ETSI ES 201 873-1 V4.7.1 (2015-06)

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 T3ed471

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 <http://portal.etsi.org/tb/status/status.asp>

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 2015.

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.
**GSM**® and the GSM logo are Trade Marks registered and owned by the GSM Association.

### 5.2.2 Uniqueness of identifiers

TTCN‑3 requires uniqueness of identifiers, i.e. all identifiers in the same scope hierarchy shall be distinctive. This means that a declaration in a lower level of scope shall not re-use the same identifier as a declaration in a higher level of scope in the same branch of the scope hierarchy.

The identifier of a module (its module name) or of an imported module belongs to the scope unit of the module and cannot be used as identifier for other definitions inside this module. Identifiers for fields of structured types, enumerated values and groups do not have to be globally unique, however in the case of enumerated values the identifiers shall only be reused for enumerated values within other enumerated types or as identifiers for fields of structured types. The rules of identifier uniqueness shall also apply to identifiers of formal parameters.

EXAMPLE 1: Nested scopes

 **module** MyModule

 { :

 **const** **integer** A := 1;

 :

 **function** MyBehaviourA()

 { :

 **const** **integer** A := 1; // Is NOT allowed: clash with global constant A

 **:**

 **if**(…)

 { :

 **const** **boolean** A := **true;** // Is NOT allowed: clash with local constant A

 :

 }

 }

 }

EXAMPLE 2: Independent scopes

 // The following IS allowed as the constants are not declared in the same scope hierarchy

 // (assuming there is no declaration of A in module header)

 **function** MyBehaviourA()

 { :

 **const** **integer** A := 1;

 **:**

 }

 **function** MyBehaviourB()

 { :

 **const** **integer** A := 1;

 **:**

 }

EXAMPLE 3: Module scopes

 **module** MyModuleB {

 **import** **from** MyModuleA { **…** }

 **function** MyFunction() {

 **var** **integer** MyModuleB:= 1; // Is NOT allowed: class with module name

 **:**

 }

 **type** **boolean** MyModuleA; // Is NOT allowed: class with imported module name

 }