ETSI TS 101 875 V1.1.1 (2000-12)

Technical Specification

**Methods for Testing and Specification (MTS);
The Tree and Tabular Combined Notation version 3
TTCN-3: Library of Additional Predefined Functions**

Reference

DTS/MTS-00070

Keywords

MTS, testing, TTCN

Contents

Intellectual Property Rights

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

1 Scope

The present document defines additional pre-defined functions that may be used in the TTCN-3 test language defined in ES 201 873-1.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] ETSI ES 201 873-2: "Methods for Testing and Specification (MTS), The Tree and Tabular Combined Notation version 3, TTCN-3: Tabular Presentation Format".

3 Abbreviations

For the purposes of the present document, the following abbreviations applies:

TTCN	Tree and Tabular Combined Notation

4 Introduction

TTCN-3 contains a number of pre-defined functions as defined in annex D of [1]. The present document is a library of additional pre-defined functions that should be supported by all TTCN-3 tools that are used in the development of ETSI Conformance Abstract Test Suites.

5 Limitations

Names of functions added to this library shall be unique within the library. Names of functions added to this library shall not be one of the predefined function names defined in [1] annex D.

The general format of the function definitions shall be:

```
functionname ( formalpartype1 value1, …, formalpartypen valuen ) returnkeyword returntype
```

For example:

bit2int(**bitstring** value) **return integer**

6 Definition of additional pre-defined TTCN-3 functions

List of functions in this library:

a) bitstring to charstring;

b) hexstring to charstring;

c) octetstring to charstring;

d) bitstring to hexstring;

e) hexstring to octetstring;

f) bitstring to octetstring;

g) hexstring to bitstring;

h) octetstring to hexstring;

i) octetstring to bitstring;

j) integer to float;

k) float to integer;

l) generate random number.

6.1 bitstring to charstring

```
bit2string (bitstring value) return charstring
```

This function converts a single **`bitstring`** value to a single **`charstring`**. The resulting **`charstring`** has the same length as the **`bitstring`** and contains only the characters '0' and '1'.

For the purpose of this conversion, a **`bitstring`** should be converted into a **`charstring`**. Each bit of the **`bitstring`** is converted into a character '0' or '1' depending on the value 0 or 1 of the bit. The consecutive order of characters in the resulting **`charstring`** is the same as the order of bits in the **`bitstring`**. For example:

```
bit2string ('1110101'B) will return "1110101"
```

6.2 hexstring to charstring

```
hex2string (hexstring value) return charstring
```

This function converts a single hexstring value to a single charstring. The resulting charstring has the same length as the hexstring and contains only the characters '0' to '9'and 'A' to 'F'.

For the purpose of this conversion, a hexstring should be converted into a charstring. Each hex digit of the hexstring is converted into a character '0' to '9' and 'A' to 'F' depending on the value 0 to 9 or A to F of the hex digit. The consecutive order of characters in the resulting charstring is the same as the order of digits in the hexstring. For example:

```
hex2string ('AB801'H) will return "AB801"
```

6.3 octetstring to charstring

```
oct2string (octetstring value) return charstring
```

This function converts a single octetstring value to a single charstring. The resulting charstring has the double length of the octetstring and contains only the characters '0' to '9'and 'A' to 'F'.

For the purpose of this conversion, a octetstring should be converted into a charstring. Each hex digit of the octetstring is converted into a character '0' to '9' and 'A' to 'F' depending on the value 0 to 9 or A to F of the hex digit. The consecutive order of characters in the resulting charstring is the same as the order of digits in the octetstring. For example:

```
oct2string ('AB8015'O)= "AB8015"
```

6.4 bitstring to hexstring

```
bit2hex (bitstring value) return hexstring
```

This function converts a single bitstring value to a single hexstring. The resulting hexstring represents the same value as the bitstring.

For the purpose of this conversion, a bitstring should be converted into a hexstring, where the bitstring is divided into groups of four bits beginning with the rightmost bit. Each group of four bits is converted into a hex digit as follows: '0000'B -> '0'H, '0001'B -> '1'H, '0010'B -> '2'H, '0011'B -> '3'H, '0100'B -> '4'H, '0101' -> '5'H, '0110' -> '6'H, '0111'B -> '7'H, '1000'B -> '8'H, '1001'B -> '9'H, '1010'B -> 'A'H, '1011'B -> 'B'H, '1100'B -> 'C'H, '1101'B -> 'D'H, '1110'B -> 'E'H, and '1111'B -> 'F'H. When the leftmost group of bits does contain less than 4 bits, this group is filled with '0'B from the left until it contains exactly 4 bits and is converted afterwards. The consecutive order of hex digits in the resulting hexstring is the same as the order of groups of 4 bits in the bitstring. For example:

```
bit2hex ('111010111'B)= '1D7'H
```

6.5 hexstring to octetstring

```
hex2oct (hexstring value) return octetstring
```

This function converts a single hexstring value to a single octetstring. The resulting octetstring represents the same value as the hexstring.

For the purpose of this conversion, a hexstring should be converted into a octetstring, where the octetstring contains the same sequence of hex digits as the hexstring when the length of the hexstring modulo 2 is 0. Otherwise, the resulting octetstring contains 0 as leftmost hex digit followed by the same sequence of hex digits as in the hexstring. For example:

```
hex2oct ('1D7'H)= '01D7'O
```

6.6 bitstring to octetstring

```
bit2oct (bitstring value) return octetstring
```

This function converts a single bitstring value to a single octetstring. The resulting octetstring represents the same value as the bitstring.

For the conversion the following holds: bit2oct(value)=hex2oct(bit2hex(value)). For example:

```
bit2oct ('111010111'B)= '01D7'O
```

6.7 hexstring to bitstring

```
hex2bit (hexstring value) return bitstring
```

This function converts a single hexstring value to a single bitstring. The resulting bitstring represents the same value as the hexstring.

For the purpose of this conversion, a hexstring should be converted into a bitstring, where the hex digits of the hexstring are converted in groups of bits as follows. '0'H -> '0000'B, '1'H -> '0001'B, '2'H -> '0010'B, '3'H -> '0011'B, '4'H -> '0100'B, '5'H -> '0101', '6'H -> '0110', '7'H -> '0111'B, '8'H-> '1000'B, '9'H -> '1001'B, 'A'H -> '1010'B, 'B'H -> '1011'B, 'C'H -> '1100'B, 'D'H -> '1101'B, 'E'H -> '1110'B, and 'F'H -> '1111'B. The consecutive order of the groups of 4 bits in the resulting bitstring is the same as the order of hex digits in the hexstring. For example:

```
hex2bit ('1D7'H)= '000111010111'B
```

6.8 octetstring to hexstring

```
oct2hex (octetstring value) return hexstring
```

This function converts a single octetstring value to a single hexstring. The resulting hexstring represents the same value as the octetstring.

For the purpose of this conversion, a octetstring should be converted into a hexstring containing the same sequence of hex digits as the octetstring. For example:

```
oct2hex ('1D74'O)= '1D74'H
```

6.9 octetstring to bitstring

```
oct2bit (octetstring value) return bitstring
```

This function converts a single octetstring value to a single bitstring. The resulting bitstring represents the same value as the octetstring.

For the conversion the following holds: oct2bit(value)=hex2bit(oct2hex(value)). For example:

```
oct2bit ('01D7'O)='000111010111'B
```

6.10 integer to float

```
int2float (integer value) return float
```

This function converts an integer value into a float value. For example:

```
int2float(4) = 4.0
```

6.11 float to integer

```
float2int (float value) return integer
```

This function converts a float value into an integer value by removing the fractional part of the argument and returning the resulting integer. For example:

```
float2int(3.12345E2) = float2int(312.345) = 312
```

6.12 rnd

```
rnd ([float seed]) return float
```

The **rnd** function returns a (pseudo) random number less than 1 but greater or equal to 0. The random number generator is initialized by means of an optional seed value. Afterwards, if no new seed is provided, the last generated number will be used as seed for the next random number. Without a previous initialization a value calculated from the system time will be used as seed value when **rnd** is used the first time.

NOTE: Each time the **rnd** function is initialized with the same seed value, it shall repeat the same sequence of random numbers.

To produce a random integers in a given range, the following formula can be used:

```
float2int(int2float(upperbound – lowerbound + 1) * rnd()) + lowerbound
// Here, upperbound and lowerbound denote highest and lowest number in range.
```

Annex A (informative): Bibliography

- ETSI ES 201 873-1: "Methods for Testing and Specification (MTS), The Tree and Tabular Combined Notation version 3, TTCN-3: Core language".

History

Document history		
V1.1.1	December 2000	Publication