ETSI ES 201 873-1 V4.9.1 (2017-05)

Methods for Testing and Specification (MTS);

The Testing and Test Control Notation version 3;

Part 1: TTCN‑3 Core Language

**ETSI Standard**

Reference

RES/MTS-201873 -1 T3ed491

Keywords

language, methodology, 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***

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.
The content of the PDF version shall not be modified without the written authorization of ETSI.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2017.

All rights reserved.

**DECT**TM, **PLUGTESTS**TM, **UMTS**TM and the ETSI logo are Trade Marks of ETSI registered for the benefit of its Members.
**3GPP**TM and **LTE**™ are Trade Marks of ETSI registered for the benefit of its Members and
of the 3GPP Organizational Partners.
**oneM2M** logo is protected for the benefit of its Members
**GSM**® and the GSM logo are Trade Marks registered and owned by the GSM Association.

### 6.2.5 Unions

#### 6.2.5.0 General

TTCN‑3 supports the **union** type. The **union** type is a collection of alternatives, each one identified by an identifier. Only one of the specified alternatives will ever be present in an actual union value. Union types are useful to model data which can take one of a finite number of known types.

EXAMPLE 1:

 **type** **union** MyUnionType

 {

 **integer** number,

 **charstring** string

 };

 // A valid instantiation of MyUnionType would be

 **var** MyUnionType v\_age, v\_oneYearOlder;

 **var integer** v\_ageInMonths;

 v\_age.number := 34; // value notation by referencing the field. Note, that this
 // notation makes the given field to be the chosen one

 v\_oneYearOlder := {number := v\_age.number+1};

 v\_ageInMonths := v\_age.number \* 12;

The assignment notation shall be used for **union**-s, and the notation shall assign a value to one field only. This field becomes the chosen field. Neither the not used symbol "-" nor **omit** is allowed in union value notations.

The value list notation shall not be used for setting values of **union** types.

At most one of the union alternatives can be declared as the default alternative by using the **@default** modifier before the type of the alternative. For unions with a default alternative, special type compatibility rules apply (see clause 6.3.2.4) which allow using the union value as compatible with the type of the default alternative. Therefore, the assignment notation does not have to be used to denote a value of the union type if the union's default alternative is to be chosen. Also, the default alternative selection does not have to be used to access the default alternative, if it is chosen.

***Restrictions***

In addition to the general static rules of TTCN‑3 given in clause 5, the following restrictions apply:

1. If the default alternative is of a record or set type, identifiers of fields of the record or set type and identifiers of alternatives of the union type containing the default alternative shall be distinct.
2. If the default alternative is of a union type, identifiers of alternatives of this union type and identifiers of alternatives of the union type containing the default alternative shall be distinct.
3. The **@default** alternative shall not be of the **anytype**.

EXAMPLE 2:

 **type** **union** MyDefaultUnionType

 {

 **@default** **integer** number,

 **charstring** string

 };

 // A valid instantiation of MyDefaultUnionType would be

 **var** MyDefaultUnionType v\_age, v\_oneYearOlder;

 v\_age := 34; // implicit usage of the default alternative: the integer type is

 // compatible with the default alternative; this is a shorthand notation

 // for v\_age.number := 34 or v\_age := { number := 34 }

 v\_oneYearOlder := v\_age+1; // implicit selection of the default alternative: the union

 // default alternative is compatible with integer, so that it

 // can be used as an integer expression; this is equivalient to:

 // v\_oneYearOlder.number := v\_age.number+1;

 **type** **union** MyDefaultUnionType2 {

 **@default**

 MyDefaultUnionType ageInYears,

 **integer** ageInDays

 }

 **var** MyDefaultUnionType2 v\_age2 := 3; // nested default usage: 3 is compatible with

 // both alternatives, but only alternative ageInYears

 // has @default, so this is equivalent to

 // v\_age2 := { ageInYears := 3 } which is equivalent

 // to v\_age2 := { ageInYears := { number := 3 } }

 **var** **integer** v\_result := v\_age + v\_age2; // v\_result is 37 as the expression is equivalent

 // to v\_age.number + v\_age2.ageInYears

v\_age := {string := "I feel young"};

v\_result := v\_age + v\_age2; // test case error: v\_age would be treated as

 // v\_age.number, which is not the selected alternative

 **type** **union** MyUnionTypeWithDefaultErr

 {

 **@default** MyDefaultUnionType ageInYears,

 **charstring** string // produces an error as the identifier "string" is

 }; // already used in the default alternative

#### 6.2.5.1 Referencing fields of a union type

Alternatives of a **union** type shall be referenced by the dot notation *TypeIdOrExpression.AlternativeId*, where *TypeIdOrExpression* resolves to the name of a union type or an expression of a union type such as variable, formal parameter, module parameter, constant, template, or function invocation. *AlternativeId* shall resolve to the name of an alternative in the union type or in case of an anytype value or template *AlternativeId* shall resolve to a known type name or a known type name qualified with a module name. Alternatives of union type definitions shall not reference themselves.

EXAMPLE 1:

 v\_myVar5 := v\_myUnion1.myChoice1;

 // If a union type is nested in another type then the reference may look like this

 v\_myVar6 := v\_myRecord1.myElement1.myChoice2;

 // Note, that the union type, of which the field with the identifier 'myChoice2' is referenced,

 // is embedded in a record type

If an alternative in a union type or a subtype of a union type is referenced by the dot notation, the resulting type is the set of values allowed for that alternative imposed by the constraints of the alternative declaration itself (i.e. any constraints applied to the union type itself are ignored).

When an alternative of a union type is referenced on the right hand side of an assignment an error shall occur if the referenced alternative is not the currently chosen alternative or if the referenced union field or value is omitted or uninitialized.

EXAMPLE 2:

 **type** **union** MyUnion2

 {

 **integer** choice1,

 **charstring** choice2

 }

 **type** **record** MyRecordEmbedsUnion

 {

 MyUnion2 field1 **optional**

 }

 ...

 **var** MyUnion2 v\_un2 := { choice1 := 1 }

 **var** **charstring** v\_char := v\_un2.choice2; // causes an error as v\_un.choice2 is not chosen

 **var** MyRecordEmbedsUnion v\_rec := { field1 := **omit** }

 **var** **integer** v\_int := v\_rec.field1.choice1; // causes an error as v\_rec.field1 is omitted

When referencing an alternative of a union type on the left hand side of an assignment, the referenced alternative shall become the chosen one. This rule shall apply recursively if the reference contains alternatives of nested unions, choosing all the referenced alternatives.

When referencing an alternative of an uninitialized union value or field or omitted field (including omitting a field at a higher level of the embedding hierarchy) on the left hand side of an assignment, the reference shall recursively be expanded up to and including the depth of the referenced alternative as follows:

1. When expanding a value or value field of **union** type, the alternative referenced in the dot notation becomes the chosen one.
2. Expansion of **record**, **record of**, **set**, **set of**, and **array** values and intermediate fields shall follow the rules of item a) in clauses 6.2.1.1 and 6.2.3, and clause 6.2.2.1 correspondingly.
3. At the end of the expansion, the value at the right hand side of the assignment shall be assigned to the referenced alternative.

EXAMPLE 3:

 **type** **union** MyUnion3

 {

 **integer** choice1,

 **union**

 {

 **bitstring** subchoice1,

 **charstring** subchoice2

 } choice2

 }

 ...

 **var** MyUnion3 v\_un3 := { choice1 := 1 };

 **var** MyRecordEmbedsUnion v\_rec2 := { field1 := **omit** };

 v\_un3.choice2.subchoice2 := "Hello!";

 // after the assignment v\_un3 equals to { choice2 := { subchoice2 := "Hello!" } }

 v\_rec2.field1.choice1 := 10; // after the assignment v\_rec2 equals to
 // { field1 := { choice1 := 10 } }

#### 6.2.5.2 Option and union

Optional fields are not allowed for the **union** type, which means that the **optional** keyword shall not be used with **union** types.

#### 6.2.5.3 Nested type definition for field types

TTCN‑3 supports the definition of types for union alternatives nested within the union definition, similar to the mechanism for record types described in clause 6.2.1.3.