ETSI ES 201 873-1 V4.10.1 (2018-05)

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-201873-1v4A1

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.

© ETSI 2018.

All rights reserved.

**DECT**TM, **PLUGTESTS**TM, **UMTS**TM and the ETSI logo are trademarks of ETSI registered for the benefit of its Members.
**3GPP**TM and **LTE**TM are trademarks 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 trademarks registered and owned by the GSM Association.

## 16.2 Altsteps

### 16.2.0 General

TTCN‑3 uses altsteps to specify default behaviour or to structure the alternatives of an **alt** statement.

***Syntactical Structure***

**altstep** [ **interleave** ] *AltstepIdentifier*

"(" [ { ( *FormalValuePar* | *FormalTemplatePar* ) [","] } ] ")"

[ **runs** **on** *ComponentType* ]

[ **mtc** *ComponentType* ]

[ **system** *ComponentType* ]

"{"

 { ( *VarInstance* | *TimerInstance* | *ConstDef* | *TemplateDef* ) [";"] }

 *AltGuardList*

"}"

***Semantic Description***

Altsteps are scope units similar to functions. The altstep body defines an optional set of local definitions and a set of alternatives, the so-called *top alternatives*, that form the altstep body. The syntax rules of the top alternatives are identical to the syntax rules of the alternatives of **alt** statements.

An altstep can also be used to define a named interleave statement by usage of the **interleave** keyword. In this case, the syntax rules of the top alternatives are identical to the syntax rules of the alternatives of the **interleave** statements.

NOTE 1: As an interleave statement is semantically equivalent with an alt statement, there are no further restrictions on usages of interleave altsteps than on usages of normal altsteps. They can both be used as activated default alternatives and as top-level alternatives in other alt statement blocks.The behaviour of an altstep can be defined by using the program statements and operations summarized in clause 18. Altsteps may invoke functions and altsteps or activate altsteps as defaults.

Altsteps may be parameterized as defined in clause 5.4.

Altsteps may have an **mtc** clause. If an altstep has an **mtc** clause, the type referenced by this clause shall be mtc-compatible (see clause 6.3.3) with the type of the **mtc** component reference. If the **mtc** clause is not present, the type of the **mtc** component reference is unknown in the scope of this altstep.

Altsteps may have a **system** clause. If an altstep has a **system** clause, the type referenced by this clause shall by system‑compatible (see clause 6.3.3) with the type of the **system** component reference**.** If the **system** clause is not present, the type of the **system** component reference is unknown in the scope of this altstep.

***Restrictions***

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

a) The local definitions of an altstep shall be defined before the set of alternatives.

b) The evaluation of formal parameters' default values and initialization of local definitions by calling value returning functions may have side effects. To avoid side effects that cause an inconsistency between the actual snapshot and the state of the component, and to prevent different results of subsequent evaluations on an unchanged snapshot, restrictions given in clause 16.1.4 shall apply to the formal parameters' default values and the initialization of local definitions.

c) If an altstep includes port operations or uses component variables, constants or timers the associated component type shall be referenced using the **runs on** keywords in the altstep header. The one exception to this rule is if all ports, variables, constants and timers used within the altstep are passed in as parameters.

d) An altstep without a **runs on** clause shall never invoke a function or altstep or activate an altstep as default with a **runs on** clause locally.

e) An altstep that is activated as a default shall only have **in** value or template parameters. An altstep that is only invoked as an alternative in an **alt** statement or as stand-alone statement in a TTCN‑3 behaviour description may have **in**, **out** and **inout** parameters. The rules for formal parameter lists shall be followed as defined in clause 5.4.

f) Altsteps started by using the start test component operation shall always have a runs on clause (see clause 22.5) and are considered to be invoked in the component to be started, i.e. not locally. However, the start test component operation may be invoked within behaviours without a runs on clause.

g) If the altstep is an interleave alstep, all restrictions of the interleave statement (see clause 20.4) apply to the top alternatives of the altstep as well.

***Examples***

EXAMPLE 1: Parameterized altstep with runs on clause

 // Given

 **type** **component** MyComponentType {

 **var** **integer** vc\_myIntVar := 0;

 **timer** tc\_myTimer;

 **port** myPortTypeOne pCO1, pCO2;

 **port** myPortTypeTwo pCO3;

 }

 // Altstep definition using pCO1, pCO2, vc\_myIntVar and tc\_myTimer of MyComponentType

 **altstep** a\_altSet\_A(**in integer** p\_myPar1) **runs on** MyComponentType {

 [] pCO1.**receive**(mw\_myTemplate(p\_myPar1, vc\_myIntVar)) {

 **setverdict**(**inconc**);

 }

 [] pCO2.**receive** {

 **if** (p\_myPar1 != 0) {

 **repeat**

}

 **else** {

 **break**

 }

 }

 [] tc\_myTimer.**timeout** {

 **setverdict**(**fail**);

 **stop**

 }

 }

EXAMPLE 2: Altstep with local definitions

 **altstep** a\_anotherAltStep(**in integer** p\_myPar1) **runs on** MyComponentType {

 **var integer** v\_myLocalVar := f\_myFunction(); // local variable

 **const float** c\_myFloat := 3.41; // local constant

 [] pCO1.**receive**(MyTemplate(p\_myPar1, v\_myLocalVar)) {

 **setverdict**(**inconc**);

 }

 [] pCO2.**receive** {

 **repeat**

 }

 }

EXAMPLE 3: Interleave Altstep

 **altstep interleave** a\_interleaveAltStep(**in integer** p\_myPar1, **in integer** p\_myPar2)

 **runs on** MyComponentType {

 **var integer** v\_myLocalVar := f\_myFunction(); // local variable

 [] pCO1.**receive**(MyTemplate(p\_myPar1, v\_myLocalVar)) {}

 [] pCO1.**receive**(MyTemplate(p\_myPar2, v\_myLocalVar)) {}

 }