ETSI ES 201 873-1 V4.5.1 (2013-04)

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 T3ed451 cor

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***

Individual copies of the present document can be downloaded from:
<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the 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:
<http://portal.etsi.org/chaircor/ETSI_support.asp>

***Copyright Notification***

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2013.

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.

### 6.2.9 Communication port types

Ports facilitate communication between test components and between test components and the test system interface.

TTCN‑3 supports message-based and procedure-based ports. Each port shall be defined as being message-based or procedure-based. Message-based ports shall be identified by the keyword **message** and procedure-based ports shall be identified by the keyword **procedure** within the associated port type definition.

Ports are bidirectional. The directions are specified by the keywords **in** (for the in direction), **out** (for the out direction) and **inout** (for both directions). Directions shall be seen from the point of view of the test component owning the port with the exception of the test system interface, where **in** identifies the direction of message sending or procedure call and **out** identifies the direction of message receive, get reply or catch exception from the point of view of the test component connected to the test system interface port.

Each port type definition shall have one or more lists indicating the allowed collection of (message) types or procedure signatures together with the allowed communication direction.

For configuration purposes the port type may have one **map** **param** and one **unmap param** declaration indicating the allowed additional parameters for the respective operation. These formal parameters shall be value parameters.

Whenever a signature (see also clause 14) is defined in the **out** direction of a procedure-based port, the types of all its **inout** and **out** parameters, its return type and its exception types are automatically part of the **in** direction of this port. Whenever a signature is defined in the **in** direction for a procedure-based port, the types of all its **inout** and **out** parameters, its return type and its exception types are automatically part of the **out** direction of this port.

Ports used for the communication with the SUT may need to address specific entities within the SUT. In addition, several address schemes may be supported by one SUT at different ports. To support such addressing schemes, TTCN-3 allows to bind an **address** type to a port. Values of this type may be used for addressing purposes in communication operations (see clause 22.1) and be stored in variables. The handling of address types bound to different ports by means of the dot notation is explained in clause 6.2.12.

***Syntactical Structure***

Message-based port:

**type** **port** *PortTypeIdentifier* **message** "{"

{ (**address** *Type* ";") |

 (**map** **param** "(" { *FormalValuePar* [","] }+ ")") |

 (**unmap** **param** "(" { *FormalValuePar* [","] }+ ")") |

 ((**in** | **out** | **inout**) { *MessageType* [ "," ] }+ ";") }

"}"

Procedure-based port:

**type** **port** *PortTypeIdentifier* **procedure** "{"

{ (**address** *Type* ";" ) |

 (**map** **param** "(" { *FormalValuePar* [","] }+ ")") |

 (**unmap** **param** "(" { *FormalValuePar* [","] }+ ")") |

 ((**in** | **out** | **inout**) { *Signature* [ "," ] }+ ";") }

"}"

***Restrictions***

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

a) At most one address type should be bound to a port type.

b) At most one map parameter list should be defined for a port type.

c) At most one unmap parameter list should be defined for a port type.

1. Formal parameters of **map param** and **unmap param** declarations shall not be of **port**, **component**, **timer** or **default** type or of structured types having fields of **port**, **component**, **timer** or **default** type.

***Examples***

EXAMPLE 1: Message-based port

 // Message-based port which allows types MsgType1 and MsgType2 to be received at, MsgType3 to be

 // sent via and any integer value to be send and received over the port

 **type** **port** MyMessagePortTypeOne **message**

{

 **in** MsgType1, MsgType2;

 **out** MsgType3;

 **inout integer**

 }

EXAMPLE 2: Procedure-based port

 // Procedure-based port which allows the remote call of the procedures Proc1, Proc2 and Proc3.

 // Note that Proc1, Proc2 and Proc3 are defined as signatures

 **type** **port** MyProcedurePortType **procedure**

 {

 **out** Proc1, Proc2, Proc3

 }

EXAMPLE 3: Message-based port with address type definition

 **type** **port** MyMessagePortTypeTwo **message**

{

 **address integer**; // if addressing is used on ports of type MyMessagePortTypeTwo

 // the addresses have to be of type integer

 **inout** MsgType1, MsgType2;

 }

NOTE: The term message is used to mean both messages as defined by templates and actual values resulting from expressions. Thus, the list restricting what may be used on a message-based port is simply a list of type names.

EXAMPLE 4: Usage of param in port declaration

 // Message based port which allows MsgType4 to be send and received over the port

 // and MsgType5 and MsgType6 as configuration parameter type

 **type** **port** MyMessagePortType **message**

 {

 **inout** MsgType4;

 **map** **param** (**in** MsgType5 p1, **out** MsgType6 p2);

 }

 // Procedure based port which allows the remote call of the procedure Proc1

 // and MsgType5 as configuration parameter type

 **type** **port** MyProcedurePortType **procedure**

 {

 **out** Proc1;

 **unmap** **param** (MsgType5 p1);

 }