User-defined Functions User Guide

Release 1.2

November 2013

IKAN Solutions N.V.
Kardinaal Mercierplein 2
B-2800 Mechelen
BELGIUM

Table of Contents

CHAPTER 1
Introduction

User-defined Functions are used in MetaMap when processing logic can be specified. A User-defined Function is always part of an MSL assignment statement:

```
<MetaSuite Field> = USER-FUNCTION '<User-defined Function Name>' [ ( { <Field-name-n> | <Literal-n> } , ... ) ]  (n = 1 to 10)
```

User functions developed by IKAN will be put in separate tables, and can be called by means of the following MSL syntax:

```
<MetaSuite Field> = SYSTEM-FUNCTION '<User-defined Function Name>' [ ( { <Field-name-n> | <Literal-n> } , ... ) ]  (n = 1 to 10)
```

The name of the User-defined Function is a string of max. 22 characters, containing no spaces, and put between single quotes. A maximum of ten parameters can be listed between left and right parenthesis, separated by a comma. The mandatory parameters precede the optional parameters. If no parameters are needed, parenthesis are superfluous. Optional parameters following missing optional parameters are not supported; all preceding parameters need to be put in the parameter list.

Examples:

```
W-TEXT = 'The quick brown fox jumps over the lazy dog'
W-TEXT2 = USER-FUNCTION 'RIGHT' ( W-TEXT , 5 )
DEBUG 'The last 5 letters are: #' ( W-TEXT2 )
```

--> The result will be: The last 5 letters are: y dog

```
T01-DATE = SYSTEM-FUNCTION 'ADD-SUB-FIXED' (T01-DATE)
```

The first parameter is mandatory

```
T01-DATE = SYSTEM-FUNCTION 'ADD-SUB-FIXED' (T01-DATE, W-OPER)
```

The second parameter is optional

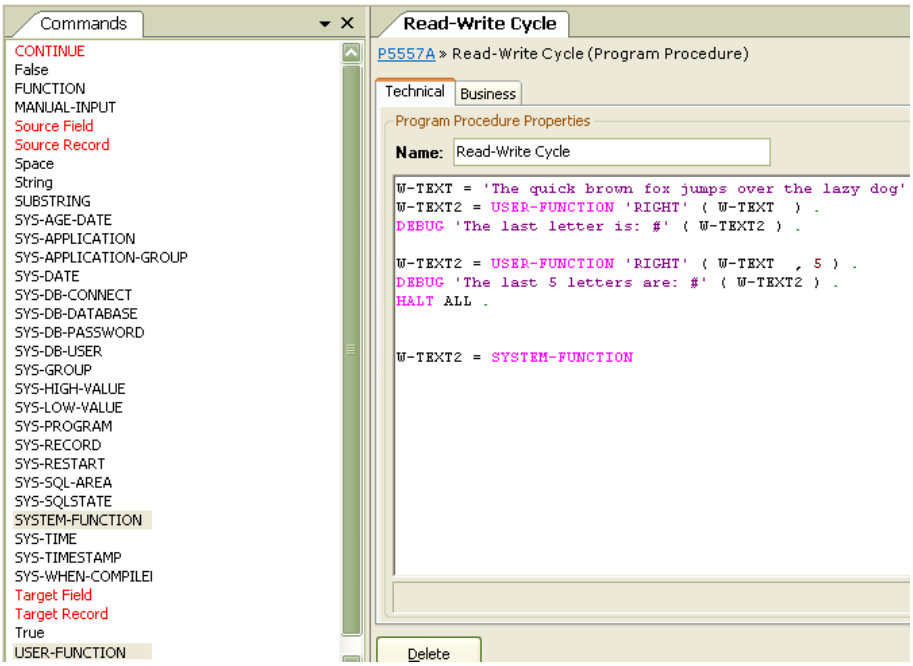
```
T01-DATE = SYSTEM-FUNCTION 'ADD-SUB-FIXED' (T01-DATE, W-OPER, W-DAYS)
```

The third parameter is optional

```
T01-DATE = SYSTEM-FUNCTION 'ADD-SUB-FIXED' (T01-DATE,       , W-DAYS)
```

This syntax form is not supported yet.

The following figure shows the appearance in the Structured Editor.



1.1. Building steps for making your own USER-FUNCTION

The description of a User Function is entirely done in the MetaSuite Dictionary (MSMGL.ddl).

The MetaSuite Dictionary contains a collection of tables. The User can add his own tables in the Dictionary using the Generator Manager. A number of rules govern the definition of User-defined Functions.

Naming conventions for the Dictionary tables implementing the User-defined Functions:

- The table “FCT000” contains the descriptions of the USER-FUNCTIONS
- The corresponding table “FCT010” contains the descriptions of the SYSTEM-FUNCTIONS
- The table called “FCT001” contains global Working-Storage definitions to support all USER-FUNCTIONS. The User is free to add data declarations.
- The corresponding table for the SYSTEM-FUNCTIONS is called “FCT011”. This table is reserved for IKAN.
- A maximum of 1000 tables containing the COBOL data declarations and the logic of the user functions are called FCU and are numbered between 000 included and 999 included (e.g., FCU099).
- A maximum of 1000 tables containing the COBOL data declarations and the logic of the system functions are called FCS and are numbered between 000 included and 999 included (e.g., FCS001).
- A maximum of 1000 tables containing the COBOL data declarations and the logic of the routines used by the functions are called FCR and are numbered between 000 included and 999 included (e.g., FCR456).
- A maximum of 1000 tables containing the COBOL logic that can be included in any other table while using the MTL CALL statement (e.g., FCC004).

  FCC001, FCC002, and FCC003 are already provided by IKAN.

One of the purposes of the FCT000 and FCT010 tables is to provide a bi-directional conversion between the “external" name of the function and the "internal" number of the function. The “external name” is the name used in MetaMap (for example “VISA-CARD”). The internal number can be found in the name of the Dictionary table. If the number belongs to a system function then the table name will start with "FCS", followed by the 3-digit function number. If the number belongs to a user function then the table name will start with "FCU", followed by the 3-digit function number.

Note: Routines do not need to be declared in the FCT000 table.

CHAPTER 2

FCT000 and FCT010 Table Layout

```
 1  @TNR FCT010 A F 43
>* SYSTEM-FUNCTION FUNCTIONS DEFINITION
>* only MTL comment & valid entries
>* no COBOL nor MTL in this table !
>*
>* Function Name-----> FCSx R M Parameters
RANDOM-INIT            001 N 0 N
ADD-SUB-FIXED          002 D 1 DAN
ADD-SUB-RANDOM-DAYS    003 D 1 DANN
HASH                   004 A 3 AAN
MASK                   005 A 4 AANN
RANDOM-DATE            006 D 0 DD
RANDOM-NBR             007 N 0 NN
RANDOM-OPERATOR        008 A 0 A
RANDOM-TEXT            009 A 2 NN
SEQ-CHAR               010 A 1 A
TRANSPOSE              011 A 2 AA
VARIANCE-1             012 N 3 NNN
VARIANCE-2             013 N 3 NNN
>*-------------------------------------
AMEX-CARD              050 N 0 N
MASTER-CARD            051 N 0 N
VISA-CARD              052 N 0 N
BE-BBAN                053 N 0 N
BE-IBAN                054 A 0 N
BE-RRN-NRN             055 N 0 NN
BE-ZIP                 056 N 0
BE-ZIP-DISTR           057 N 1 N
FIRST-NAME             058 A 0
FR-BBAN                059 A 0 NA
FR-IBAN                060 A 0 NA
```

Note: The table RANDOM-NBR will be used throughout all the examples.

2.1. Table Parameters

Col. 1-22	External name of the function (max. 22 characters).
Col. 24-26	Number of the function (3 numeric positions). The table containing the processing logic will be OBPnnn where nnn is the function number.

Col. 30	Data type of the by the function returned value (A=Alphanumeric, N=Numeric, D=Date)
Col. 32	Number of the mandatory parameters (0-9)
Col. 34-43	Data type of each parameter (max. 10) (A=Alphanumeric, N=Numeric, D=Date)

The line numbers are not part of the table; the external function name starts at position one.

The entries do not need to be sorted.

The function numbers do not need to follow each other.

All parameters are mandatory, excepted the parameter data types when the function doesn't support parameters.

Line one should never be modified (this is true for each Dictionary table).

This table should never contain any COBOL syntax nor MTL syntax. Lines that are not describing a User-defined Function should begin at position one with the MTL comment characters ('>*').

There is no restriction on the use of comments, as long as they appear on a separate line.

Function numbers are always three digits long, leading zeros are mandatory.

The Working-Storage column should only contain 'Y' or 'N'.

The Returned Data Type column should only contain 'A', 'N' or 'D'.

The Number of Mandatory Parameters column should only contain a value between 0 and 9.

The Parameters Data Type columns should only contain 'A', 'N', 'D' or space if the corresponding parameter is not needed.

The table has to be generated by the MetaSuite Generator each time a modification is made.

The user is responsible for backups and versioning of his MetaSuite Dictionary. When IKAN delivers a new version, only the basic set of generation tables are provided. All the by the user created and modified tables have to be added to the new Dictionary.

Note: Erroneous modification of the basic generation table set will result in the corruption of the COBOL Generator.

The FCT000 table, which is used for user functions, has the same syntactical format

CHAPTER 3
FCU Tables

Those tables (max. 999 of each) will contain valid COBOL and MTL syntax. The first position of a line has a special meaning and is mandatory:

*	COBOL comment
A	COBOL line starts at position 8
B	COBOL line starts at position 12
C	COBOL line starts at position 16
>	this line contains valid MTL syntax

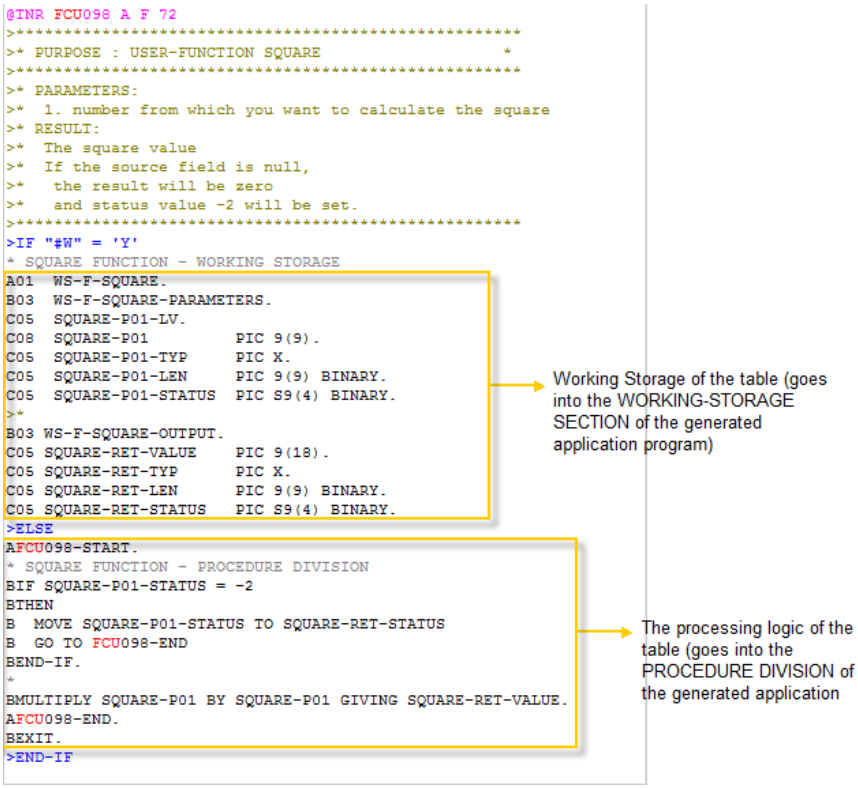
Blank lines are not allowed. If you need one, start the blank line with '>', '*', '>*','A', 'B', or 'C'.

A best practice is to modify the name of the table on the first line of an existing table and to save it. You can modify the newly created table.

3.1. Structure of the FCU Tables

In order to facilitate the maintenance of a user function, both working storage as procedure logic have been put into one table. This is not a common practice, but in this particular case very efficient.

The following graphic shows a typical Dictionary table declaring a User-defined Function.



IKAN advises the users to add a comment part just after the first line. This block will document useful information for the designers, especially the function description and the parameters processed by the function, both mandatory and optional.

The ">IF "#W" = 'Y' ... >ELSE ... >END-IF" MTL statement is needed to control the usage of the table at different periods of time during the COBOL code generation phase of the MetaSuite generator. This is mandatory because an FCU table is always used twice by the generator: at WORKING-STORAGE SECTION generation time and at PROCEDURE DIVISION generation time.

3.2. The Working-Storage of the Table

The data declaration part of the table will start with a 01 COBOL level (mandatory), for example “A01 WS-F-< name of the function as in the FCT001 table>.” in order to avoid conflicts with the data fields declared by the generator and will be divided into a number of parts, typically three:

1. the declaration of the parameters (optional)
2. the declaration of the work fields (optional)
3. the declaration of the output returned by the function (mandatory)

The user is free to add more parts. It is advisable to create a group level for each part.

If a table does not need any working storage at all, the ">IF "#W" = 'Y' ... >ELSE ... >END-IF" MTL statement still needs to be used as in:

```
>IF "#W" = 'Y'
* no W-S
>ELSE
BMOVE ...
   :
>END-IF
```

And the opposite is true:

```
>IF "#W" = 'Y'
A01 ...
>ELSE
* no logic
>END-IF
```

3.3. Declaration of the Function Parameters

As mentioned before, information about the parameters is stored in the FCT001 table (in case of USER functions) or FCT011 table (in case of SYSTEM functions).

The "M" column contains the number of mandatory parameters. This is a value between 0 and 9.

The ten "Parameters" columns contain the data types of each parameter, in correct sequence. The data types are coded as "A" for alphanumeric parameters, as "N" if they are numeric, or as "D" if it concerns a date parameter. Parameters need to be added in sequence (no gaps). The number of mandatory parameters should be less or equal to the total number of parameters. All the mandatory parameters come first, optional ones come last.

When the FCT001 table entry has been saved and generated, the FCU table should be created. The general rules are explained in the section FCT000 and FCT010 Table Layout. The sample picture used in that chapter shows how the parameter data are to be defined. The used method is very self-explanatory, however ... it is not the best, nor the easiest, nor the most maintainable way to do it. It is advised to create the parameter property blocks by means of the FCC000 macro.

This is done as follows:

For each parameter that the function uses, the PICTURE property of the input parameter field should be specified, by means of a '>SET' command:

The command for parameter 1 is:

```
>SET @P01 = '<picture-clause>'
```

The command for parameter 2 is:

```
>SET @P02 = '<picture-clause>'
```

Etc...

The PICTURE property of the output parameter field should also be specified by means of a '>SET' command:

```
>SET @RET = '<picture-clause>'
```

These parameters will be picked up and translated by the FCC000 table, which should be called.

```
>CALL FCC000
```

After this call, the user is free to add work fields. Please be careful: use unique names.

Sample

```
>IF "#W" = 'Y'
* RIGHT FUNCTION - WORKING STORAGE
>SET @P01 = 'X(256)'
>SET @P02 = '9(6)'
>SET @RET = 'X(256)'
>CALL FCC000
*
A01  WS-F-RIGHT-WORK.
B10  WS-F-RIGHT-LEN         PIC 9999 BINARY.
B10  WS-F-RIGHT-FROM        PIC S9999 BINARY.
*
>ELSE
```

The generated result will be as follows:

```
      * RIGHT FUNCTION - WORKING STORAGE                                  FCU09
      * FUNCTION RIGHT: PARAMETER BLOCK                                   FCC000
       01  RIGHT-INP-BLOCK.                                              FCC000
           10  RIGHT-P01-LV.                                             FCC000
               20  RIGHT-P01-VALUE     PIC X(256).                       FCC000
           10  RIGHT-P01-TYP       PIC X.                                FCC000
           10  RIGHT-P01-LEN       PIC 9(9) BINARY.                      FCC000
           10  RIGHT-P01-FS        PIC S9(4) BINARY.                     FCC000
           10  RIGHT-P01-FMT       PIC 99.                               FCC000
      *                                                                   FCC000
           10  RIGHT-P02-LV.                                             FCC000
               20  RIGHT-P02-VALUE     PIC 9(6).                         FCC000
           10  RIGHT-P02-TYP       PIC X.                                FCC000
           10  RIGHT-P02-LEN       PIC 9(9) BINARY.                      FCC000
           10  RIGHT-P02-FS        PIC S9(4) BINARY.                     FCC000
           10  RIGHT-P02-FMT       PIC 99.                               FCC000
      *                                                                   FCC000
       01  RIGHT-OUT-BLOCK.                                              FCC000
           10  RIGHT-RET.                                                FCC000
               20  RIGHT-RET-VALUE     PIC X(256).                       FCC000
           10  RIGHT-RET-TYP       PIC X.                                FCC000
           10  RIGHT-RET-LEN       PIC 9(9) BINARY.                      FCC000
           10  RIGHT-RET-FS        PIC S9(4) BINARY.                     FCC000
           10  RIGHT-RET-FMT       PIC 99.                               FCC000
      *                                                                   FCU099
       01  WS-F-RIGHT-WORK.                                              FCU099
           10  WS-F-RIGHT-LEN         PIC 9999 BINARY.                   FCU099
           10  WS-F-RIGHT-FROM        PIC S9999 BINARY.                  FCU099
```

Description of the generated parameters

Parameter	Description
Pxx-VALUE	The input parameter value
Pxx-TYP	The input parameter type, which can be A,N or D (Alpha-numeric, Numeric or Date)
Pxx-LEN	The size in bytes of the input parameter
Pxx-FS	The field status of the input parameter. Zero means ok, minus two means null, and this can also mean that the parameter is optional and not specified.
Pxx-FMT	The date format of the input parameter, in case of a date.
RET-TYP	The output parameter type, which can be A,N or D (Alpha-numeric, Numeric or Date)
RET-LEN	The size in bytes of the output parameter
RET-FMT	The date format of the output parameter, in case of a date.

Depending on the values of the input parameters, some other fields should be filled in order to make the function work.

Parameter	Description
RET-VALUE	The output parameter value
RET-FS	The field status of the output parameter. Zero means ok, minus two means null. Other values can be entered as well.

Special remark for functions that return a date:

- Please set the picture clause of @RET to 9(8).
- Return the date in the date format YYYYMMDD
- The generator will take care of the target date format of the receiving field and will convert the computed date to the proper date format.

3.4. Examples

Example 1

```
… = SYSTEM-FUNCTION 'MY-OWN-FUNCTION'  (MY-OWN-DATA-FIELD)
```

MY-OWN-DATA-FIELD is of character data type, and has a length of 30 bytes.

```
MY-OWN-FUNCTION-P01-LV     will not contain LOW-VALUES
MY-OWN-FUNCTION-P01        will contain the value of data field MY-OWN-DATA-FIELD
MY-OWN-FUNCTION-P01-LEN    will contain 30
MY-OWN-FUNCTION-P01-TYP    will contain "A"
MY-OWN-FUNCTION-P01-FMT    will not be filled in by the generator
MY-OWN-FUNCTION-P01-STATUS will contain the status of field MY-OWN-DATA-FIELD
```

Example 2

```
… = SYSTEM-FUNCTION 'MY-OWN-FUNCTION'
```

(Parameter number one is optional)

```
MY-OWN-FUNCTION-P01-LV     will be initialized
MY-OWN-FUNCTION-P01        will contain LOW-VALUES
MY-OWN-FUNCTION-P01-LEN    will contain 8
MY-OWN-FUNCTION-P01-TYP    will contain "D"
MY-OWN-FUNCTION-P01-FMT    will contain 29
MY-OWN-FUNCTION-P01-STATUS will contain the value -2, meaning that the parameter
                           is NULL
```

Example 3

```
… = SYSTEM-FUNCTION 'MY-OWN-FUNCTION'  (MY-OWN-DATE-FIELD)
```

MY-OWN-DATE-FIELD is of character data type, has a length of 8 bytes and is a date of 'YYYYMMDD' format.

```
MY-OWN-FUNCTION-P01-LV     will not contain LOW-VALUES
MY-OWN-FUNCTION-P01        will contain the value of data field MY-OWN-DATE-FIELD
MY-OWN-FUNCTION-P01-LEN    will contain 8
MY-OWN-FUNCTION-P01-TYP    will contain "D"
```

```
MY-OWN-FUNCTION-P01-FMT    will contain 29
MY-OWN-FUNCTION-P01-STATUS will contain the status of field MY-OWN-DATE-FIELD
```

3.5. Declaration of the Workfields

No specific rules. Needs to be valid COBOL syntax.

3.6. Declaration of the Output Returned by the Function

Every function returns a value, whatever it is (return code, alphanumeric/numeric/date value, …).

The structure and the usage by the generator of the return value definition are similar to those of a parameter.

The by the function returned value is declared using at least three or four COBOL lines, depending on the data type of the value. The user is free to add more COBOL lines (REDEFINES, sub-divisions, …).

Line 1	<COBOL level> <name of the function as in the FCT000 table>-VALUE <COBOL picture clause>, ending with a full stop. For example: A 05 RANDOM-TEXT-VALUE PIC X(<your own value>).
Line 2	<COBOL level> <name of the function as in the FCT000 table>-RET-LEN PIC 9(4) BINARY, ending with a full stop. For example: A 05 RANDOM-TEXT-RET-LEN PIC 9(9) BINARY.
Line 3	<COBOL level> <name of the function as in the FCT000 table>-RET-TYP PIC X, ending with a full stop. For example: A 05 RANDOM-TEXT-RET-TYP PIC X.
Line 4	<COBOL level> <name of the function as in the FCT000 table>-RET-STATUS PIC S9(4)BINARY. For example: A 05 RANDOM-TEXT-RET-STATUS PIC S9(4) BINARY.
Line 5	This line is mandatory if the returned value has a date data type <COBOL level> <name of the function as in the FCT000 table>-RET-FMT PIC 99, ending with a full stop. For example: A 05 ADD-SUB-RANDOM-DAYS-RET-FMT PIC 99.

All those data declarations will be added to the WORKING-STORAGE SECTION of the generated application program.

The SYSTEM-FUNCTIONS do not use this output field declaration. System functions use pre-defined declarations that start with the FCT-prefix.

3.7. The Processing Logic of the Table

Probably, the first thing to do will be to test the parameters in order to decide action has to be performed if they are not provided. This can be done by testing the STATUS field. If this status field is -2 then the parameter value is NULL or not provided..

Example:

```
BIF ADD-SUB-RANDOM-DAYS-P03-STATUS = -2
*  assign a default value to parameter 3 of the ADD-SUB-RANDOM-DAYS function
     :
```

Once this is done, the processing logic of the function can be written. The …-LEN, …-TYP, and … FMT data field values provided by the generator can be used to provide great flexibility to the function.

Returning a Result

Once the function is ready with the work to be done, the end result has to be returned. Use the data field(s) declared in the output returned block of the table. The …-LEN, …-TYP, and …-FMT data fields contain the correct values as provided by the generator and they can be tested to control the formatting of the result value.

3.8. FCC Tables

If some working-storage or logic is repeatedly made, then it can be useful to put this piece of logic in a seperate table.

We advise the user to use table names in the range of FCC100 to FCC999. This easens the process of finding them back amongst the numerous tables. IKAN guarantees not to touch this range of tables.

APPENDIX A
Functions Overview

A.1. RANDOM-INIT

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
RANDOM-INIT	001	0	Numeric	The returned value will be the result of the COBOL 'Random' function on this random seed.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	System Time (in hours, minutes, seconds, hundreds of a second)	N	Sets the random seed to this value. If omitted, a variable depending on the system-time will be used.

Note: Be careful when using RANDOM-INIT in combination with SYS-RANDOM, as this could create an overlap and result in an incorrect, misleading SYS-RANDOM setting.

A.2. ADD-SUB-FIXED

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
ADD-SUB-FIXED	002	1	Date	Adds or subtracts a given number of days to or from a date.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		D	The date to modify.
2	Optional	+	A	'+' to add or '-' to subtract.
3	Optional	36525	N	The number of days to be added or subtracted.

A.3. ADD-SUB-RANDOM-DAYS

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
ADD-SUB-RANDOM-DAYS	003	1	Date	Adds or subtracts a randomly generated number of days to or from a date.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		D	The date to modify.
2	Optional	+	N	'+' to add or '-' to subtract.
3	Optional	1	N	The minimum number of days.
4	Optional	36525	N	The maximum number of days.

A.4. HASH

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
HASH	004	3	Alphanumeric	Returns the hashed value of a text based on the requested algorithm.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		A	The hashing algorithm: 'SHA1', 'SHA224' or 'SHA256'.
2	Mandatory		A	The text to be hashed.
3	Mandatory		N	The number of bytes to be used by the hashing algorithm. This value has to be a number between 1 and the text size

In cryptography, SHA-1 and SHA-2 are cryptographic hash functions designed by the United States National Security Agency and published by the United States NIST as a U.S. Federal Information Processing Standard. SHA stands for "secure hash algorithm". It calculates – with a large amount of certainty – a unique key out of a string.

SHA-1 is the most widely used of the existing SHA hash functions, and is employed in several widely used security applications and protocols. SHA-2 is more severe and has some dialects. SHA-224 and SHA-256 are SHA-2 dialects.

More information about the hashing algorithms SHA-1 and SHA-2 (224/256) can be found on Wikipedia.

A.5. MASK

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
MASK	005	4	Alphanumeric	Masks a requested portion of a text using the provided masking character.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		A	The text to be masked.
2	Optional	STD	A	The following options are available: • STD: all numbers and alphabetic characters will be replaced. • ALL: all characters will be replaced. • NUM: all numbers will be replaced. • ALPHA: all alphabetic characters will be replaced.
3	Mandatory	*	A	The masking character.
3	Mandatory	1	N	The starting position of the mask.
4	Mandatory	The remaining size of the string, counting from the starting position.	N	The length of the mask.

A.6. NUMERIC-TRANSPOSE

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
NUMERIC-TRANSPOSE	084	3	Numeric	This function changes a numeric key into another numeric key based on an algorithm and a random key.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		N	The key value to be transposed.
2	Mandatory		N	The minimum value.
3	Mandatory		N	The maximum value.
4	Optional		N	The key to use to transpose. This key must be a prime number higher than all mandatory parameters.

Each number will be replaced by another using a random seed.

A.7. RANDOM-DATE

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
RANDOM-DATE	006	0	Date	Randomly generates a date within a range of two dates.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	The day that was exactly a hundred years in the past.	D	The first date of the date range. This date can be older or newer than the second date.
2	Optional	Today's date	D	The second date of the date range. This date can be older or newer than the first date.

A.8. RANDOM-NBR

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
RANDOM-NBR	007	0	Numeric	Randomly generates an integer

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	zero	N	The minimum value.
2	Optional	999.999.999.999.999 .999	N	The maximum value.

A.9. RANDOM-OPERATOR

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
RANDOM-OPERATOR	008	0	Alphanumeric	Randomly generates an arithmetic operator

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	plus & minus	A	A sequence of maximum five arithmetical operators ('+' '-' '*' '/' '%') as in "+-*/%" or "+%*+".

A.10. RANDOM-TEXT

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
RANDOM-TEXT	009	2	Alphanumeric	Randomly generates text

Sample of a random text: `Jiff urn appyaee fl yyip nrna siz yu. Hcyxqb rzjfir.`

Note: The text is written in lowercase, except for the first character of a sentence.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory	zero	N	The minimum acceptable text length.
2	Mandatory	80 characters	N	The maximum acceptable text length.

A.11. SEQ-CHAR

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
SEQ-CHAR	010	1	Alphanumeric	Adds 1 to a base 62 character string

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		A	The base 62 text.

The letter “a” will be changed into ‘b’, and so on. This is a very easy way of encrypting a text.

A.12. TRANSPOSE

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
TRANSPOSE	011	2	Alphanumeric	Converts a text using a key

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		A	The text to be transposed.
2	Mandatory		A	The key to use to transpose.

Each character will be replaced by another using a random seed.

A.13. VARIANCE-1

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
VARIANCE-1	012	3	Numeric	Randomly adds or subtracts a provided amount

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		N	The base amount to be modified.
2	Mandatory		N	The minimum amount to add or subtract.
3	Mandatory		N	The maximum amount to add or subtract.

A.14. VARIANCE-2

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
VARIANCE-2	013	3	Numeric	Randomly adds or subtracts a provided percentage

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		N	The base amount to modify.
2	Mandatory		N	The minimum percentage to add or subtract.
3	Mandatory		N	The maximum percentage to add or subtract.

A.15. AMEX-CARD

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
AMEX-CARD	050	0	Numeric	Randomly generates an American Express card number.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	Provider number is randomly generated	N	The first six digits (the card provider number).

A.16. MASTER-CARD

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
MASTER-CARD	051	0	Numeric	Randomly generates a Master card number.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	Provider number is randomly generated	N	The first six digits (the card provider number).

A.17. VISA-CARD

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
VISA-CARD	052	0	Numeric	Randomly generates a Visa card number.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	Provider number is randomly generated	N	The first six digits (the card provider number).

A.18. BE-BBAN

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
BE-BBAN	053	0	Numeric	Randomly generates a Belgian BBAN

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	zero - 989	N	The first three digits (the bank code).

A.19. BE-IBAN

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
BE-IBAN	054	0	Alphanumeric	Randomly generates a Belgian IBAN

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	zero - 989	N	The first three digits (the bank code).

A.20. BE-RRN-NRN

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
BE-RRN-NRN	055	0	Numeric	

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	Gender code is randomly generated	N	1 for a female person, 2 for a male person.
2	Optional	today - 105 years	N	The minimum birth year (YYYY).

A.21. BE-ZIP

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
BE-ZIP	056	0	Numeric	Randomly returns a zip code from a table containing a list of Belgian zip codes

Parameter

This function has no parameters.

A.22. BE-ZIP-DISTR

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
BE-ZIP-DISTR	057	1	Numeric	Randomly returns a valid Belgian zip code based on the first parameter (which has to be valid)

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		N	The first digit or the first two digits of the wanted Belgian zip code.

A.23. FIRST-NAME

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
FIRST-NAME	058	0	Alphanumeric	Randomly returns a first name from a table containing a list of first names

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	All	A	M (male) or F (female). If not specified, first names of both genders will be randomly selected.

A.24. FR-BBAN

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
FR-BBAN	059	0	Alphanumeric	Randomly generates a French BBAN

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	1 - the number of entries in the MOT001 table	N	The five digits bank code as provided by the Banque de France.
2	Optional	no initialization	A	Y(es) or N(o) to initialize or not the random generation with the system time.

A.25. FR-IBAN

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
FR-IBAN	060	0	Alphanumeric	Randomly generates a French IBAN.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Optional	1 - the number of entries in the MOT001 table	N	The five digits bank code as provided by the Banque de France.
2	Optional	no initialization	A	Y(es) or N(o) to initialize or not the random generation with the system time.

A.26. SUR-NAME

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
SUR-NAME	061	0	Alphanumeric	This function randomly returns a surname from a table containing a list of surnames.

Parameters

This function has no parameters.

A.27. PICK-STRING

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
PICK-STRING	062	1	Alphanumeric	Picks a string out of a series of 1 to 9 possible strings.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		A	The first string of the series.
2-9	Optional		A	8 additional strings may be added.

Example

```
animal = SYSTEM-FUNCTION PICK-STRING ( 'monkey', 'elephant', 'snake', 'bear', 'li-
on', 'pig', 'fly', 'flamingo' )
```

A.28. PICK-NUMBER

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
PICK-NUMBER	063	1	Numeric	Picks a number out of a series of 1 to 9 possible numbers.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		N	The first number of the series.
2-9	Optional		N	8 additional numbers may be added.

Example

```
age = SYSTEM-FUNCTION PICK-NUMBER ( 0 , 1 , 15 , 18 , 32 , 53, 60 , 75 , 100 )
```

A.29. PICK-DATE

Function

Function Name	Function Number	Number of Mandatory Parameters	Returned Data Type	Description
PICK-NUMBER	064	1	Date	Picks a date out of a series of 1 to 9 possible dates. These dates may be defined as litterals and/or as variables.

Parameters

Parameter Number	Mandatory/ Optional	Default Value	Data Type	Description
1	Mandatory		D	The first date of the series.
2-9	Optional		D	8 additional dates may be added.

Example

```
date = SYSTEM-FUNCTION PICK-DATE ( 20121221, 20121231, date-field-3, date-field-4 )
```