ETSI ES 203 022 V1.1.1 (2017-07)

Methods for Testing and Specification (MTS);
The Testing and Test Control Notation version 3;
TTCN-3 Language Extensions: Advanced Matching

Reference
DES/MTS-203022AdvMatch ed111

Keywords
conformance, testing, TTCN-3

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

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

The present document relates to the multi-part standard ETSI ES 201 873 covering the Testing and Test Control Notation version 3, as identified below:

Part 1: "TTCN-3 Core Language";

Part 4: "TTCN-3 Operational Semantics";

Part 5: "TTCN-3 Runtime Interface (TRI)";

Part 6: "TTCN-3 Control Interface (TCI)";

Part 7: "Using ASN.1 with TTCN-3";

Part 8: "The IDL to TTCN-3 Mapping";

Part 9: "Using XML schema with TTCN-3";

Part 10: "TTCN-3 Documentation Comment Specification";

Part 11: "Using JSON with TTCN-3".

NOTE: Part 2 is in status "historical" and part 3 is no longer maintained.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document defines the support of advance matching of TTCN-3. TTCN-3 can be used for the specification of all types of reactive system tests over a variety of communication ports. Typical areas of application are protocol testing (including mobile and Internet protocols), service testing (including supplementary services), module testing, testing of OMG CORBA based platforms, APIs, etc. TTCN-3 is not restricted to conformance testing and can be used for many other kinds of testing including interoperability, robustness, regression, system and integration testing. The specification of test suites for physical layer protocols is outside the scope of the present document.

TTCN-3 packages are intended to define additional TTCN-3 concepts, which are not mandatory as concepts in the TTCN-3 core language, but which are optional as part of a package which is suited for dedicated applications and/or usages of TTCN-3.

While the design of TTCN-3 package has taken into account the consistency of a combined usage of the core language with a number of packages, the concrete usages of and guidelines for this package in combination with other packages is outside the scope of the present document.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references,only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference/.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI ES 201 873-1: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language".

[2] ETSI ES 201 873-4: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 4: TTCN-3 Operational Semantics".

[3] ETSI ES 201 873-5: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 5: TTCN-3 Runtime Interface (TRI)".

[4] ETSI ES 201 873-6: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 6: TTCN-3 Control Interface (TCI)".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references,only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI ES 201 873-7: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 7: Using ASN.1 with TTCN-3".

[i.2] ETSI ES 201 873-8: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 8: The IDL to TTCN-3 Mapping".

[i.3] ETSI ES 201 873-9: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 9: Using XML schema with TTCN-3".

[i.4] ETSI ES 201 873-10: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 10: TTCN-3 Documentation Comment Specification".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ETSI ES 201 873-1 [1], ETSI ES 201 873-4 [2], ETSI ES 201 873-5 [3] and ETSI ES 201 873-6 [4] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI ES 201 873-1 [1], ETSI ES 201 873-4 [2], ETSI ES 201 873-5 [3] and ETSI ES 201 873-6 [4] apply.

4 Package conformance and compatibility

The package presented in the present document is identified by the package tag:

`"TTCN-3:2017 Advanced Matching"` - to be used with modules complying with the present document.

For an implementation claiming to conform to this package version, all features specified in the present document shall be implemented consistently with the requirements given in the present document and in ETSI ES 201 873-1 [1] and ETSI ES 201 873-4 [2].

The package presented in the present document is compatible to:

- ETSI ES 201 873-1 [1], version 4.9.1;
- ETSI ES 201 873-4 [2], version 4.6.1;
- ETSI ES 201 873-5 [3], version 4.8.1;
- ETSI ES 201 873-6 [4], version 4.9.1;
- ETSI ES 201 873-7 [i.1];
- ETSI ES 201 873-8 [i.2];
- ETSI ES 201 873-9 [i.3];
- ETSI ES 201 873-10 [i.4].

If later versions of those parts are available and should be used instead, the compatibility to the package presented in the present document has to be checked individually.

5 Package Concepts for the Core Language

5.0 General

This package defines advanced matching mechanisms for TTCN-3, i.e. new matching mechanisms which go beyond the matching mechanisms defined in ETSI ES 201 873-1 [1]. This package realizes the following concepts:

- *Dynamic matching* allows to specify matching in a function-like fashion, i.e. statement blocks and functions can be used to define matching.
- *Templates with variable bindings* ease the retrieval field values of received messages and signatures. For this the "->" symbol is used to denote a value assignment to a variable or an **out** parameter in the scope of a template definition in case of a successful template matching.
- The *additional logical operators* conjunction, implication, exclusion and disjunction allow the combination matching mechanisms for advanced matching.
- *Repetition* allows to match repetitions of a sub-sequence templates inside values of a certain type.
- Restrictions for the omit symbol and templates with restriction omit are relieved *allowing omit symbols and templates with restriction omit as operands for the equality operator*.

5.1 Dynamic Matching

A dynamic matching is a special matching mechanism. Similar to other matching mechanisms, it can be considered as a Boolean function that indicates successful matching for the value to be matched by returning the value **`true`** and unsuccessful matching by returning the value **`false`**.

Syntactical Structure

@dynamic (*StatementBlock* | *FunctionRef*)

Semantic Description

The *StatementBlock* shall return a value of type **`boolean`**. The value to be matched is referenced by the special keyword **`value`**. When applying this matching mechanism to a value, the *StatementBlock* is executed and if and only if the execution returns **`true`**, the dynamic matching function matches. Unsuccessful matching shall return **`false`**.

A dynamic matching function can only be used in the context of a template, the **`value`** expression inside the *StatementBlock* shall have the same type as the whole template.

The notation **`@dynamic`** *FunctionRef* denotes a shorthand for the special case **`@dynamic { return`** *FunctionRef*(**`value) }`** where *FunctionRef* is a reference to a Boolean function with a single parameter compatible with the template's type. Here, the type of the parameter of the referenced function determines the type context, if this template's place of usage does not provide a type context.

Restrictions

a) The dynamic matching syntax shall only be used in a typed context.

b) The *StatementBlock* shall compute a value of type **`boolean`**.

c) The *StatementBlock* shall be deterministic and side-effect free and follow the restrictions of clause 16.1.4 of ETSI ES 201 873-1 [1].

d) The *StatementBlock* shall not use variables that are declared outside of the *StatementBlock*.

e) The *StatementBlock* shall not use **`inout`** or **`out`** parameters.

f) Only if the dynamic matching syntax appears on the right-hand-side of a parameterized template definition, the formal **`in`** parameters of that definition may be referenced inside the *StatementBlock*. All other formal **`in`** parameters shall not be used by the *StatementBlock*.

EXAMPLE:

```
type record of integer Numbers;
template Numbers mw_sorted := @dynamic { // value is of type Numbers
   for (var integer v_i := 1; v_i < lengthof(value); v_i := v_i + 1)  {
      if (value[v_i-1] > value[v_i]) { return false }
   }
   return true;
}  // mw_sorted(v_recInt) matches all values of type Numbers
// if elements of v_recInt do not break an ascending order

type record Coordinate { float x, float y };
external function fx_distance(Coordinate p_a, Coordinate p_b) return float;
template float mw_closeTo(Coordinate p_origin := { 0.0, 0.0 }, float p_maxDistance := 1.0) :=
   // access to in parameters is allowed
   @dynamic { return fx_distance(p_origin, value) <= p_maxDistance; };
   // mw_closeTo(c,d) matches all values of type Coordinate
// which have maximum distance of d from Coordinate c

external function fx_isPrime(integer p_x) return boolean;
:
p.receive(@dynamic fx_isPrime)
// is the same as p.receive(integer:@dynamic { return fx_isPrime(value) })
```

5.2 Templates with variable bindings

5.2.0 General

The possibilities to retrieve field values of received messages and signatures in ETSI ES 201 873-1 [1] are restricted and cumbersome. To overcome this situation, this clause implements the definition of templates with ***variable bindings*** that store the actual value matched by a template instance if the matching of all containing templates is successful. This feature is implemented by using the "->" symbol for denoting a value assignment to a variable or an **out** parameter in the scope of a template definition in case of a successful template matching. Such a value assignment can syntactically be specified at all places where a template matching mechanism or a template reference is used in a template definition or an inline template.

5.2.1 Value retrieval from matching

In case of a successful template match, the value which matches the template can be assigned to a variable. This can be specified by using the "->" symbol.

Syntactical Structure

TemplateInstance **"->"** *VariableRef*

Semantic Description

If a template successfully matches a value and contains a value retrieval assignment for a *TemplateInstance*, then, if during the matching process that *TemplateInstance* and all its containing *TemplateInstances* contribute to the successful template match, the part of the value the *TemplateInstance* was matching is assigned to the *VariableRef* referenced in the value retrieval assignment.

EXAMPLE:

```
template integer mw_t1 := (?, (1..3) -> v); // mw_t1 always matches, but if the (1..3) does not
                             // contribute to the successful match
                             // then v is not assigned a value
template integer mw_t2 := ((1..3) -> v_small, (3..5) -> v_big)
 // matching mw_t2 to number in (1..3) will cause only v_small to be assigned,
 // matching it to number in (4..5) will cause only v_big to be assigned (preference of (1..3))
```

Restrictions

a) If *TempateInstance* describes the matching of a mandatory element in a template definition, *VariableRef* shall refer to a variable of the same type as the mandatory element.

b) If *TempateInstance* describes the matching of an optional element in a template definition, *VariableRef* shall refer to a template variable of the same type as the optional element. In case of a successful matching, the matching value or omit shall be assigned to the template variable denoted by *VariableRef*.

c) Value retrieval shall not be used in special places as described in clause 16.1.4 of ETSI ES 201 873-1 [1] with the exception of the matching parts of receiving operations. If value retrieval is used in the matching part of a receiving operation, the assignment to the variables of all the templates is done once the whole matching part matches and all other conditions for the selection of the alternative are met.

d) Value retrieval shall not be applied to templates of set of types or any of their direct or indirect elements, elements of a permutation matching mechanism or any of their direct or indirect elements.

NOTE: TTCN-3 makes no assumptions about the value of the variable or template variable denoted by *VariableRef* My in case of an unsuccessful match.

5.2.2 Declaring out parameters for template definitions

The retrieval of field values in case of successful matching from a structured value can be specified by means of **`out`** parameters of template definitions.

Syntactical Structure

The syntactical structure for global and local template definitions does not change:

template [*restriction*] [**@fuzzy**] *Type TemplateIdentifier* ["(" *TemplateFormalParList* ")"]
[**modifies** *TemplateRef*] ":=" *TemplateBody*

Only the static semantics restriction for formal template parameters change in such a way that formal template parameters shall evaluate to **`in`** or **`out`** parameters.

Semantic Description

The semantics of the value retrieval using templates with `out` parameters can be described by means of the **`match`** operation:

```
// Given the record type definition
type record MyRecordType {
    integer     field1,
    boolean     field2
}

// the constant definition
const MyRecordType c_myRecord := {
    field1 := 7,
    field2 := true
}

// the template definition
template MyRecordType mw_myTemplate1 := {
    field1 := 7,
    field2 := ?
}

// the same template definition with an out parameter matchinstring
template MyRecordType mw_myTemplate2 (out boolean p_matchingBool):= {
    field1 := 7,
    field2 := ? -> p_matchingBool
}

// and the variable declarations
var boolean v_a, v_b;

// then the effect of
v_a := match (c_myRecord, mw_myTemplate2(v_b));
```

```
// is identical to
if (match (c_myRecord, mw_myTemplate1) {
    v_a := true;
    v_b := c_myRecord.field2        // i.e., true
}
else {
    v_a:= false;
}
```

Restrictions

a) *TempateInstance* with `out` parameters shall not be used as templates of `set of` types or any of their direct or indirect elements.

b) *TempateInstance* with `out` parameters shall not be used as elements of a permutation matching mechanism or any of their direct or indirect elements. *TempateInstance* with `out` parameters shall not used in special places as described in clause 16.1.4 of ETSI ES 201 873-1 [1] with the exception of the matching parts of receiving operations. If *TempateInstance* with `out` parameters are used in the matching part of a receiving operation, the assignment to the actual `out` parameter variables of all the templates is done only once the whole matching part matches and all other conditions for the selection of the alternative are met.

c) Every formal `out` parameter of a template with `out` parameters shall be referenced at most once inside the template body.

NOTE: In certain cases a template may match without assigning a value to a declared `out` parameter. In such cases, the boundness of the actual `out` parameter has to be checked even after a successful match before using it.

EXAMPLE 1:

```
type union Tree {
   integer leaf,
   Branch branch
}
type record Branch {
   Tree left optional,
   Tree right optional
}

template Branch mw_branch(out omit Tree p_left, out omit Tree p_right) := {
   left := * -> p_left,
   right := * -> p_right
}
template Tree mw_treebranch(out omit Tree p_left, out omit Tree p_right) :=
   { branch := mw_branch(p_left, p_right) }
template Tree mw_leaf(out integer p_value, in template integer p_expected := ?) :=
   { leaf := p_expected -> p_value }

// compute the sum of absolute values of all leafs in the given tree
function f_absSum(omit Tree p_tree) return integer {
   var omit Tree v_left, v_right;
   var integer v_value;
   if (ispresent(p_tree)) {
      select (p_tree) {
         case (mw_treebranch(v_left, v_right)) {
               return f_absSum(v_left) + f_absSum(v_right);
            }
         case (mw_leaf(v_value, (0 .. infinity)) { return v_value; }
         case (mw_leaf(v_value) { return -v_value; }
      }
   }
   else {
      return 0; }
}
```