ETSI ES 201 873-1 V4.14.1 (2022-05)

Methods for Testing and Specification (MTS);

The Testing and Test Control Notation version 3;

Part 1: TTCN‑3 Core Language

**ETSI Standard**

## 19.12 The Break statement

A **break** statement causes the exit from a loop, from a **select** statement, from an altstep or from an **alt** or **interleave** statement.

***Syntactical Structure***

**break** [*LabelIdentifier*]

***Semantic Description***

On executing a **break** statement the innermost, currently executed loop, **select** statement, **alt** statement or **interleave** statement is left. Execution continues with the statement following the construct which is left. Using **break** outside the body of a loop (**for**, **while**, **do-while**) or an alternative of an **alt** or **interleave** statement shall cause an error.

On executing a **break** statement with a label the enclosing loop, **select** statement, **alt** statement or **interleave** statement which has the same identifier as its label is left.

Altsteps are always executed within a surrounding **alt** statement. If the execution of a top alternative of an altstep (see clause 16.1.5) ends with a **break** statement, the altstep and the surrounding **alt** statement are left. Execution continues with the statement following the surrounding **alt** statement.

NOTE: TTCN-3 allows optional statement blocks that may follow altstep calls within **alt** statements. These statement blocks are not executed when the altstep is left by executing a **break** statement. A **return** statement has to be used, if such an optional statement block has to be executed (see clause Error: Reference source not found).

***Restrictions***

No specific restrictions in addition to the general static rules of TTCN‑3 given in clause Error: Reference source not found and shown in table Error: Reference source not found.

***Examples***

 **do** {

 …

 **if** (v\_cond1) {

 **break**; // the do-while loop is left

}

 …

 **for** (**var** **integer** v\_j:=1; v\_j<=10; v\_j:= v\_j+1) {

 …

 **if** (v\_cond2) {

 **break**; // the for-loop is left but the do-while loop is continued

 }

 …

 }

 …

 }

 **while** (v\_j<10);

 **label** L;

 **for** (**var** i := 0; i<n; i++) {

 **for** (**var** j := 0; j<m; j++) {

 **if** (a[i][j] > 0) {

 state := FOUND;

 **break** L;

 }

 }

 }

 **label** L;

 **var** **integer** i;

 **for** (i := 0; i<**lengthof**(a); i++) {

 **if** (a[i]) {

 **break** L; *// Is NOT allowed: label does not belong to enclosing loop.*

 }

 }

## 19.13 The Continue statement

A **continue** statement causes the start of the next iteration of a loop.

***Syntactical Structure***

**continue** [*LabelIdentifier*]

***Semantic Description***

On executing a **continue** statement, the subsequent statements of the body of the innermost, currently executed loop are skipped and the next iteration starts. Using **continue** outside the body of a loop (**for**, **while**, **do**-**while**) shall cause an error.

On executing a **continue** statement with a label, the enclosing loop which has the same identifier as its label shall skip subsequent statements and start the next iteration.

***Restrictions***

No specific restrictions in addition to the general static rules of TTCN‑3 given in clause Error: Reference source not found and shown in table Error: Reference source not found.

***Examples***

 **do** {

 …

 **if** (v\_cond) {

 **continue**; // execution continues with the next iteration of the do-while-loop

}

 …

 …

 **for** (**var** **integer** v\_j:=1; v\_j<=10; v\_j:= v\_j+1) {

 …

 **if** (v\_cond2) {

 **continue**; // continues with the next iteration of the for-loop

 }

 …

 }

 …

 }

 **while** (v\_j<10);

#### A.1.6.8.2 Behaviour statements

1BehaviourStatements ::= [TestcaseInstance](#TTestcaseInstance) |

 [FunctionInstance](#TFunctionInstance) |

 [ReturnStatement](#TReturnStatement) |

 [AltConstruct](#TAltConstruct) |

 [InterleavedConstruct](#TInterleavedConstruct) |

 [LabelStatement](#TLabelStatement) |

 [GotoStatement](#TGotoStatement) |

 [RepeatStatement](#TRepeatStatement) |

 [DeactivateStatement](#TDeactivateStatement) |

 [AltstepInstance](#TAltstepInstance) |

 [ActivateOp](#TActivateOp) |

 [BreakStatement](#TBreakStatement) |

 [ContinueStatement](#TContinueStatement)

2SetLocalVerdict ::= [SetVerdictKeyword](#TSetVerdictKeyword) "(" [SingleExpression](#TSingleExpression) {"," [LogItem](#TLogItem)} ")"

3SetVerdictKeyword ::= "setverdict"

4GetLocalVerdict ::= "getverdict"

5SUTStatements ::= [ActionKeyword](#TActionKeyword) "(" [ActionText](#TActionText) {[StringOp](#TStringOp) [ActionText](#TActionText)} ")"

6ActionKeyword ::= "action"

7ActionText ::= [FreeText](#TFreeText) | [Expression](#TExpression)

8ReturnStatement ::= [ReturnKeyword](#TReturnKeyword) [[TemplateInstance](#TTemplateInstance)]

/\* STATIC SEMANTICS - *TemplateInstance* shall evaluate to a value of a type compatible with the return type for functions returning a value. It shall evaluate to a value, template (literal or template instance), or a matching mechanism compatible with the return type for functions returning a template.\*/

9AltConstruct ::= [AltKeyword](#TAltKeyword) [ [NoDefaultModifier](#TNoDefaultModifier) ] "{" [AltstepLocalDefList](#TAltstepLocalDefList) [AltGuardList](#TAltGuardList) "}"

10AltKeyword ::= "alt"

11AltGuardList ::= {[GuardStatement](#TGuardStatement) | [ElseStatement](#TElseStatement) [[SemiColon](#TSemiColon)]}

12GuardStatement ::= [AltGuardChar](#TAltGuardChar) ([AltstepInstance](#TAltstepInstance) [[StatementBlock](#TStatementBlock)] |

 [GuardOp](#TGuardOp) [StatementBlock](#TStatementBlock))

13ElseStatement ::= "[" [ElseKeyword](#TElseKeyword) "]" [StatementBlock](#TStatementBlock)

14AltGuardChar ::= "[" [[BooleanExpression](#TBooleanExpression)] "]"

15GuardOp ::= [TimeoutStatement](#TTimeoutStatement) |

 [ReceiveStatement](#TReceiveStatement) |

 [TriggerStatement](#TTriggerStatement) |

 [GetCallStatement](#TGetCallStatement) |

 [CatchStatement](#TCatchStatement) |

 [CheckStatement](#TCheckStatement) |

 [GetReplyStatement](#TGetReplyStatement) |

 [DoneStatement](#TDoneStatement) |

 [KilledStatement](#TKilledStatement)

16InterleavedConstruct ::= [InterleavedKeyword](#TInterleavedKeyword) [ [NoDefaultModifier](#TNoDefaultModifier) ] "{" [InterleavedGuardList](#TInterleavedGuardList) "}"

17InterleavedKeyword ::= "interleave"

18InterleavedGuardList ::= {[InterleavedGuardElement](#TInterleavedGuardElement) [[SemiColon](#TSemiColon)]}+

19InterleavedGuardElement ::= [InterleavedGuard](#TInterleavedGuard) [StatementBlock](#TStatementBlock)

20InterleavedGuard ::= "[" "]" [GuardOp](#TGuardOp)

21LabelStatement ::= [LabelKeyword](#TLabelKeyword) [Identifier](#TIdentifier)

22LabelKeyword ::= "label"

23GotoStatement ::= [GotoKeyword](#TGotoKeyword) [Identifier](#TIdentifier)

24GotoKeyword ::= "goto"

25RepeatStatement ::= "repeat"

26ActivateOp ::= [ActivateKeyword](#TActivateKeyword) "(" [AltstepInstance](#TAltstepInstance) ")"

27ActivateKeyword ::= "activate"

28DeactivateStatement ::= [DeactivateKeyword](#TDeactivateKeyword) ["("ObjectReference ")"]

29DeactivateKeyword ::= "deactivate"

30BreakStatement ::= "break" [Identifier]

31ContinueStatement ::= "continue" [Identifier]