** *or* ***<Format3>*** *only one line must be used. A missing mandatory code word, or an invalid code word in any line of the option 'J' party fields will be rejected with the error code* ***"T78****". Any additional line will be rejected with the generic* ***"T30"*** *error code.*

The examples below illustrate some valid, and invalid messages:

Valid messages:

:53J:/ABIC/CHASUS33(CrLf)
/NAME/12345xxxxx12345xxxxx12345+++++1234(CrLf)
/ACCT/12345xxxxx12345xxxxx12345+++++1234(CrLf)
/ADD1/12345xxxxx12345xxxxx12345+++++1234(CrLf)
5/ADD2/12345xxxxx12345xxxxx12345+++++123(CrLf)

:53J:/NAME/12345xxxxx12345xxxxx12345++++(CrLf)
+1234/ABIC/UKWN(CrLf)
/CITY/12345xxxxx12345xxxxx12345+++++1234(CrLf)
5/USFW/123456789(CrLf)
/USCH/123456(CrLf)

:53J:/GBSC/123456/CLRC/12345(CrLf)
xxxxx12345xxxxx12345+++++12345(CrLf)
/NAME/xxxxx(CrLf)
/ABIC/CHASUS33CHI(CrLf)

Invalid message: invalid code word at the beginning of line 3, because /NAME/ is split across lines 2 and 3.

:53J:/ABIC/CHASUS33XXX(CrLf)
/CLRC/12345xxxxx12345/NA(CrLf)
ME/xxxxxxxxxxxxx(CrLf)

Invalid messages: codes appearing in format 1, 2 and 3 must not be combined within the same field, even if both formats are (used separately) valid for this field.

(Error Code T78, invalid code word)

:53J:/ABIC/CHASUS33(CrLf)
/NAME/12345XXXXX12345xxxxx12345+++++1234(CrLf)
/SSIS/(CrLf)

(Error Code T30, too many lines)

:57J:/SSIS/(CrLf)
/NETS/(CrLf)

4.16 S.W.I.F.T. Payments Reject/Return

4.16.1 <REJT/RETN/72>

The <REJT/RETN/72> function will check the content of the field 72 based on the rules defined below.

<REJT/RETN/72> generic format: 6*35x

the generic rules that must be applied to the <REJT/RETN/72> field are:

- max 6 lines,
- max of 35 characters per line.

<REJT/RETN/72> specific format:

In addition to the generic format documented above, the content of the <REJT/RETN/72> fields must be validated as specified below.

This special function must be performed for the MTs where this format is formally specified in Part III Ch. 3 Field 72 specifications, and only if the first line begins with the code word '/REJT/' or '/RETN/'.

If these conditions are not met, then the other message format specifications must be followed.

```
IF <field '72' : subf 1> = ('/REJT/' OR '/RETN/')
     THEN PERFORM <REJT/RETN/72> SYNTAX VALIDATION
ENDIF
```

<REJT/RETN/72> format:

72	<REJT/RETN/72>	6!x 2!n [1a] ['/'2c] 'CRLF' ('/'2!c 2!n'/') | ('/X' 1c 2!n'/') [29x] (*1)	
		'CRLF' '/MREF/' 16x ['CRLF' '/TREF/' 16x] ['CRLF' '/CHGS/'<CUR><AMOUNT>15] ['CRLF' '/TEXT/' 29x ['CRLF' '//' 33x]0-2]	T80 invalid code word

(*1): Specific code words validation applied to <REJT/RETN/72>:

72			(6!x)		’/REJT/’, ’/RETN/’	
			(‘/’2!c 2!n’/’)		’/AC01/’,’/AC02/’, ’/AC03/’,’/AC04/’, ’/AC05/’,’/AC06/’, ’/AM01/’,’/AM02/’, ’/AM03/’,’/AM04/’, ’/AM05/’,’/AM06/’, ’/AM07/’,’/AM08/’, ’/AG01/’,’/AG02/’, ’/BE01/’,’/BE02/’, ’/BE03/’,’/BE04/’, ’/BE05/’,’/DT01/’ ’/MS01/’,’/PY01/’, ’/RC01/’,’/RC02/’, ’/RC03/’,’/RC04/’, ’/RF01/’,’/TM01/’	T80

4.16.2 <REJT/RETN/79>

The <REJT/RETN/79> function will check the content of the field 79 based on the rules defined below.

<REJT/RETN/79> generic format: 35*50x

the generic rules that must be applied to the <REJT/RETN/79> field are:

- max 35 lines,
- max of 50 characters per line.

<REJT/RETN/79> specific format:

In addition to the generic format documented above, the content of the <REJT/RETN/79> fields must be validated as specified below.

This special function must be performed for the MTs where this format is formally specified in Part III Ch. 3 Field 79 specifications, and only if the first line begins with the code word ’/REJT/’ or ’/RETN/’.

If these conditions are not met, then the other message format specifications must be followed.

```
IF <field ’79’ : subf 1> = (’/REJT/’ OR ’/RETN/’)
     THEN PERFORM <REJT/RETN/79> SYNTAX VALIDATION
ENDIF
```

<REJT/RETN/79> format:

79	<REJT/RETN/79>	6!x 2!n [1a] [‘/’2c] ‘CRLF’ (‘/’2!c 2!n’/’) | (‘/X’ 1c2!n’/’) [44x] (*2)	
		’CRLF’ ’/MREF/’ 16x [’CRLF’ ’/TREF/’ 16x] [’CRLF’ ’/CHGS/’<CUR><AMOUNT>15] [’CRLF’ ’/TEXT/’ 44x [’CRLF’ ’//’ 48x]0-31]	T80 Invalid code word

(*2): Specific code words validation applied to <REJT/RETN/79>:

79			(6!x)		’/REJT/’, ’/RETN/’	
			(‘/’2! c 2!n’/’)		’/AC01/’,’/AC02/’, ’/AC03/’,’/AC04/’, ’/AC05/’,’/AC06/’, ’/AM01/’,’/AM02/’, ’/AM03/’,’/AM04/’, ’/AM05/’,’/AM06/’, ’/AM07/’,’/AM08/’, ’/AG01/’,’/AG02/’, ’/BE01/’,’/BE02/’, ’/BE03/’,’/BE04/’, ’/BE05/’,’/DT01/’ ’/MS01/’,’/PY01/’, ’/RC01/’,’/RC02/’, ’/RC03/’,’/RC04/’, ’/RF01/’,’/TM01/’	T80

4.17 <CC> Country Code

The <CC> function will check the Country Code component.

The syntax of the component is : 2!a

The function will check if the Country Code consists of 2 alphabetic characters. The Country Code must be one of those defined in the BIC Directory in the section :

General Information “Country Codes”.

An invalid <CC> check will return an error code “T73”.

4.18 ERI (Euro Related Info) validation

Referring to <ERI-F...> for any fields where the ERI validation must be applied :

- the field generic format (syntax and conditional checks) must be validated first, ref. applicable sections in Part II, chapter 4, Part III, chapters 3 and 4, and Part IV, chapter 2,
- refer to : <• remove continuation line breaks:> and <• remove line breaks:> in each <ERI–F...> as documented below, and create the
  <ERI–F...> specific format.
  The code words /OCMT/ and /CHGS/, as well as their data content, may be split across lines (note: since the generic format is validated first, only the maximum number of lines defined in the generic format is allowed).
- with reference to the **<ERI-F...> specific format**, apply the **ERI code words format** and the **ERI validation rules**.

ERI code words format :

['/OCMT/'<CUR><NUMBER>15'/'['/CHGS/'<CUR><NUMBER>15'/']]

ERI validation rules :

- the code word /OCMT/ is not positional (floating), i.e. it may occur anywhere within the frame of the **<ERI–F...>** specific format,
- the code word /OCMT/ must not be used more than once anywhere in the **<ERI–F...>** specific format (error code "T47"),
- if the 6 character code word /OCMT/ is found in the **<ERI–F...>** specific format then the following data <CUR><AMOUNT>15'/' must be validated,
- the first occurrence of the code word /CHGS/ is identified only if it follows immediately after the code word /OCMT/ and its corresponding data, i.e. if it is recorded right after :
  '/OCMT/'<CUR><AMOUNT>15'/'
- if the 6 character code word /CHGS/ is identified, then the following data <CUR><AMOUNT>15'/' must be validated,
- if the code word /CHGS/ is identified (i.e. immediately after the code word /OCMT/ and data), then it must not be used again in the remaining portion of the field content (error code "T47").

ERI fields, generic & ERI-specific formats :

<ERI-F61> generic format : **34x** **(subfield 9 of field 61)**

- no need to remove characters

<ERI-F61> specific format : **34x**

<ERI-FLD72structured> generic format : **<INSTR>**

['CrLf'(<INSTR> | '//' 33x)]0-5

- remove continuation line breaks : remove all "CrLf//" characters
- remove line breaks : remove all "CrLf" characters

<ERI-FLD72structured> specific format : **210x**

<ERI-F72narrative> generic format : **35x['CrLf'35x]0-5**

- remove line breaks : remove all "CrLf" characters

<ERI-F72narrative> specific format : **210x**

<ERI-F77A> generic format : **35x['CrLf'35x]0-19**

- remove line breaks : remove all "CrLf" characters

<ERI-F77A> specific format : **700x**

<ERI-F79> generic format : **50x['CrLf'50x]0-34**

- remove line breaks : remove all "CrLf" characters

<ERI-F79> specific format : **1750x**

<ERI-F86> generic format : **65x['CrLf'65x]0-5**

- remove line breaks : remove all "CrLf" characters

<ERI-F86> specific format : **390x**

Examples :
to simplify this list of examples, all of them are documented with reference to the field-72 structured format.

Field 72 structured format :
<INSTR>
['CrLf'(<INSTR> | '//' 33x)]0-5

Valid examples :

:72:/OCMT/DEM123,/'CrLf' (/OCMT/ validated)
//BLABLABLAXXXX'CrLf'

:72:/OCMT/DEM123,50/'CrLf' (/OCMT/ validated)
///CHGS/BEF456,00/'CrLf' (/CHGS/ validated)
//AAAAAAAABBBBBBBB'CrLf'

:72:/CHGS/xxxx/OCMT/DEM123,'CrLf' (/CHGS/ not validated)
//50/BLABLABLAXXXX'CrLf' /OCMT/ validated)
/CHGS/BEF456,00/'CrLf' (/CHGS/ not validated)

:72: /CHGS/BEF456,00/'CrLf' //BLABLABLAXXXX'CrLf' /CHGS/BEF9876,00/'CrLf'	(/CHGS/ not validated) (/CHGS/ not validated)

Invalid examples ("T47") :

:72:/CHGS/xxxx/OCMT/DEM123,'CrLf' //50//CHGS/BEF456,00/'CrLf' /AA/BBBBBBBBBBBBB'CrLf' /CHGS/BEF456,00/'CrLf'	(/CHGS/ not validated) /OCMT/ & /CHGS/ validated) (double /CHGS/)
:72:/OCMT/DEM123,/'CrLf' ///CHGS/BEF678,50/BLABLABLAXX'CrLf'	(/OCMT/ & /CHGS/ validated)
//XXXXXXXXX/OCMT/DEM123,/xxx'CrLf'	(double /OCMT/)

4.19 <FLD72_not_STP>

Field 72 validation in MT102_not_STP and MT103_not_STP

The function Sender to Receiver (or <FLD72_not_STP>) validates the content of the field 72, according to the following syntax:

<FLD72>: <INSTR>
['CRLF'(<INSTR>|'//'33x)]0-5

Errors found by this validation function will be reported with the usual codes T12, T17, T30, T31, T32, T33, and T34.

The format <FLD72_not_STP> does not require any further validation, currently this format is only used in the MT103 without the tag119:STP. (implemented in Standards release 2000)

Note : the ERI format validation is not performed, but the usage of the ERI format is not rejected (contrary to <FLD72_STP> where the ERI format is not allowed).

4.20 <FLD72_STP>

Field 72 validation in MT102_STP and MT103_STP

The function Sender to Receiver (or <FLD72_STP>) validates the content of the field 72, according to the following syntax:

<FLD72>: <INSTR>
['CRLF'(<INSTR>|'//'33x)]0-5

Errors found by this validation function will be reported with the usual codes T12, T17, T30, T31, T32, T33, and T34.

In addition to the field 72 structured format validation (i.e. <FLD72>) the following validation must be performed:

- the ERI format is not allowed, i.e. the code "/OCMT/" is not allowed (anywhere in the text, not even across lines) error code "T82"
- the REJT/RETN format is not allowed, i.e. the first 6 character must not be ("/REJT/" | "/RETN/") error code "T81".

/INS/ format: ['/INS/'<SWIFTBIC> | <NON-SWIFTBIC>'CrLf']

- the code '/INS/' is optional
- it must not be used more than once ("T47"), but in order to be validated by the system it must be coded as the first 5 characters of a line
- if the code /INS/ is used at the beginning of a line, it must be validated according to the format: '/INS/'<SWIFTBIC> | <NON-SWIFTBIC>'CrLf'

error codes "T27, T28, T29, T44, T45, T46"

- consequently, if the code '/INS/' is used anywhere else in the text of field 72, the specific validation will not be performed

1. examples: code /INS/ validated and ACK

:72:/INS/CITIUS33CrLf

:72:/ACC/BLABLABLACrLf
/INS/CITIUS33MIACrLf

:72:/ACC/BLABLABLACrLf
//BLABLABLACrLf
/INS/CITIUS33MIACrLf

:72:/ACC/BLABLABLACrLf
//BLABLABLACrLf
/INS/CITIUS33MIACrLf
//BLABLABLACrLf

:72:/INS/CITIUS33CrLf
/ACC/BLABLABLACrLf
//BLABLA /INS/CITIUS33MIACrLf (not identified as double)

2. examples: code /INS/ validated and NACK

:72:/INS/CITIUS333CrLf (not a valid 8 or 11 char BIC)

:72:/ACC/BLABLABLACrLf
/INS/CITIUSCrLf (not allowed across lines)
//33MIACrLf

:72:/INS/CITIUS33/ACCD/AAACrLf (must be terminated by CrLf)

:72:/INS/CITIUS33CrLf
/ACC/BLABLABLACrLf
/INS/CITIUS33MIACrLf (double not allowed)

3. examples: code /INS/ not validated by the system (ACK)

:72:/ACCD/BLABLA/INS/CITIUS33XCrLf

:72:/ACC/BLABLABLACrLf
//ACCD/INS/CITIUS33MIA FLORIDACrLf

:72:/ACC/BLABLABLACrLf
//BLABLABLA/INS/CITIUCrLf
//S33MIA MIAMI FLORIDACrLf

:72:/ACC/BLABLABLACrLf
//BLABLABLA/INS/CITIUS33MIACrLf
//BLABLABLA/INS/CITIUS33MIACrLf

:72:/ACC/BLABLABLACrLf
///INS/CITIUS33MIA MIAMICrLf (not the beginning of a line)

:72:/ACC/BLABLABLACrLf
//BLABLABLACrLf
/INS/CITIUS33MIACrLf (this /INS/ code is validated)
//BLA/INS/CITIUS33MIA FLCrLf (this /INS/ code is not validated)

4.21 <SIGN>

The **<SIGN>** function checks the sign component value: '+' or '-'.

Standard presentation: **<SIGN>**
format: 1!x

The code error "T15" is returned if the <SIGN> component is not '+' or '-'.

4.22 <OFFSET>

The **<OFFSET>** function checks the time component, it is similar to <HHMM> function where the validation is as follows:

00<=HH<=13

00<=MM<=59
Standard presentation: <HHMM>
format: 4!n

The code error "T16" is returned if the <OFFSET> component is invalid.

4.23 <IBAN>

This function validates the format and the syntax of an IBAN (International Bank Account Number), the following specs are reproduced from the ISO Standards documents: ref. ISO 13616:1997 (E), First edition 1997-10-01.

<IBAN> may be required in the following fields letter options:

Refer Part IV, Rule 119

59		'/'<IBAN>'CRLF'35x['CRLF'35x]0-3	
59A		'/'<IBAN>'CRLF' <SWIFTBIC>|<NON-SWIFTBIC>	

Format & syntax validation.

IBAN Structure: 2!A2!n1!B[B]0-29

Or: 2 alphabetic characters followed by 2 numeric characters, followed by 1 or up to maximum 30 alphanumeric characters.

Alphabetic characters may be coded upper or lower case.

New character type:
A : alphabetic character, upper or lower case
B : alphanumeric character, alphabetic character may be upper or lower case

2!A must be a valid country code, refer <CC> function, error code "T73".
Exception to the standards Country Code notation: in an IBAN the country code may contain lower case.

Checksum validation.
1. Move the first four characters to the right-hand end of the account number.
2. Convert letters to digits in accordance with the following:

A=10	F=15	K=20	P=25	U=30
B=11	G=16	L=21	Q=26	V=31
C=12	H=17	M=22	R=27	W=32
D=13	I=18	N=23	S=28	X=33
E=14	J=19	O=24	T=29	Y=34
				Z=35

The numeric values are the same whether using upper or lower case alphabetic characters.
3. Apply the check character system MOD 97-10 (ref. ISO 7064).
4. If the remainder of the division is 1 (one) then the number (i.e. the IBAN) is valid, otherwise <Error code D19>.

Example: to validate field :59:/BE88320034713441<CrLf>… do the following:
Extract BE88320034713441 from field 59
1. move BE88 to the right: 320034713441BE88
2. convert alpha char. to numeric char.: 320034713441111488
3. calculate the modulo 97 (remainder after division by 97).
The remainder of 320034713441111488 divided by 97 is 1 (one).
Result: the check digits of this IBAN are correct.

Chapter 5
Character Sets Validation

5.1 <X> Function

The function <X> checks if characters of a component belong to the X S.W.I.F.T. character set.

Moreover, there must be:

1. length of the component consisting of characters defined in the S.W.I.F.T. character set
2. check for meaningless lines, fields, sub-fields, and components: lines, fields, sub-fields, and components containing only blank(s) or only a 'CrLf' are not allowed, with the exception of some special fields, like the 1st subfield of 77E. (refer to Part III Chapter 2 for field rules and character restrictions)
   The following are all valid examples of field '77E':
   :77E:eee'CrLf' 1st subfield contains only blanks
   ABCDXYZ'CrLf'
   :77E:'CrLf' 1st subfield = empty line
   ABCDXYZ'CrLf'
   :77E::'CrLf' 1st subfield begins with ':'
   :'CrLf' 2nd subfield begins with ':'
   ABCDXYZ'CrLf'
   :77E:-'CrLf' 1st subfield begins with '-'
   :'CrLf' 2nd subfield begins with ':'
   ABCDXYZ'CrLf'
3. check for a spurious 'Cr' or 'Lf': each 'Cr' detected must be followed by a 'Lf', and each 'Lf' preceded by a 'Cr'.

note : the "CrLf" pair of characters may only be used together as "end-of-line" AND "begining-of-line", it is NOT actually part of the <x> character set which defines the valid characters of the field components. The number of "CrLf" determines the number of lines in the text block, or the number of lines in an individual field.

e.g. : in field ":20: 16x", the characters "CrLf" are NOT allowed.

X S.W.I.F.T. Character Set	Code
Alphabetical Characters	
A to Z (upper case)	EBCDIC
a to z (lower case)	EBCDIC
Numeric Characters	
0 to 9	EBCDIC
Special Characters	
/ - ? : ( ) . , ' + SPACE CrLf	EBCDIC

5.2 <Y> Function

The function <Y> checks if characters of a component belong to the Y S.W.I.F.T. character set (dedicated to the specific EDI service).

Y S.W.I.F.T. Character Set	Code
Alphabetical Characters	
A to Z (upper case)	EBCDIC
Numeric Characters	
0 to 9	EBCDIC
Special Characters	
SPACE . , - ( ) / = ' + : ?	EBCDIC
Special Characters (incompatible with international telex)	
! " % & * ; < >	EBCDIC

5.3 <Z> Function

The function <Z> checks if characters of a component belong to the Z S.W.I.F.T. character set.

Z S.W.I.F.T. Character Set	Code
Alphabetical Characters	
A to Z (upper case)	EBCDIC
a to z (lower case)	EBCDIC
Numeric Characters	
0 to 9	EBCDIC
Special Characters	
. , - ( ) / = ' + : ? @ # Cr Lf SPACE {	EBCDIC
Special Characters (incompatible with international telex)	
! " % & * ; < >	EBCDIC

note : in the <z> character set, the characters "Cr" and " Lf" may be used together or separately, unless the char set <z> is used to define a field format with several lines, example fields 29F and 29G, in those 2 fields any "CrLf" will be interpreted as a "line-separator" per the field format definition, ref. Part III, Chap 3, Field 29.

REMARK :
In all FIN messages, whatever the character set being used, the combination "CrLf-" indicates always the end the message text.

5.4 Hexadecimal values :

of the <x>, <y>, <z> S.W.I.F.T. character sets :
The hexadecimal representation of the characters belonging to the S.W.I.F.T. character sets is documented in the table below. The first digit of the hexadecimal value is the column id, and the second digit is the row id. eg. character "*" = "5C".

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0					Sp	&	-						{			0
1							/		a	j			A	J		1
2									b	k	s		B	K	S	2
3									c	l	t		C	L	T	3
4									d	m	u		D	M	U	4
5			Lf						e	n	v		E	N	V	5
6									f	o	w		F	O	W	6
7									g	p	x		G	P	X	7
8									h	q	y		H	Q	Y	8
9									i	r	z		I	R	Z	9
A								:								
B					.		,	#								
C					<	*	%	@								
D	Cr				(	)		‘								
E					+	;	>	=								
F					!		?	“								

Note : the character "!" is coded "4F" according to the Unisys EBCDIC representation, which is different from the IBM hexadecimal representation "5A".
Cr = Carriage return
Lf = Line feed
Sp = Space (printed as a blank character)

Chapter 6
Message User Group Text Validation Rules

The MUG-textval rules are defined and performed within the frame of the MUG service profiles definition. These rules are performed in addition to the generic message standards, therefore the message special format (defined in these rules) must be a subset of the standard message format. For example (referring to the Australian PDS service below), the AU/PDS selected field format option "A" is:

«'//AU'6!n'CRLF'<SWIFTBIC>|<Non-SWIFTBIC>»,

this format is a subset of the generic format:

«['/'<DC>]['/'34x]['CRLF']<SWIFTBIC>|<Non-SWIFTBIC>».

"Gnn" type error codes have been defined for the MUG rules text-validations (ref. Part IV, Chapter 4).

The "Selection rules" (Snn) are provided for documentation purpose, the main reason being to indicate in which case a particular profile will be "DE-selected" (i.e. because the validation flag -tag119- is not present or does not contain the appropriate code word).

ref. UHB publications:

FIN System Messages: 2.7 User Header in FIN Messages, Format.

FIN System Messages: 5.3 List of Tags, Names and Attributes: Tag 119.

FIN Service Description: 5.8 User Header Block.

6.1 Summary of MUG services and related textval rules:

AU/PDS	Australian PDS service, FIN-Copy service and MUG Validation rules V06, ~~V15~~V19 tag103:PDS or PDT
LCH	London Clearing House, ACCORD for LCH M-Copy V02 tag103:LCH
DER	London Clearing House, ACCORD for DER M-Copy V18 tag103:DER
TPS	Third Party Service, M-Copy to ACCORD for TPS messages S20, P02 tag103:TPS
LVTS	Canadian Large Value Transfer System, FIN-Copy service and MUG Validation rule V07

tag103:CAD

REMIT — Extended Remittance Information, special validation (MT103)
V08, S08
tag119: REMIT

<S05> — (de)-selection rule
S05 (applied only to FIN-Copy MUG)

RFDD — Request For Direct Debit, special validation (MT104)
V03, S03
tag119: RFDD

CLSB — Continuous Link Settlement, FIN-Copy service and MUG Validation rules
V10, V11, V12, V16, V17, S01, S05, P01
tag103:CLS or CLT

Tag119 usage :

The tag119: must be specified in FIN messages where a special text validation needs to be performed, the users who want to use the tag119: must register for the MUG defined for that service. A special MUG where the tag119: is defined will not be used to validate a message where the tag119: is not present, or where the tag119: contains another value other than the one defined for that MUG.

The Tag119 presence and values in FIN MTs are:

Message Type	Tag119 Presence	Tag119 Value
102	Optional	STP
103	Optional	STP REMIT
104	Optional	RFDD
503 504 505 506 507	Mandatory	4!c (ref. C94)
574	Mandatory	W8BENO IRSLST
Other MTs	Not allowed	N/A

Note: tag119 is not repeatable (ref. FIN System Message UHB), if present it must contain only one code word, therefore the codes STP and REMIT can never be used at the same time in the MT 103.

Generic validation for tag 119:

```
IF TAG119_present

THEN begin

IF MT = 102
     THEN IF TAG_119 = 'STP'
     THEN continue
     ELSE <error:'U08'> (invalid combination/value MT-tag119)

ELSE IF MT = 103
     THEN IF TAG_119 = 'STP' | 'REMIT'
     THEN continue (ref. MUG rule V08 / REMIT)
     ELSE <error:'U08'> (invalid combination/value MT-tag119)

ELSE IF MT = 104
     THEN IF TAG_119 = 'RFDD'
     THEN continue (ref. MUG rule V03 / RFDD)
     ELSE <error:'U08'> (invalid combination/value MT-tag119)

ELSE IF MT = 574
     THEN IF TAG_119 = 'W8BENO' | 'IRSLST'
     THEN continue
     ELSE <error:'U08'> (invalid combination/value MT-tag119)

ELSE IF MT = 503 | 504 | 505 | 506 | 507
     THEN continue      (ref. C94 for tag119 validation)

ELSE <error:'U09'> (tag119 is not allowed in this MT)
     End_begin

ELSE begin

IF MT = 503 | 504 | 505 | 506 | 507 | 574
     THEN <error:'U08'> (tag119 is missing in this message)
     ELSE continue
     End_begin
```

ref. Validation Rules: V03, V08 and Conditional Rule: 94.

Here after, a short description of the MUG's defined for the special tag:119 validations.

MUG id.	nnn	nnn
Name	REMIT	RFDD
description		
service		
statistics		
Addserv		
Emitter category	catREMIT	catRFDD

Receiver category	catREMIT	catRFDD
FIN msg types	103	104
excluding		
Emitter branch code		
Receiver branch code		
Validation Rules	08	03
Selection Rules	08	03
Processing Rules		

6.2 AU/PDS

Specific field validation checks must be performed on the MT103 and 202 selected for the AU/PDS and AU/PDT FIN-Copy services.
The AU/PDS enhancement implemented in November 2006 allows option "C" in the group of fields 56a and 57a. If field 56C or 57C is the first field of the group, it must be validated with the format: "//AU"6!n, otherwise the generic format ("/"34x) must be validated.

Ex: :56C://AU120456

MT103:
The first field of the group: 56a, 57a, must be used with the letter option "A", or "C", or "D", and must be validated according to the format defined below, this requirement applies only to the first field.
At least one field of the group: 56a, 57a, must be present in the message.

MT202:
The first field of the group: 56a, 57a, 58a, must be used with the letter option "A" or "D", and must be validated according to the format defined below, this requirement applies only to the first field.
At least one field of the group: 56a, 57a, 58a, must be present in the message.

Specific fields formats for AU/PDS, AU/PDT.
Fields 56A, 57A, 58A (only first field):
'//AU'6!n'CRLF'<SWIFTBIC>|<Non-SWIFTBIC>
Fields 56C, 57C (only first field):
'//AU'6!n
Fields 56D, 57D, 58D(only first field):
'//AU'6!n'CRLF'35x['CRLF'35x]0-3

V06 (in MT202, apply field 56, 57 or 58 special validation)

```
IF <message'202'>
     THEN VALIDATE_SELECTED_FIELD
ENDIF
```

Note: for T&T purpose, prior to cutover, a new Message User Group must be defined for AU/PDS, during the Standards T&T period, rule V19 must be defined in the T&T MUG. At cutover of the AU/PDS enhancement (November 2006), V15 must be replaced by V19 in the LIVE Message User Group for AU/PDS.

V19 (in MT103, apply field 56 or 57 special validation, per Nov. 2006 enhancement)

```
IF <message'103'>
     THEN VALIDATE SELECTED FIELD 2006
ENDIF
```

VALIDATE SELECTED FIELD 2006 special validation procedure.
BEGIN

```
IF <message'103'>
     THEN IF <field'56*'>
          THEN SF* := <field'56*'>
          ELSE IF <field'57*'>
               THEN SF* := <field'57*'>
               ELSE SF* := NULL (no field 56a, 57a in this msg)
               ENDIF
          ENDIF
     ELSE IF <message'202'>
          THEN IF <field'56*'>
               THEN SF* := <field'56*'>
               ELSE IF <field'57*'>
                    THEN SF* := <field'57*'>
                    ELSE IF <field'58*'>
                         THEN SF* := <field'58*'>
                         ELSE SF* := NULL  (invalid situation because
                         field 58a is mandatory in MT202. This MT should
                         have been NAK)
                         ENDIF
                    ENDIF
               ENDIF
          ENDIF
     ENDIF
```

```
IF SF* = NULL
     THEN <error 'G04'>                    (no selected field present)
     ELSE IF <SF 'A'>                      (check letter option 'A')
          THEN VALIDATE SF A
               IF OK
               THEN continue
               ELSE <error 'G02'>          (invalid format 'A')
               ENDIF
          ELSE IF <SF 'D'>                 (check letter option 'D')
               THEN VALIDATE SF D
                    IF OK
                    THEN continue
                    ELSE <error 'G03'>     (invalid format 'D')
                    ENDIF
          ELSE IF <SF 'C'>                 (check letter option 'C')
               THEN VALIDATE SF C
                    IF OK
                    THEN continue
                    ELSE <error 'G18'>     (invalid format 'C')
                    ENDIF
               ELSE <error 'G01'>          (invalid letter option)

               ENDIF
          ENDIF
     ENDIF
ENDIF
```

END VALIDATE_SELECTED FIELD_2006

~~V15 (in MT103, apply field 56 or 57 special validation)~~

```
~~IF <message'103'>~~
     ~~THEN VALIDATE_SELECTED_FIELD~~
~~ENDIF~~
```

VALIDATE_SELECTED_FIELD special validation procedure.
BEGIN

```
IF <message'103'>
      THEN IF <field'56*'>
            THEN SF* := <field'56*'>
            ELSE IF <field'57*'>
                  THEN SF* := <field'57*'>
                  ELSE SF* := NULL (no field 56a, 57a in this msg)
                  ENDIF
            ENDIF
      ELSE IF <message'202'>
            THEN IF <field'56*'>
                  THEN SF* := <field'56*'>
                  ELSE IF <field'57*'>
                        THEN SF* := <field'57*'>
                        ELSE IF <field'58*'>
                              THEN SF* := <field'58*'>
                              ELSE SF* := NULL  (invalid situation because
                              58a                is mandatory in generic
                              MT202)
                              ENDIF
                        ENDIF
                  ENDIF
            ENDIF
      ENDIF
ENDIF

IF SF* = NULL
      THEN <error 'G04'>                        (no selected field present)
      ELSE IF <SF 'A'>                          (check letter option 'A')
            THEN VALIDATE_SF_A
                  IF OK
                  THEN continue
                  ELSE <error 'G02'>            (invalid format 'A')
                  ENDIF
            ELSE IF <SF 'D'>                    (check letter option 'D')
                  THEN VALIDATE_SF_D
                        IF OK
                        THEN continue
                        ELSE <error 'G03'>      (invalid format 'D')
                        ENDIF
                  ELSE <error 'G01'>            (invalid letter option)
                  ENDIF
            ENDIF
      ENDIF
ENDIF
```

END VALIDATE_SELECTED FIELD

Selected Field Format: Fields Tags 56A, 57A, 58A VALIDATE_SF_A: «'//AU'6!n'CRLF'<SWIFTBIC>|<Non-SWIFTBIC>»

Selected Field Format: Fields Tags 56C, 57C VALIDATE_SF_C: «'//AU'6!n»

Selected Field Format: Fields Tags 56D, 57D, 58D VALIDATE_SF_D: «'//AU'6!n'CRLF'35x['CRLF'35x]0-3»

6.3 LCH

If a FIN user-to-user message contains the 'London Clearing House' code in the tag103, it must be exchanged between members of the LCH service (i.e. both emitter and receiver must be members), otherwise the message is rejected.

V02

```
IF <tag103>                          (if tag103 is present)
THEN IF <tag103>="LCH"               (if tag103 contains "LCH")
     THEN IF NOT V02
          THEN <error:"B03">
          ELSE continue
          ENDIF
     ELSE continue
     ENDIF
ELSE continue
ENDIF
```

(note : the LCH category is assigned only to ACCORD_MEMBERS)

6.4 LVTS

If 2 LVTS members (i.e. emitter & receiver are LVTS subscribers) exchange a FIN message type (103 or 205), AND the first 6 characters of their SWIFTBIC are different, AND the currency code used in the tag 32A is "CAD", then the tag 103 must be present in the User Header AND it must contain the code "CAD".

V07

```
IF V07 THEN

IF (<message'103'> OR <message'205'>)
     THEN IF <field32A:currency_code> = "CAD"
          THEN IF (tag 103 NOT present OR tag 103 NOT= "CAD")
               THEN IF (emitter_SWIFTBIC_1_to_6 NOT=
               receiver_SWIFTBIC_1_to_6)
                    THEN <error:"G05">
                    ELSE continue
                    ENDIF
               ELSE continue
               ENDIF
          ELSE continue
          ENDIF
     ELSE continue
     ENDIF
ENDIF
```

6.5 REMIT

The usage of the code word "REMIT" in the tag119 is limited to the exchange of the MT103 between subscribers of the special MUG for 'extended MT103' (i.e. the 'extended MT103' contains the field 77T in the text of the message). And vice versa, the usage of the tag 77T in the text of the MT103 is limited to the exchange between subscribers of the 'extended MT103' with in the User Header the tag119 containing the code word "REMIT".
If the tag 119: REMIT is used in a FIN message, but the MUG_ REMIT is not selected (i.e. V08 is off) then the message is rejected with the error code "U09".
If the MUG_ REMIT is selected (i.e. S08 is ON), but the tag 119 is not present, or the tag 119 is present but does not contain "REMIT", then the MUG_ REMIT is deselected.

IF the tag 119:REMIT is present then the MUG_REMIT must be selected.

```
IF <tag119>

THEN IF <tag119> = "REMIT"
     THEN  IF NOT V08
          THEN <error:"U09">
          ELSE continue
          ENDIF
     ELSE continue
     ENDIF

ELSE continue
ENDIF
```

S08

```
IF S08

THEN IF (NOT <tag119>)
     OR (<tag119> NOT= "REMIT")
     THEN DE_SELECT_REMIT_MUG
     ELSE continue
     ENDIF

ELSE continue
ENDIF
```

V08

```
IF V08

THEN IF <tag77T>
     THEN continue
     ELSE <error:”G06”>
     ENDIF

ELSE IF <tag77T>
     THEN <error:”G06”>
     ELSE continue
     ENDIF
ENDIF
note : the check for V08 applies only in the context of the MT103
```

6.6 S05 (de-)selection rule

The rule S05 may be applied to any MUG profile where the tag 103 (any FIN-Copy VAS code) is required. In the situation where a MUG user profile (with S05 defined) is selected but there is no tag 103 present in the FIN message, or the tag103 VAS code is not= the MUG VAS code, then the MUG profile is deselected.

S05

```
<for any selected profile>

IF Profile_Pn_selected
     THEN IF S05_defined_in_Pn
           THEN IF Tag103_NOT_present
                (OR Tag103_VAS NOT= Profile_Pn_VAS) (*)
                     THEN Deselect_Pn
                     ELSE continue
                ENDIF
           ELSE continue
           ENDIF
     ELSE continue

ENDIF
```

(*) this checks is applied per the generic MUG selection: a MUG profile where a FIN-Copy VAS code is defined will not be selected if in the FIN message the tag 103 contains a different VAS code.

Please note that this rule does not apply to the FINinform VAS codes (nor M-Copy VAS), those codes must never be used in the tag103, hence a MUG profile where a FINinform VAS code (or M-Copy VAS) is defined will remain selected. A FINinform VAS code (or M-Copy VAS) is defined in a MUG profile in order to trigger a VAS copy without the need for a tag103 in the message. This is sometime referred to as a ‘transparent’ copy.

6.7 RFDD

The usage of the code word "RFDD" in the tag119 is limited to the exchange of the MT104 between subscribers of the special MUG for “Request For Direct Debit MT104”.
If the tag 119:RFDD is used in a FIN message, but the MUG_RFDD is not selected (i.e. V03 is off) then the message is rejected with the error code “U09”.
If the MUG_RFDD is selected (i.e. S03 is ON), but the tag 119 is not present, or the tag 119 is present but does not contain “RFDD”, then the MUG_RFDD is deselected.

IF the tag 119:RFDD is present then the MUG_RFDD must be selected.

```
IF <tag119>

THEN IF <tag119> = "RFDD"
     THEN  IF NOT V03
           THEN <error:"U09">
           ELSE continue
           ENDIF
     ELSE continue
     ENDIF

ELSE continue
ENDIF
```

S03

```
IF S03

THEN IF (NOT <tag119>)
     OR (<tag119> NOT= "RFDD")
     THEN DE_SELECT_RFDD_MUG
     ELSE continue
     ENDIF

ELSE continue
ENDIF
```

V03

The special validation checks for tag119:RFDD, present or not present, are documented in the Conditional Rules 75, 94 and 96 (scan for "RFDD").

V03 is synonymous with : "tag119:RFDD present".

6.8 CLSB

The MT300's candidates for CLS (or CLT) FIN-Copy service require additional validation. When a MT300 is selected for CLS (or CLT) FIN-Copy service process, the MUG validation rules V10, V11, V12 and V16 are performed, even if the message is not copied to CLSB (ref. CLS FIN-Copy FRS).

The MUG's CLSMEMB-to-CLSSERV, and CLTMEMB-to-CLTSERV are implemented for MT304, the validation rule V17 is defined in these 2 MUG's.

S01

The selection rule S01 is used as a flag to indicate that the MUG is selected, ref. P01 below.

V10

```
IF V10
     THEN IF MT300
          THEN IF <any field 53, in Seq. B (i.e. index 18, 22), does NOT
          have letter option "A">
               THEN <error:"G07">
```

V11

```
IF V11
     THEN IF MT300
          THEN IF <any field 57, in Seq. B (i.e. index 20, 24), does NOT
          have letter option "A"> OR < field 57A, in Seq. B1 (i.e. index
          20), does NOT contain "CLSB****">
                    THEN <error:"G08">
```

V12

```
IF V12
     THEN IF MT300
          THEN IF <tag17U_present> (i.e. index 8),
               THEN IF tag17U NOT= "N"
                    THEN <error:"G09">
```

V16

```
IF V16
     THEN IF MT300
          THEN BEGIN
               IF any field 56a, (index 19, 23) is present in seq. B
               and does NOT have letter option "A"
               THEN <error_G10>
               END_BEGIN
```

V17

```
IF V17
     THEN IF MT304
          THEN BEGIN
          IF field 94A index 5 in seq A NOT= "ASET"
          THEN <error_G13>
          IF field 53a index 17 in seq B1 does NOT have letter option
          "A" THEN <error_G16>
          IF field 53a index 21 is present in seq B2 and does NOT have
          letter option "A" THEN <error_G16>
          IF (field 57a index 19 in seq B1 is NOT PRESENT) OR (does NOT
          have letter option "A") THEN <error_G17>
          IF field 57a index 23 in seq B2 does NOT have letter option
          "A" THEN <error_G17>
          IF (field 57a index 19 in seq B1 NOT="CLSB")
          OR (field 57a index 23 in seq B2 NOT="CLSB")
          THEN <error_G17>
          END_BEGIN
```

P01

```
IF P01
     THEN IF S01
          THEN BEGIN
               DO_NOT_PERFORM_FIN-COPY_TO_THIS_TAG103_VASCODE
               ADD_VASFE_INTERFACE_IN_COPY_TO_ACCORD
               END
```

In the framework of the FIN-Copy service CLSB, the selection rule P01 is defined in a MUG defined for "CLAMEMB-to-CLAMEMB MT300 | MT392", the processing rule S01 is defined in the MUG defined for the FIN-Copy CLS, note: S01 is not defined in the MUG FIN-Copy CLT. A message MT300, or MT392, which is a candidate for FIN-Copy CLS and has also selected the MUG "CLAMEMB-to-CLAMEMB MT300 | MT392" (where P01 is defined) will not be copied to the CLSB server.

This exception is requested because the message is copied to ACCORD VASFE (because emitter & receiver are ACCOMEMB, and message type is an ACCORD type MT). An additional interface is added by VASFE to indicate to ACCORD that the FIN-Copy process has not been performed, although the message contains a field tag 103:CLS (ref FIN-Copy for CLS FRS v2.2).

Assumption: the category CLAMEMB is only assigned to users who have the category ACCMEMB, this is maintained via COS.

6.9 TPS

If a FIN user-to-user message contains the 'Third Party Service' code 'TPS' in the tag103, it must be exchanged between members of the TPS service (i.e. both emitter and receiver must be members), otherwise the message is rejected, this validation is implemented via the MUG definition process. (Error code B03)

TPS (similar to LCH) is not a FIN-Copy, there no FIN-Copy service profile for TPS, the messages with the tag103:TPS must be validated via a MUG where the VAS ACCORD is specified. The PAC trailer is mandatory in the TPS messages (Error code Z09 PAC missing), the PAC trailer is copied to ACCORD but it is removed before delivery to the intended receiver.

After processing and additional validation by ACCORD, the TPS messages will be copied by ACCORD to the Settlement Member bank (note: the PAC trailer is calculated between the pair Emitter-SettlementBank).

The Rule S20 is defined in the MUG for TPS messages, the rule validates the messages for tag103:TPS and PAC trailer, if the tag103:TPS is not present the MUG is deselected.

The Rule P02 is defined in the MUG for TPS messages; the rule is stored in the preamble to indicate the required additional processing to be done by the ORP, i.e. removal of the PAC trailer before delivery to the Receiver.

S20

```
IF <tag103>                          (if tag103 is present)
THEN IF <tag103>='TPS'               (if tag103 contains 'TPS')
     THEN IF PAC_present
          THEN Copy to ACCORD, Remove PAC, Send to addressee
          ELSE NAK_Z09
          ENDIF
     ELSE deselect_this_MUG    (tag103 NOT= 'TPS')
     ENDIF
ELSE deselect_this_MUG  (tag103 NOT present)
ENDIF
```

P02

```
IF (P02)
THEN Store_P02_in_Preamble
```

If (P02) is true, it indicates that the MUG TPS has been selected.

The purpose of Production Rule P02 is to have the ORP remove the PAC trailer from the message before it is delivered to the intended addressee.

Note: if the MUG TPS has been selected based on the emitter category, receiver category and MT, but then the MUG has been deselected because of the Rule S20 (tag 103 not present, or tag 103 not=TPS) then the condition (P02) is not true.

6.10 DER

If a FIN user-to-user message contains the code “DER” in the tag103, it must be exchanged between members of the “Accord Third Party/Derivative Central” service, otherwise the message is rejected

This check (like for LCH), must be applied to all FIN messages which contain a field tag 103 in the User Header; the objective is to check that when the tag 103 is present and contains the code “DER” then the special MUG with the rule V18 must have been selected.

V18

IF <tag103> (if tag103 is present)
THEN IF <tag103>= “DER” (if tag103 contains “DER”)
THEN IF NOT V18 (rule V18 is not in any MUG selected by this msg)
THEN <error:”B03”>
ELSE continue
ENDIF
ELSE continue
ENDIF
ELSE continue
ENDIF

note : the DER category is assigned only to ACCORD_MEMBERS, because those messages must be copied to ACCORD

6.11 Accord M-Copy

The selection of a FIN message for Copy to ACCORD is based (1) on the MUG selection (a MUG must contain the VAS code “ACR”), (2) on the Message Type (it must belong to a hard coded list), and (3) both the sender and the receiver must not be an ACCORD server destination. The M-Copy selection is not based on the presence and value recorded in the field tag 103 of the user header; the following conditions must be met to trigger the M-Copy mechanism for ACCORD:

IF MUG_ACCORD is selected % VAS code in MUG = ACR %

AND (MT = 300 | 305 | 320 | 330 | 340 | 341 | 360 | 361 | 362 | 392 | 395 | 396 | 399)

AND emitter_BIC4 NOT= “ACCO”

AND receiver_BIC4 NOT = “ACCO”

THEN Set_Copy_to_ACCORD

ELSE continue

The LCH, TPS and DER transactions are also copied to ACCORD, but they carry a field tag 103 in the user header with the 3 character value LCH, TPS or DER; however the codes LCH, TPS and DER are not generic FIN-Copy transactions (there is no corresponding FIN-Copy service profiles).

The TPS transactions must have a PAC trailer, the LCH and DER must not.

Since these transactions are not generic FIN-Copy transactions we must validate the PAC trailer based on the MUG validation rules, selection rules, and processing rules.

The following pseudo code documents the validations that is applied to the FIN transactions that contain a field tag 103 with a code LCH, TPS, or DER:

```
IF Tag103 = (TPS | LCH | DER)

THEN BEGIN
     IF Tag103 = TPS
     THEN IF RuleS20 is present
          THEN IF PAC is present
               THEN Set_Copy_to_ACCORD
               ELSE Error_Z09
          ELSE Error_B03
     ELSE IF PAC is present
          THEN Error-B01
          ELSE IF ((Tag103 = LCH AND Rule_V02 is present)
               OR (Tag103 = DER AND Rule_V18 is present))
               AND (MT = (340 | 341 | 360 | 361 | 362 | 392))
               THEN Set_Copy_to_ACCORD
               ELSE Error_B03

END_BEGIN

ELSE continue
```