# ETSI ES 203 119-4 V1.1.1 (2015-06)


**Methods for Testing and Specification (MTS);**
**The Test Description Language (TDL);**
**Part 4: Structured Test Objective Specification (Extension)**

Reference
DES/MTS-203119-4

Keywords
language, MBT, methodology, testing, TSS&TP, TTCN-3, UML

***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

# Contents

# Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

# Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

The present document is part 4 of a multi-part deliverable covering the Test Description Language as identified below:

Part 1: "Abstract Syntax and Associated Semantics";

Part 2: "Graphical Syntax";

Part 3: "Exchange Format";

**Part 4: "Structured Test Objective Specification (Extension)".**

# Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

# Introduction

Test purposes play an essential role in test specification processes at ETSI. Currently, TDL treats test purposes, and test objectives in general as informal text without any additional structural constraints. This extension package for TDL refines and formalizes test objective specification within TDL by introducing relevant meta-model concepts and a corresponding syntactical notation, both of which are related to TPLan ETSI ES 202 553 [i.1] and TPLan-like notations already established at ETSI. This enables test purpose specification to enter the modelling world and paves the way for improved tool support and better structured test objectives, as well as additional formal verification and validation facilities down the road by integrating and unifying the means for the specification of test purposes and test descriptions, while relying on the same underlying meta-model and benefiting from other related technologies built around this meta-model.

The present document describes the relevant abstract syntax (meta-model) extensions as well as the corresponding concrete syntactical notation.

# 1 Scope

The present document specifies an extension of the Test Description Language (TDL) enabling the specification of structured test objectives. The extension covers the necessary additional constructs in the abstract syntax, their semantics, as well as the concrete syntactical notation for the added constructs. The intended use of the present document is to serve both as a foundation for TDL tools implementing support for the specification of structured test objectives, as well as a reference for end users applying the standardized syntax for the specification of structured test objectives with TDL.

# 2 References

## 2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI ES 203 119-1 (V1.2.1): "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 1: Abstract Syntax and Associated Semantics".

[2] ETSI ES 203 119-2 (V1.1.1): "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 2: Graphical Syntax".

[3] ETSI ES 203 119-3 (V1.1.1): "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 3: Exchange Format".

## 2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI ES 202 553 (V1.2.1): "Methods for Testing and Specification (MTS); TPLan: A notation for expressing Test Purposes".

[i.2] ETSI TS 136 523-1 (V10.2.0): "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Packet Core (EPC); User Equipment (UE) conformance specification; Part 1: Protocol conformance specification (3GPP TS 36.523-1 version 10.2.0 Release 10)".

[i.3] ETSI TS 186 011-2: "Core Network and Interoperability Testing (INT); IMS NNI Interoperability Test Specifications (3GPP Release 10); Part 2: Test descriptions for IMS NNI Interoperability".

# 3 Definitions and abbreviations

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in ETSI ES 203 119-1 [1] and the following apply:

**context:** set of circumstances related to the occurrence of an event

**entity:** object that may be involved in the occurrence of an event as part of a specific context

**entity type:** alias for additional meta-information that may be used to describe one or more entities

**event:** observable phenomenon or state that may occur in a specific context

NOTE: Related to a term of the same name defined in ETSI ES 202 553 [i.1].

**event occurrence:** description of the occurrence of an event in a specific context

**event type:** alias for additional meta-information that may be used to describe one or more events

## 3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

| | |
|---|---|
| BNF | Backus-Naur Form |
| EBNF | Extended Backus-Naur Form |
| IMS | IP Multimedia Subsystem |
| IUT | Implementation Under Test |
| PICS | Protocol Implementation Conformance Statement |
| SUT | System Under Test |
| TDL | Test Description Language |
| TPLan | Test Purpose Notation |

# 4 Basic principles

## 4.1 Structured Test Objective Specification

The present document defines an extension for TDL enabling the specification of structured test objectives. Rather than rely on external documents or informal text provided by the default test objective specification facilities of TDL, this extension enables users to describe test objectives in a more structured and formalized manner which can enable subsequent generation of test description skeletons and consistency checking against test descriptions realizing a given test objective. In addition, the structured approach to test objective specification also enables syntactical and semantical consistency checking of the test objectives themselves.

The abstract concepts and the concrete syntax are based on TPLan ETSI ES 202 553 [i.1] to a large extent, as they also reflect concepts and practices already established at ETSI. The fundamental concept in the specification of a structured test objectives is the event occurrence which describes the occurrence of an abstract event in a specific context, comprising one or more involved entities, an event argument, as well as a time label and/or a time constraint.

Events and entities referenced in an event occurrence need to be defined in advance as part of a domain description which can then be reused across all structured test objective specifications in that domain. An entity is an abstract representation of an object involved in an event occurrence that may be realized as a component instance or a gate instance within a test description realizing the structured test objective.

An event argument may either refer to a data instance for data already defined with the facilities provided by TDL, or, following a more light weight approach, describe data inline without the need to define all data types and instances in advance. Pre-defined data and inline data can be integrated to a certain degree in that inline data may refer to pre-defined data, but not the other way around.

Event occurrence specifications are organized in the different compartments of a structured test objective, including initial conditions, expected behaviour, and final conditions. Multiple event occurrences are combined by means of an 'and' or 'or' operand indicating how subsequent event occurrences are related to each other (as a sequence or as alternatives, respectively).

Structured test objectives may also include references to PICS which may be used as selection criteria for the concrete realization of the test objectives. The PICS need to be defined in advance as part of the domain description. Multiple PICS references within the same structured test objective are combined by means of an 'and' or 'or' operand indicating how subsequent referenced PICS are related to each other.

## 4.2 Document Structure

The present document defines the structured test objective specification extension for TDL comprising:

- Meta-model extension describing additional concepts required for the specification of structured test objectives (clause 5).
- Concrete syntax extension describing corresponding shapes for the representation of the additional concepts (clause 6).
- An informative annex with examples in a textual concrete syntax (annex A).
- An informative annex with production rules for the example textual syntax (annex B).

## 4.3 Notational Conventions

The present document inherits the notational conventions defined in ETSI ES 203 119-1 [1] and ETSI ES 203 119-2 [2].

The abstract syntax specification and the classifier descriptions follow the notational conventions defined in clause 4.5 of Abstract Syntax and Associated Semantics [1]. The concrete syntax notation specification follows the notational conventions described in clause 4.5 of the Graphical Syntax [2].

## 4.4 Conformance

For an implementation claiming to conform to this extension of TDL meta-model, all concepts specified in the present document and in ETSI ES 203 119-1 [1], as well as the concrete syntax representation specified in the present document shall be implemented consistently with the requirements given in the present document and the referenced documents. The electronic attachment from annex A in ETSI ES 203 119-1 [1] can serve as a starting point for a TDL meta-model implementation conforming to the present document and the overall abstract syntax of TDL [1].

# 5 Meta-Model Extensions

## 5.1 Overview

The structured test objective specification is defined within a single package in the TDL meta-model. It relies on several concepts from the 'Foundation', 'Data', and 'Time' packages of the TDL meta-model.

## 5.2 Foundation Abstract Syntax and Classifier Description

### 5.2.1 Entity

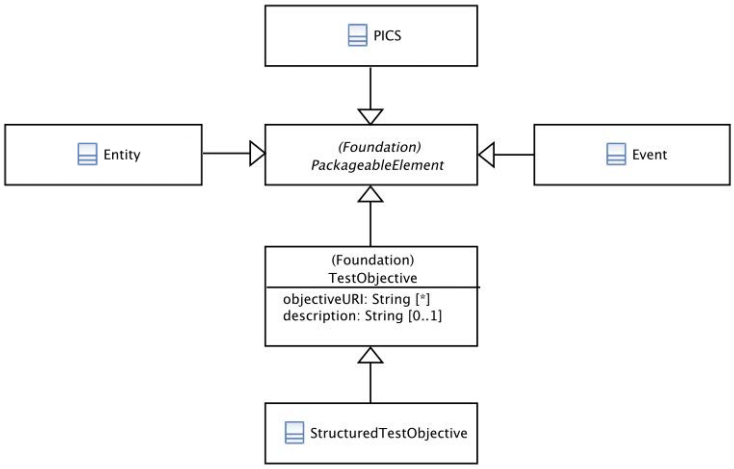



**Figure 5.1: Structured Test Objective Specification Foundation Concepts**

#### Semantics

An 'Entity' is a 'PackageableElement' that describes a participant in an 'EventOccurrence'. User defined entities, such as IUT, SUT, Tester, etc. may be referenced by means of an 'EntityReference' within an 'EventOccurrence' as the source and/or target of an 'Event' referenced in a corresponding 'EventReference'. Whether an 'Entity' corresponds a 'ComponentInstance' or a 'GateInstance' is not specified in advance. 'Annotation's may be used to provide an indication for the type and role of the 'Entity'.

#### Generalizations

- PackageableElement

#### Properties

There are no properties specified.

#### Constraints

There are no constraints specified.

### 5.2.2 Event

#### Semantics

An 'Event' is a 'PackageableElement' that describes a user defined event or activity that may be referenced in an 'EventOccurrence'. The direction of an 'Event' with respect to the 'Entity' or 'Entity's referenced in the 'EventOccurrence' depends on the interpretation of the 'Event', where 'Annotation's may be used to provide additional information as an indication of the intended interpretation.

#### Generalizations

- PackageableElement

#### Properties

There are no properties specified.

#### Constraints

There are no constraints specified.

### 5.2.3 PICS

Semantics

A 'PICS' is a 'PackageableElement' that may be referenced in 'StructuredTestObjective's to indicate selection criteria for the 'StructuredTestObjective' based on features required for and/or tested with the realization of the 'StructuredTestObjective'.

Generalizations

- PackageableElement

Properties

There are no properties specified.

Constraints

There are no constraints specified.

## 5.3 Test Objective Abstract Syntax and Classifier Description

### 5.3.1 StructuredTestObjective

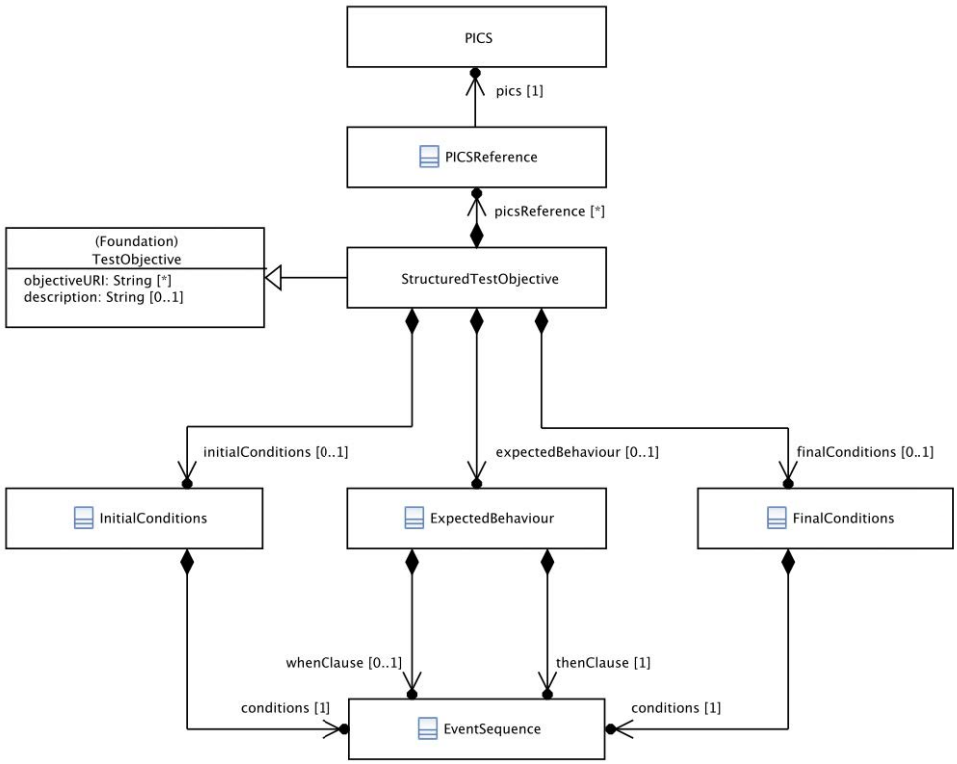



**Figure 5.2: Structured Test Objective Concepts**

Semantics

A 'StructuredTestObjective' is a refinement of 'TestObjective' that enables the use of additional constructs in order to formalize the description of 'TestObjective's. In addition to the 'description' and 'objectiveURI' properties inherited from 'TestObjective', a 'StructuredTestObjective' includes 'PICSReferences', 'InitialConditions', 'ExpectedBehaviour', and 'FinalConditions'.

Generalizations

- TestObjective

Properties

- picsReference : PICSReference [*] {ordered}
  An ordered set of 'PICSReferences' to 'PICS'.
- initialConditions : InitialConditions [0..1]
  Initial conditions description for the 'StructuredTestObjective'.
- expectedBehaviour : ExpectedBehaviour [0..1]
  Expected behaviour description for the 'StructuredTestObjective'.
- finalConditions : FinalConditions [0..1]
  Final conditions description for the 'StructuredTestObjective'.

Constraints

There are no constraints specified.

### 5.3.2 PICSReference

Semantics

A 'PICSReference' is an 'Element' that enables the referencing of 'PICS' within a 'StructuredTestObjective'. A 'Comment' with body containing an 'and' or 'or' shall be used as a Boolean operand if there are two or more 'PICSReference's specified within a 'StructuredTestObjective', starting with the second 'PICSReference' to indicate how the referenced 'PICS' shall be interpreted with regard to the other referenced 'PICS' within the same 'StructuredTestObjective'.

Generalizations

- Element

Properties

- pics : PICS [1]
  The referenced 'PICS'.

Constraints

- **Combining Multiple 'PICSReference's**
  A 'Comment' with body containing an 'and' or 'or' shall be attached to the 'PICSReference' as a Boolean operand if there are two or more 'PICSReference's and it is not the first 'PICSReference'.

### 5.3.3 InitialConditions

Semantics

'InitialConditions' is an 'Element' containing an 'EventSequence' describing the initial conditions of a 'StructuredTestObjective'.

Generalizations

- Element

Properties

- conditions : EventSequence [1]
  An 'EventSequence' containing the 'EventOccurrence's describing the initial conditions for the 'StructuredTestObjective'.

Constraints

There are no constraints specified.

### 5.3.4 ExpectedBehaviour

Semantics

'ExpectedBehaviour' is an 'Element' containing an 'EventSequence' describing the expected behaviour specified in a 'StructuredTestObjective'.

Generalizations

- Element

Properties

- whenClause : EventSequence [0..1]
  An 'EventSequence' containing the 'EventOccurrence's describing the stimuli for the 'ExpectedBehaviour' of the 'StructuredTestObjective'.

- thenClause : EventSequence [1]
  An 'EventSequence' containing the 'EventOccurrence's describing the expected reaction for the 'ExpectedBehaviour' of the 'StructuredTestObjective' or the resulting expected state.

Constraints

There are no constraints specified.

### 5.3.5 FinalConditions

Semantics

'FinalConditions' is an 'Element' containing an 'EventSequence' describing the final conditions of a 'StructuredTestObjective'.

Generalizations

- Element

Properties

- conditions : EventSequence [1]
  An 'EventSequence' containing the 'EventOccurrence's describing the final conditions for the 'StructuredTestObjective'.

Constraints

There are no constraints specified.

## 5.4 Events Abstract Syntax and Classifier Description

### 5.4.1 EventSequence

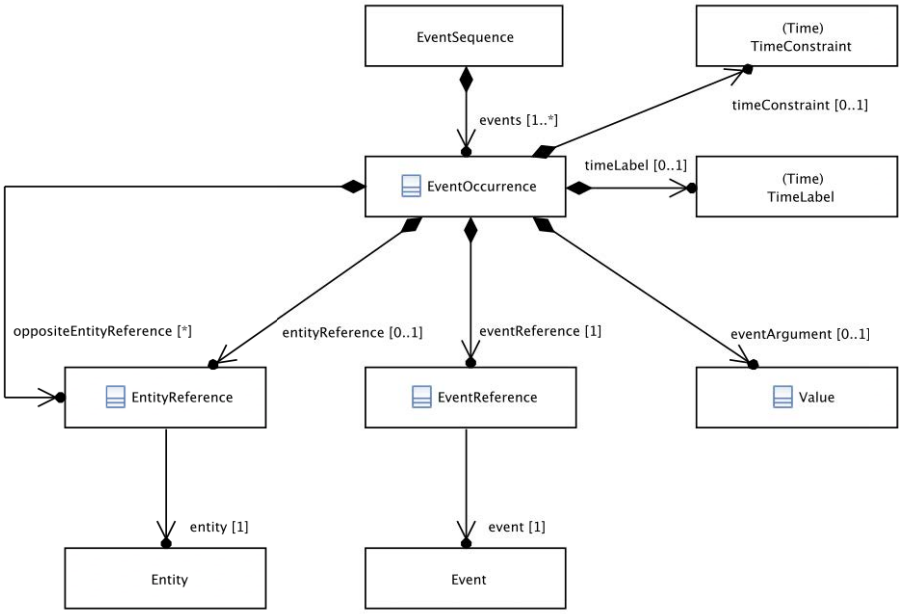



**Figure 5.3: Events Concepts**

Semantics

'EventSequence' is an 'Element' containing 'EventOccurrence's.

Generalizations

- Element

Properties

- events : EventOccurrence [1..*] {ordered}
  A sequence of 'EventOccurrence's.

Constraints

There are no constraints specified.

### 5.4.2 EventOccurrence

Semantics

An 'EventOccurrence' is an 'Element' describing a concrete occurrence of an 'Event', including qualified references to the 'Event', to the 'Entity' related to the occurrence of the 'Event' and to any other 'Entity's involved in the 'EventOccurrence'. It also includes a 'Value' as an argument describing the details of the 'EventOccurrence' such as the data being sent or received, or a state an involved 'Entity' is in. The 'EventOccurrence' also includes an optional 'TimeLabel' and/or a 'TimeConstraint' for the specification of temporal relationships between 'EventOccurrence's. In case there is more than one 'EventOccurrence' within an 'EventSequence', a 'Comment' with body containing an 'and' or 'or' shall be used as an operand, starting with the second 'EventOccurrence' to indicate how the 'EventOccurrence' shall be related to the previous 'EventOccurrence' within the same 'EventSequence', i.e. whether both 'EventOccurrence's are required or whether only one of the 'EventOccurrence's shall take place. The 'or' operand takes precedence, thus given an 'EventSequence' *EO1* and *EO2* or *EO3*, the intended interpretation is that *EO1* takes place followed by *EO2* or *EO3* taking place. While this is opposite to conventional logical operator precedence (i.e. 'and' takes precedence over 'or'), conventional logical operator precedence is not applicable in the context of 'EventOccurrence's as the intended interpretation shall be implementable by means of an 'AlternativeBehaviour' or a 'ConditionalBehaviour' in TDL.

Additional 'Comment's may be added to describe the 'EventOccurrence'.

Generalizations

- Element

Properties

- entityReference : EntityReference [0..1]
  An 'EntityReference' to the 'Entity' related to the occurrence of the 'Event'.
- oppositeEntityReference : EntityReference [0..*]
  'EntityReference's to other 'Entity's involved in the 'EventOccurrence'.
- eventReference : EventReference [1]
  An 'EventReference' to the occurring 'Event'.
- eventArgument : Value [0..1]
  A 'Value' describing the details of the 'EventOccurrence'.
- timeLabel : TimeLabel [0..1]
  A 'TimeLabel' that may be added to the 'EventOccurrence' in order to be able to specify 'TimeConstraint's for subsequent 'EventOccurrence's with relation to the 'EventOccurrence'.
- timeConstraint : TimeConstraint [0..1]
  A 'TimeConstraint' that may be added to the 'EventOccurrence' to describe temporal relationships to previous 'EventOccurrence's.

Constraints

- **Combining Multiple 'EventOccurrence's**
  A 'Comment' with body containing an 'and' or 'or' shall be attached to the 'EventOccurrence' as an operand if there are two or more 'EventOccurrence's and it is not the first 'EventOccurrence'.
- **'TimeConstraint' expression restriction**
  Only a 'DataInstanceUse' referring to a 'TimeLabel' may be used in 'TimeConstraint's contained in 'EventOccurrence's.

### 5.4.3 EntityReference

Semantics

An 'EntityReference' is an 'Element' that enables the referencing of 'Entity's within 'EventOccurrence's. 'Comment's may be used to add qualifiers describing peculiarities of the referenced 'Entity' related to the specific 'EventOccurrence'.

Generalizations

- Element

Properties

- entity : Entity [1]
  The referenced 'Entity'.

Constraints

There are no constraints specified.

### 5.4.4 EventReference

Semantics

An 'EventReference' is an 'Element' that enables the referencing of 'Events' within 'EventOccurrence's. 'Comment's may be used to add qualifiers describing peculiarities of the referenced 'Event' related to the specific 'EventOccurrence'.

Generalizations

- Element

Properties

- event : Event [1]
  The referenced 'Event'.

Constraints

There are no constraints specified.

## 5.5 Data Abstract Syntax and Classifier Description

### 5.5.1 Value

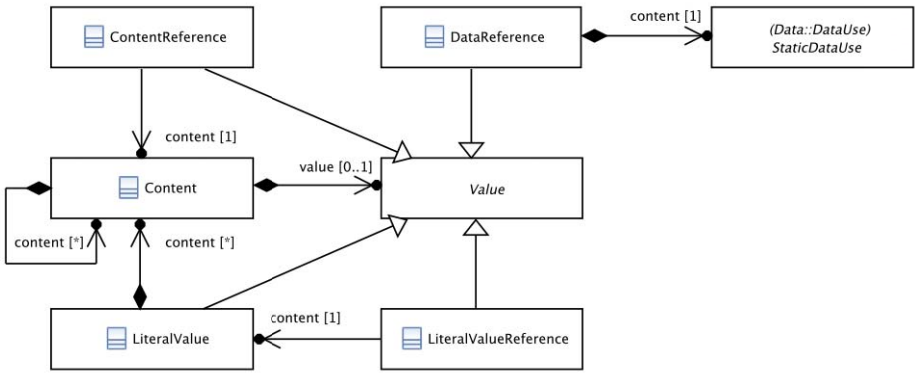



**Figure 5.4: Data Concepts**

Semantics

A 'Value' is an abstract 'Element' that is refined into 'DataReference', 'LiteralValue', 'LiteralValueReference' and 'ContentReference'. A 'DataReference' enables the referencing of 'DataInstance's defined in advance, as well as the corresponding 'AnyValue', 'AnyValueOrOmit', and 'OmitValue' specifications for a predefined 'DataType'. The remaining 'Value' refinements enable the inline description of data content and data structures, without the requirement of defining 'DataType's and 'DataInstance's in advance. 'DataInstance's and inline data descriptions may be combined to the extent that inline data descriptions may contain 'DataReference's to 'DataInstance's, but 'DataInstance's relying on declared 'DataType's may not reference inline data descriptions. 'Comment's may be used to add qualifiers describing further details related to the 'Value' with regard to the specific context of its usage. With the exception of 'DataInstance's, all inline descriptions are only visible within the containing 'StructuredTestObjective' and may only be referenced within the same 'StructuredTestObjective', where only 'LiteralValue's and 'Content' used in previous 'EventOccurrence's may be referenced in subsequent 'EventOccurrence's.

Generalizations

- Element

Properties

There are no properties specified.

Constraints

There are no constraints specified.

### 5.5.2 LiteralValue

#### Semantics

A 'LiteralValue' is a 'Value' that represents any literal label used as an argument of an 'EventOccurrence' or as a value of 'Content'. 'Comment's may be used to provide additional information related to the type and semantics of the 'LiteralValue'. A 'LiteralValue' may contain 'Content's enabling the definition of a substructure of the 'LiteralValue' that describes the details of the 'LiteralValue'.

#### Generalizations

- Value

#### Properties

- content : Content [0..*] {ordered}
  The 'Content's of the 'LiteralValue'.

#### Constraints

There are no constraints specified.

### 5.5.3 Content

#### Semantics

A 'Content' is an 'Element' that enables the specification of composite 'LiteralValue's which contain additional 'Value's assigned to the 'Content'. Alternatively, 'Content' may contain nested 'Content' without specifying a 'Value' enabling the specification of relevant sub-structures without full details of the 'Values' assigned to each structural feature.

#### Generalizations

- Element

#### Properties

- content : Content [0..*] {ordered}
  Nested contents of the 'Content'.

- value : Value [0..1]
  A 'Value' assigned to the 'Content'.

#### Constraints

- **No nested 'Content's if 'Value' is provided**
  Either nested 'Content's or 'Value' may be specified within 'Content', but not both.

### 5.5.4 LiteralValueReference

#### Semantics

A 'LiteralValueReference' is a 'Value' that enables the referencing of 'LiteralValues' from previous 'EventOccurrence's within the containing 'StructuredTestObjective' as an argument of an 'EventOccurrence' or as a value of 'Content'.

#### Generalizations

- Value

#### Properties

- content : LiteralValue [1]
  The referenced 'LiteralValue'.

Constraints

- **Referenced 'LiteralValue' visibility**
  Only 'LiteralValue's defined within previous 'EventOccurrence's of the containing 'StructuredTestObjective' may be referenced.

### 5.5.5 ContentReference

Semantics

A 'ContentReference' is a 'Value' that enables the referencing of the 'Content' of 'LiteralValues' from previous 'EventOccurrence's within the containing 'StructuredTestObjective' as an argument of an 'EventOccurrence' or as a value of 'Content'.

Generalizations

- Value

Properties

- content : Content [1]
  The referenced 'Content'.

Constraints

- **Referenced 'Content' visibility**
  Only 'Content' defined within previous 'EventOccurrence's of the containing 'StructuredTestObjective' may be referenced.

### 5.5.6 DataReference

Semantics

A 'DataReference' is a 'Value' that enables the referencing of 'DataInstance's by means of a 'DataInstanceUse', as well as the use of 'AnyValue', 'AnyValueOrOmit', and 'OmitValue' specifications for a predefined 'DataType' as an argument of 'EventOccurrence's or as a value of 'Content'.

Generalizations

- Value

Properties

- content : StaticDataUse [1]
  Specification of the referenced 'DataInstance'.

Constraints

- **'DataInstanceUse' restrictions within 'DataReference'**
  Only simple and structured data instances may be referenced directly or indirectly in 'ParameterBinding's of the 'StaticDataUse' within a 'DataReference'.

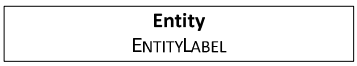
- **No 'reduction' within 'DataReference'**
  The 'reduction' property of 'StaticDataUse' inherited from 'DataUse' shall not be used within a 'DataReference'.

# 6 Graphical Syntax Extensions

## 6.1 Foundation

### 6.1.1 Entity

Concrete Graphical Notation




Formal Description

**context Entity**

ENTITYLABEL ::= self.name

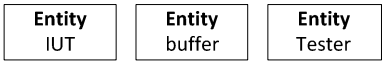
Constraints

There are no constraints specified.

Comments

No comments.

Example




### 6.1.2 Event

Concrete Graphical Notation


Formal Description

**context Event**

EVENTLABEL ::= self.name

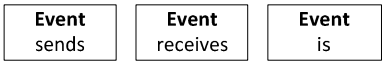
Constraints

There are no constraints specified.

Comments

No comments.

Example




### 6.1.3 PICS

Concrete Graphical Notation


Formal Description

**context PICS**

PICSLABEL ::= self.name

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

| PICS | Entity | Entity |
|---|---|---|
| PICS_F1 | PICS_F2 | PICS_F3 |

## 6.1.4 Comment

### Concrete Graphical Notation

Inherited from ETSI ES 203 119-2 [2] for 'Comment's not contained in a 'StructuredTestObjective', overridden for 'Comment's directly or indirectly contained in a 'StructuredTestObjective'.

### Formal Description

```
context Comment
QUALIFIER ::= self.body

ANDORQUALIFIER ::=  if self.body = 'and'
                    or self.body = 'or'
                    then
                        self.body
                    endif

ARTICLEQUALIFIER ::=  if self.body = 'a'
                      or self.body = 'an'
                      or self.body = 'the'
                      then
                          self.body
                      endif

ASSIGNMENTQUALIFIER ::=  if self.body = 'indicating value'
                         or self.body = 'set to'
                         then
                             self.body
                         endif

COMMONWORDQUALIFIER ::=  if self.body = 'after'
                         or self.body = 'before'
                         or self.body = 'from'
                         or self.body = 'of'
                         or self.body = 'to'
                         then
                             self.body
                         endif

DIRECTIONQUALIFIER ::=  if self.body = 'by'
                        or self.body = 'for'
                        or self.body = 'from'
                        or self.body = 'in'
                        or self.body = 'into'
                        or self.body = 'to'
                        then
                            self.body
                        endif
```

```
QUANTIFIEDQUALIFIER ::=  if self.body = 'all'
                         or self.body = 'any'
                         or self.body = 'few'
                         or self.body = 'multiple'
                         or self.body = 'no'
                         or self.body = 'only'
                         or self.body = 'several'
                         or self.body = 'some'
                         then
                             self.body
                         endif

REFERENCEQUALIFIER ::=   if self.body = 'associated with'
                         or self.body = 'carrying'
                         or self.body = 'contained in'
                         or self.body = 'corresponding to'
                         or self.body = 'derived from'
                         then
                             self.body
                         endif

TIMECONSTRAINTQUALIFIER ::=  if self.body = 'after'
                             or self.body = 'before'
                             or self.body = 'during'
                             or self.body = 'within'
                             then
                                 self.body
                             endif

NOTELABEL ::= '(' 'Note' self.name ':' self.body ')'
```

### Constraints

- **Default comment label**
  The QUALIFIER label only applies to 'Comment's that do not match the conditions for any of the other qualifier labels.

### Comments

No comments.

### Example

Not available.

# 6.2 Test Objective

## 6.2.1 StructuredTestObjective

### Concrete Graphical Notation

| | |
|---|---|
| **TP Id** | TESTOBJECTIVENAMELABEL |
| **Test Objective** | DESCRIPTIONLABEL |
| **Reference** | URIOFOBJECTIVELABEL |
| **PICS Selection** | <PICSSELECTIONLABEL> |
| **Initial Conditions** | |
| INITIALCONDITIONSLABEL | |
| **Expected Behaviour** | |
| EXPECTEDBEHAVIOURLABEL | |
| **Final Conditions** | |
| FINALCONDITIONSLABEL | |

| **Test Purpose**<br>TESTOBJECTIVENAMELABEL |
|---|
| **Test Objective**<br>DESCRIPTIONLABEL |
| **Reference**<br>URIOFOBJECTIVELABEL |
| **PICS Selection**<br><PICSSELECTIONLABEL> |
| **Initial Conditions**<br>INITIALCONDITIONSLABEL |
| **Expected Behaviour**<br>EXPECTEDBEHAVIOURLABEL |
| **Final Conditions**<br>FINALCONDITIONSLABEL |

### Formal Description

**context StructuredTestObjective**

TESTOBJECTIVENAMELABEL ::= self.name

DESCRIPTIONLABEL ::= self.description

URIOFOBJECTIVELABEL ::= self.objectiveURI->**newline**()

PICSSELECTIONLABEL ::= **foreach** p:PICSReference **in** self.picsReferences p **as context in** <PICSREFERENCELABEL> **end**

PICSREFERENCELABEL ::= [p.comment->**first**() **as context in** <ANDORQUALIFIER>] p.pics.name

```
INITIALCONDITIONSLABEL ::= 'with' '{'
                               self.initialConditions.conditions as context in <EVENTSEQUENCELABEL>
                           '}'

EXPECTEDBEHAVIOURLABEL ::= 'ensure' 'that' '{'
                               if self.expectedBehaviour.whenClause.oclIsUndefined() then
                                   self.expectedBehaviour.thenClause as context in <EVENTSEQUENCELABEL>
                               else
                                   'when' '{'
                                       self.expectedBehaviour.whenClause as context in <EVENTSEQUENCELABEL>
                                   '}'
                                   'then' '{'
                                       self.expectedBehaviour.thenClause as context in <EVENTSEQUENCELABEL>
                                   '}'
                               endif
                           '}'
```

FINALCONDITIONSLABEL ::= 'with' '{'
self.finalConditions.conditions **as context in** <EVENTSEQUENCELABEL>
'}'

### Constraints

- **Spaces in the 'name' of an 'Element' and the 'body' of a 'Comment'**
  A 'name' of an 'Element' or a 'body' of a 'Comment' shall be enclosed in single or double quotes when the corresponding 'Element' or 'Comment' is contained within a 'PICSReference' or an 'EventSequence'.

### Comments

The labels for the DESCRIPTIONLABEL, URIOFOBJECTIVELABEL, and PICSSELECTIONLABEL are optional and displayed only if the respective model elements are defined. The corresponding compartments are always displayed.

The compartments containing the INITIALCONDITIONSLABEL, the EXPECTEDBEHAVIOURLABEL, and the FINALCONDITIONSLABEL are optional and displayed only if the respective model elements are defined. The corresponding headings containing the keywords **Initial Conditions**, **Expected Behaviour**, and **Final Conditions** are always displayed.

In the alternate notation shown above, all compartments except the TestObjective compartment are optional and only displayed if the respective model elements are defined.

### Example

| | |
|---|---|
| **TP Id** | TP/GEONW/FDV/BAH/BV/01 |
| **Test Objective** | Check defined values of default Gn parameters in the basic header |
| **Reference** | |
| **PICS Selection** | PICS_F1 |
| **Initial Conditions** | |
| with {<br>the IUT entity being in the initial *state*<br>} | |
| **Expected Behaviour** | |
| ensure that{<br>when {<br>the IUT entity is requested to send a *"GUC packet"*<br>}<br>then {<br>the IUT entity sends a *"GUC packet"* containing<br>*BasicHeader* containing<br>*"version field"* indicating value *"itsGnProtocolVersion MIB parameter"* ,<br>*"RHL field"* indicating value *"itsGnDefaultHopLimit MIB parameter"*<br>;<br>;<br>}<br>} | |
| **Final Conditions** | |

# 6.3 Events

## 6.3.1 EventSequence

### Concrete Graphical Notation

There is no shape associated with this element. Instead, it is represented as a label within the context of a 'StructuredTestObjective'.

### Formal Description

**context EventSequence**
EVENTSEQUENCELABEL ::= **foreach** e:EventOccurrence **in** self.events **newline**() e **as context in** <EVENTOCCURRENCELABEL> **end**

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
the IUT entity being in the initial state and
the IUT entity using a "CBF algorithm" and
the IUT entity having received a "Beacon information" from the ItsNodeB or
the IUT entity having received any message from the ItsNodeD
```

## 6.3.2 EventOccurrence

### Concrete Graphical Notation

There is no shape associated with this element. Instead, it is represented as a label within the context of a 'StructuredTestObjective'.

### Formal Description

```
context EventOccurrence
EVENTOCCURRENCELABEL ::= [self.comment->first() as context in <ANDORQUALIFIER>]
                         if self.timeLabel.oclIsUndefined() then
                             if not self.timeConstraint.oclIsUndefined() then
                                 self.timeConstraint as context in <TIMECONSTRAINTLABEL>
                             endif
                         else
                             self.timeLabel as context in <TIMELABELLABEL>
                             if self.timeConstraint.oclIsUndefined() then
                                 ':'
                             else
                                 ',' self.timeConstraint as context in <TIMECONSTRAINTLABEL>
                             endif
                         endif
                         [self.entityReference as context in <ENTITYREFERENCELABEL>]
                         self.eventReference as context in <EVENTREFERENCELABEL>
                         [self.eventArgument as context in <EVENTARGUMENTLABEL>]
                         [foreach e:EntityReference in self.oppositeEntityReference separator(',') e as context in <OppositeENTITYLABEL> end]
                         [foreach c:Comment in self.comment separator(',') e as context in <NoteLABEL> end]
```

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
the IUT entity having received a "Beacon information" from the ItsNodeB
(Note 1: "Beacon information may be incomplete")
at time point t the IUT entity receives a "message"
at time point t2, 3s after t : the IUT entity sends an invitation to the ItsNodeD
1s after t : the IUT entity receives a confirmation from the ItsNodeD
```

## 6.3.3 EntityReference

### Concrete Graphical Notation

There is no shape associated with this element. Instead, it is represented as a label within the context of a 'StructuredTestObjective'.

### Formal Description

**context EntityReference**

ENTITYREFERENCELABEL ::= self.comment->**first**() **as context in** <ARTICLEQUALIFIER>
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.entity.name
'entity'

OPPOSITEENTITYLABEL ::= self.comment->**first**() **as context in** <DIRECTIONQUALIFIER>
self.comment->**at**(1) **as context in** <ARTICLEQUALIFIER>
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.entity.name

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
the IUT entity
from the ItsNodeB
in the location service buffer, for ItsNodeB
```

## 6.3.4 EventReference

### Concrete Graphical Notation

There is no shape associated with this element. Instead, it is represented as a label within the context of a 'StructuredTestObjective'.

### Formal Description

**context EventReference**

EVENTREFERENCELABEL ::= [**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.event.name

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
being in
having automatically received
sends
```

# 6.4 Data

## 6.4.1 Value

### Concrete Graphical Notation

There is no shape associated with this element as it is abstract.

### Formal Description

**context Value**

EVENTARGUMENTLABEL ::= **if** self.**oclIsTypeOf**(DataReference) **then** self **as context in** <DATAREFERENCEARGUMENTLABEL>

**else if** self.**oclIsTypeOf**(LiteralValue) **then** self **as context in** <LITERALVALUEARGUMENTLABEL>
**else if** self.**oclIsTypeOf**(LiteralValueReference) **then** self **as context in** <LITERALVALUEREFERENCEARGUMENTLABEL>
**else if** self.**oclIsTypeOf**(ContentReference) **then** self **as context in** <CONTENTREFERENCEARGUMENTLABEL>
**endif**

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

Not available.

## 6.4.2 LiteralValue

### Concrete Graphical Notation

There is no shape associated with this element. Instead, it is represented as a label within the context of a 'StructuredTestObjective'.

### Formal Description

**context LiteralValue**

LITERALVALUEARGUMENTLABEL ::= self.comment->**first**() **as context in** <ARTICLEQUALIFIER>
| self.comment->**first**() **as context in** <QUANTIFIEDQUALIFIER>
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.name
['containing' **foreach** c:Content **in** self.content **separator**(',') c **as context in** <CONTENTLABEL> **end** ';']

LITERALVALUELABEL ::= self.comment->**first**() **as context in** <ASSIGNMENTQUALIFIER>
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.name
['containing' **foreach** c:Content **in** self.content **separator**(',') c **as context in** <CONTENTLABEL> **end** ';']

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
the "GUC packet"
a GUC packet
several GUC packets
indicating value itsGnProtocolVersion "MIB parameter" ,
set to itsGnDefaultHopLimit MIB parameter
```

## 6.4.3 Content

### Concrete Graphical Notation

There is no shape associated with this element. Instead, it is represented as a label within the context of a 'StructuredTestObjective'.

### Formal Description

**context Content**

CONTENTLABEL ::= [**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.name

```
if self.value.oclIsUndefined() then
    ['containing' foreach c:Content in self.content separator(',') c as context in <CONTENTLABEL> end ';']
else
    self.value as context in <VALUE>
endif
```

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
a "GUC packet" containing
  BasicHeader containing
    "version field" indicating value "itsGnProtocolVersion MIB parameter" ,
    "RHL field" indicating value "itsGnDefaultHopLimit MIB parameter"
  ;
;
```

## 6.4.4 LiteralValueReference

### Concrete Graphical Notation

There is no shape associated with this element. Instead, it is represented as a label within the context of a 'StructuredTestObjective'.

### Formal Description

**context LiteralValueReference**

LITERALVALUEREFERENCEARGUMENTLABEL ::= 'the' 'value' 'of'
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.content.name

LITERALVALUEREFERENCELABEL ::= self.comment->**first**() **as context in** <REFERENCEQUALIFIER>
'the' 'value' 'of'
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.content.name

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
the value of itsGnDefaultHopLimit MIB parameter
corresponding to the value of itsGnDefaultHopLimit MIB parameter
derived from the value of itsGnDefaultHopLimit MIB parameter
```

## 6.4.5 ContentReference

### Concrete Graphical Notation

There is no shape associated with this element. Instead, it is represented as a label within the context of a 'StructuredTestObjective'.

### Formal Description

**context ContentReference**

CONTENTREFERENCEARGUMENTLABEL ::= 'the' 'value' 'contained' 'in'
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.content.name

CONTENTREFERENCELABEL ::= self.comment **->first**() **as context in** <REFERENCEQUALIFIER>
'the' 'value' 'contained' 'in'
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.content.name

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
the value contained in "RHL field"
corresponding to the value contained in "version field"
derived from the value contained in "BasicHeader"
```

## 6.4.6 DataReference

### Concrete Graphical Notation

There is no shape associated with this element. Instead, it is represented as a label within the context of a 'StructuredTestObjective'.

### Formal Description

**context DataReference**

DATAREFERENCEARGUMENTLABEL ::= self.comment->**first**() **as context in** <ARTICLEQUALIFIER>
| self.comment->**first**() **as context in** <QUANTIFIEDQUALIFIER>
'(predefined)'
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.content **as context in** <STATICDATAUSELABEL>

DATAREFERENCELABEL ::= [self.name]
self.comment->**first**() **as context in** <REFERENCEQUALIFIER>
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.content **as context in** <STATICDATAUSELABEL>

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
the (predefined) FullHeader
the (predefined) FullHeader containing
  RHLField indicating value itGnDefaultHopLimit
;
```

### 6.4.7 StaticDataUse

#### Concrete Graphical Notation

Inherited from ETSI ES 203 119-2 [2] for 'StaticDataUse's not contained in a 'StructuredTestObjective', overridden for 'StaticDataUse's directly or indirectly contained in a 'StructuredTestObjective'.

#### Formal Description

**context StaticDataUse**

STATICDATAUSELABEL ::= **if** self.**oclIsTypeOf**(DataInstanceUSe) **then** self **as context in** <DATAINSTANCEUSELABEL>
**else if** self.**oclIsTypeOf**(AnyValue) **then** self **as context in** <ANYVALUELABEL>
**else if** self.**oclIsTypeOf**(AnyValueOrOmitValue) **then** self **as context in** <ANYVALUEORMITLABEL>
**else if** self.**oclIsTypeOf**(OmitValue) **then** self **as context in** <OMITVALUELABEL>
**endif**

#### Constraints

There are no constraints specified.

#### Comments

No comments.

#### Example

```
FullHeader
any Header
any or no Header
no Header
```

### 6.4.8 AnyValue

#### Concrete Graphical Notation

Inherited from ETSI ES 203 119-2 [2] for 'AnyValue's not contained in a 'StructuredTestObjective', overridden for 'AnyValue's directly or indirectly contained in a 'StructuredTestObjective'.

#### Formal Description

**context AnyValue**

ANYVALUELABEL ::= 'any' self.dataType.name

#### Constraints

There are no constraints specified.

#### Comments

No comments.

#### Example

```
any Header
```

### 6.4.9 AnyValueOrOmit

#### Concrete Graphical Notation

Inherited from ETSI ES 203 119-2 [2] for 'AnyValueOrOmit's not contained in a 'StructuredTestObjective', overridden for 'AnyValueOrOmit's directly or indirectly contained in a 'StructuredTestObjective'.

#### Formal Description

**context AnyValueOrOmit**

ANYVALUEORMITLABEL ::= 'any' 'or' 'no' self.dataType.name

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
any or no Header
```

## 6.4.10 OmitValue

### Concrete Graphical Notation

Inherited from ETSI ES 203 119-2 [2] for 'OmitValue's not contained in a 'StructuredTestObjective', overridden for 'OmitValue's directly or indirectly contained in a 'StructuredTestObjective'.

### Formal Description

**context OmitValue**
OMITVALUELABEL ::= 'no' self.dataType.name

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
no Header
```

## 6.4.11 DataInstanceUse

### Concrete Graphical Notation

Inherited from ETSI ES 203 119-2 [2] for 'DataInstanceUse's not contained in a 'StructuredTestObjective', overridden for 'DataInstanceUse's directly or indirectly contained in a 'StructuredTestObjective'.

### Formal Description

**context DataInstanceUse**
DATAINSTANCEUSELABEL ::= self.dataInstance.name
['containing'
**foreach** a:ArgumentSpecification **in** self.argument **separator**(', ') c **as context in** <ARGUMENTSPECIFICATIONLABEL> **end**
';']

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
FullHeader
FullHeader containing
  RHLField indicating value itGnDefaultHopLimit
;
```

### 6.4.12 ArgumentSpecification

#### Concrete Graphical Notation

Inherited from ETSI ES 203 119-2 [2] for 'ArgumentSpecification's not contained in a 'StructuredTestObjective', overridden for 'ArgumentSpecification's directly or indirectly contained in a 'StructuredTestObjective'.

#### Formal Description

**context ArgumentSpecification**
ARGUMENTSPECIFICATIONLABEL ::= self.member.name
self.comment->**first**() **as context in** <ASSIGNMENTQUALIFIER>
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER> **end**]
self.dataUse **as context in** <STATICDATAUSELABEL>

#### Constraints

There are no constraints specified.

#### Comments

No comments.

#### Example

```
RHLField indicating value itGnDefaultHopLimit
RHLField indicating value itGnDefaultHopLimit containing
  VersionField indicating value baseVersion
;
```

## 6.5 Time

### 6.5.1 TimeLabel

#### Concrete Graphical Notation

Inherited from ETSI ES 203 119-2 [2] for 'TimeLabel's not contained in a 'StructuredTestObjective', overridden for 'TimeLabel's directly or indirectly contained in a 'StructuredTestObjective'.

#### Formal Description

**context TimeLabel**
TIMELABELLABEL ::= 'at' 'time' 'point' self.name

#### Constraints

There are no constraints specified.

#### Comments

No comments.

#### Example

```
at time point t
```

### 6.5.2 TimeConstraint

#### Concrete Graphical Notation

Inherited from ETSI ES 203 119-2 [2] for 'TimeConstraint's not contained in a 'StructuredTestObjective', overridden for ' TimeConstraint 's directly or indirectly contained in a 'StructuredTestObjective'.

### Formal Description

**context TimeConstraint**

TIMECONSTRAINTLABEL ::= [**foreach** c:Comment **in** self.comment **as context in** <QUALIFIER> **end**]
self.comment **as context in** <TIMECONSTRAINTQUALIFIER>
[**foreach** c:Comment **in** self.comment c **as context in** <QUALIFIER|COMMONWORDQUALIFIER|ARTICLEQUALIFIER> **end**]
self.timeConstraintExpression.dataInstance.name

### Constraints

There are no constraints specified.

### Comments

No comments.

### Example

```
30s after t
within 5s of t
during the 5s after t
```

# 7 Exchange Format Extensions

The exchange format for the extension is fully governed by the exchange format for TDL as specified in ETSI ES 203 119-3 [3]. No additional specification is provided.

# Annex A (informative): Textual Syntax

## A.0 Overview

This annex specifies a textual syntax for the additional concepts and the minimal set of required TDL concepts to facilitate the specification and representation of 'StructuredTestObjective's in pure text. The syntax for the constituents of the 'StructuredTestObjective's, such as 'InitialConditions', 'ExpectedBehaviour', and 'FinalConditions' is identical to the corresponding compartment specifications in clause 6.1. The complete BNF production rules are specified in annex B.

## A.1 A 3GPP Test Objective in Textual Syntax

This example describes one possible way to translate the test objectives in clause 7.1.3.1 from ETSI TS 136 523-1 [i.2] into the proposed textual syntax for the structured test objective specification with TDL, by mapping the concepts from the representation in the source document to the corresponding concepts for the structured test objective specification with TDL described in the present document. The example has been reformulated and interpolated where applicable to fit into the framework of the present document.

```
Package "3GPP, Clause 7.1.3.1" {
  //a possible specification of the test objectives from Clause 7.1.3.1 in [i.2]
  //some interpolation has been applied to fit into the overall framework and concrete syntax
  //of the present document

  Domain{
    entities:
    - UE
    - UE_A
    - UE_B
    - IMS_B
    - "HSS of IMS_A"
    - IUT
    - buffer
    ;
    events :
    - "in"
    - sends
    - receives
    - performs
    - send
    ;
  }

  Test Purpose {
    TP Id TP_7_1_3_1_1
    Test objective ""
    Reference "3GPP TS 36.321 clause 5.3.1"
    PICS Selection
    Initial conditions
    with {
      the UE entity "in" the "E-UTRA RRC_CONNECTED state"
    }
    Expected behaviour
    ensure that {
      when {
        the UE entity receives a "downlink assignment on the PDCCH for the UE's C-RNTI" and
        the UE entity receives a "data in the associated subframe" and
        the UE entity performs a HARQ operation
      }
      then {
        the UE entity sends a "HARQ feedback on the HARQ process"
      }
    }
  }

  Test Purpose {
    TP Id TP_7_1_3_1_2
    Test objective ""
    Reference "3GPP TS 36.321 clause 5.3.1"
```

```
    PICS Selection
    Initial conditions
    with {
      the UE entity "in" the "E-UTRA RRC_CONNECTED state"
    }
    Expected behaviour
    ensure that {
      when {
        the UE entity receives a "downlink assignment on the PDCCH unknown by the UE" and
        the UE entity receives a "data in the associated subframe"
      }
      then {
        the UE entity does not send any "HARQ feedback on the HARQ process"
      }
    }
  }

}
```

## A.2 An IMS Test Objective in Textual Syntax

This example describes one possible way to translate the test objective clause 4.5.1 from ETSI TS 186 011-2 [i.3] into the proposed textual syntax for the structured test objective specification with TDL, by mapping the concepts from the representation in the source document to the corresponding concepts for the structured test objective specification with TDL described in the present document. The example has been reformulated and interpolated where applicable to fit into the framework of the present document.

```
Package "3GPP_7_1_3_1" {
  //a possible specification of the test objectives from Clause 7.1.3.1 in [i.2]
  //some interpolation has been applied to fit into the overall framework and concrete syntax
  //of the present document

  Domain{
    entities:
    - UE
    - UE_A
    - UE_B
    - IMS_B
    - "HSS of IMS_A"
    - IUT
    - buffer
    ;
    events :
    - "in"
    - sends
    - receives
    - performs
    - send
    ;
  }

  Test Purpose {
    TP Id TP_7_1_3_1_1
    Test objective ""
    Reference "3GPP TS 36.321 clause 5.3.1"
    PICS Selection
    Initial conditions
    with {
      the UE entity "in" the "E-UTRA RRC_CONNECTED state"
    }
    Expected behaviour
    ensure that {
      when {
        the UE entity receives a "downlink assignment on the PDCCH for the UE's C-RNTI" and
        the UE entity receives a "data in the associated subframe" and
        the UE entity performs a HARQ operation
      }
      then {
        the UE entity sends a "HARQ feedback on the HARQ process"
      }
    }
  }
```

```
  Test Purpose {
    TP Id TP_7_1_3_1_2
    Test objective ""
    Reference "3GPP TS 36.321 clause 5.3.1"
    PICS Selection
    Initial conditions
    with {
      the UE entity "in" the "E-UTRA RRC_CONNECTED state"
    }
    Expected behaviour
    ensure that {
      when {
        the UE entity receives a "downlink assignment on the PDCCH unknown by the UE" and
        the UE entity receives a "data in the associated subframe"
      }
      then {
        the UE entity does not send any "HARQ feedback on the HARQ process"
      }
    }
  }

}
```

# Annex B (informative): Textual Syntax BNF Production Rules

## B.0 Overview

This annex describes the grammar for the representation of structured test objectives in pure text. It covers the additional concepts and the minimal set of required TDL concepts to facilitate the specification and representation of 'StructuredTestObjective's.

## B.1 Conventions

The notations is based on the Extended Backus-Naur Form (EBNF) notation. The EBNF representation may be used either as a concrete syntax reference for Structured Test Objective Specification with TDL for end users or as input to a parser generator tool. Table B.1 defines the syntactic conventions that are to be applied when reading the EBNF rules.

**Table B.1: Syntax definition conventions used**

| | |
|---|---|
| ::= | is defined to be |
| abc | the non-terminal symbol abc |
| abc xyz | abc followed by xyz |
| abc \| xyz | alternative (abc or xyz) |
| [abc] | 0 or 1 instance of abc |
| {abc}+ | 1 or more instances of abc |
| {abc} | 0 or more instances of abc |
| 'a'-'z' | all characters from a to z |
| (...) | denotes a textual grouping |
| 'abc' | the terminal symbol abc |
| ; | production terminator |
| \ | the escape character |

## B.2 Production Rules

```
Package                         ::=  'Package' Identifier '{'
                                       { ElementImport }
                                       [ 'Domain' '{'
                                         [ 'pics' ':' { PICS }+ ';' ]
                                         [ 'entity' 'types' ':' { EntityType }+ ';' ]
                                         [ 'entities' ':' { Entity }+ ';' ]
                                         [ 'event' 'types' ':' { EventType }+ ';' ]
                                         [ 'events' ':' { Event }+ ';' ] '}' ]
                                       [ 'Data' '{'
                                         { StructuredDataType }
                                         { StructuredDataInstance } '}' ]
                                       { StructuredTestObjective }
                                       { Group } '}' ;
ElementImport                   ::=  'import'
                                       ( 'all' | ( Identifier | { ',' Identifier } ) )
                                       'from' Identifier ';' ;
Group                           ::=  'Group' Identifier '{'
                                       { StructuredTestObjective }
                                       { Group } '}' ;
PICS                            ::=  '-' Identifier [ '(' Qualifier ')' ] ;
PICSReference                   ::=  [ AndOrQualifier ] Identifier ;
EntityType                      ::=  '-' Identifier ;
Entity                          ::=  '-' Identifier
                                       [ '(' Annotation { ',' Annotation } ')' ] ;
EventType                       ::=  '-' Identifier ;
Annotation                      ::=  Identifier ;
Event                           ::=  '-' Identifier
                                       [ '(' Annotation { ',' Annotation } ')' ] ;
StructuredTestObjective         ::=  'Test Purpose' '{'
                                       'TP Id' Identifier
                                       'Test objective' Identifier
                                       'Reference' [ Identifier { ',' Identifier } ]
```

```
                                  'PICS Selection' { PICSReference }
                                  InitialConditions
                                  ExpectedBehaviour
                                  FinalConditions '}' ;
InitialConditions            ::=  'Initial conditions'
                                  'with' '{' EventSequence '}' ;
ExpectedBehaviour            ::=  FullExpectedBehaviour | PartialExpectedBehaviour ;
FullExpectedBehaviour        ::=  'Expected behaviour'
                                  'ensure that' '{'
                                    'when' '{' EventSequence '}'
                                    'then' '{' EventSequence '}'
                                  '}' ;
PartialExpectedBehaviour     ::=  'Expected behaviour'
                                  'ensure that' '{' EventSequence '}' ;
FinalConditions              ::=  'Final conditions'
                                  'with' '{' EventSequence '}' ;
EventSequence                ::=  FirstEventOccurrence { EventOccurrence } ;
FirstEventOccurrence         ::=  [ ( TimeLabel
                                      ( ( ',' TimeConstraint )
                                        | ':'
                                      )
                                    )
                                    | TimeConstraint ]
                                  EntityReference
                                  EventReference
                                  Argument
                                  [ OppositeEntityReference { ',' OppositeEntityReference }
                                  ]
                                  { Note } ;
Note                         ::=  '(' 'Note' NumberAsIdentifier ':' Identifier ')' ;
EventOccurrence              ::=  AndOrQualifier
                                  [ ( TimeLabel
                                      ( ( ',' TimeConstraint )
                                        | ':'
                                      )
                                    )
                                    | TimeConstraint ]
                                  EntityReference
                                  EventReference
                                  Argument
                                  [ OppositeEntityReference { ',' OppositeEntityReference }
                                  ]
                                  { Note } ;
TimeLabel                    ::=  'at' 'time' 'point' Identifier ;
TimeConstraint               ::=  { Qualifier }
                                  TimeConstraintQualifier
                                  { Qualifier | CommonWordQualifier | ArticleQualifier }
                                  TimeConstraintExpression ':' ;
TimeConstraintExpression     ::=  Identifier ;
TimeConstraintQualifier      ::=  ( 'before' | 'after' | 'during' | 'within' ) ;
EntityReference              ::=  ArticleQualifier
                                  { Qualifier }
                                  Identifier
                                  'entity' ;
OppositeEntityReference      ::=  DirectionQualifier
                                  ArticleQualifier
                                  { Qualifier }
                                  Identifier ;
EventReference               ::=  { Qualifier | CommonWordQualifier } Identifier ;
Argument                     ::=  LiteralValueAsArgument
                                   | DataReferenceAsArgument
                                   | ContentReferenceAsArgument
                                   | LiteralValueReferenceArgument ;
Value                        ::=  LiteralValue
                                   | DataReference
                                   | ContentReference
                                   | LiteralValueReference ;
LiteralValueAsArgument       ::=  ( ArticleQualifier | QuantifiedQualifier )
                                  { Qualifier }
                                  ( Identifier | NumberAsIdentifier )
                                  [ 'containing' DataContent { ',' DataContent } ';' ] ;
LiteralValue                 ::=  AssignmentQualifier
                                  { Qualifier }
                                  ( Identifier | NumberAsIdentifier )
                                  [ 'containing' DataContent { ',' DataContent } ';' ] ;
DataContent                  ::=  { Qualifier }
                                  ( Identifier | NumberAsIdentifier )
                                  [ ( 'containing' | DataContent | { ',' DataContent } | ';'
```

```
                                    )
                                    | Value ] ;
Identifier                  ::=   STRING | ID ;
Qualifier                   ::=   Identifier | NumberAsIdentifier ;
CommonWordQualifier         ::=   'before'
                                   | 'after'
                                   | 'from'
                                   | 'to'
                                   | 'of' ;
ArticleQualifier            ::=   'a'
                                   | 'an'
                                   | 'the' ;
QuantifiedQualifier         ::=   'all'
                                   | 'any'
                                   | 'few'
                                   | 'multiple'
                                   | 'no'
                                   | 'only'
                                   | 'several'
                                   | 'some' ;
AssignmentQualifier         ::=   'indicating value' | 'set to' ;
AndOrQualifier              ::=   'and' | 'or' ;
DirectionQualifier          ::=   'by'
                                   | 'in'
                                   | 'into'
                                   | 'for'
                                   | 'from'
                                   | 'to' ;
ReferenceQualifier          ::=   'corresponding to'
                                   | 'derived from'
                                   | 'carrying'
                                   | 'contained in'
                                   | 'associated with' ;
DataInstanceUse             ::=   ( Identifier | NumberAsIdentifier )
                                  [ 'containing'
                                     ArgumentSpecification { ',' ArgumentSpecification } ';'
                                  ] ;
StaticDataUse               ::=   DataInstanceUse
                                   | AnyValue
                                   | AnyValueOrOmit
                                   | OmitValue ;
AnyValue                    ::=   'any' Identifier ;
AnyValueOrOmit              ::=   'any' 'or' 'no' Identifier ;
OmitValue                   ::=   'no' Identifier ;
ArgumentSpecification       ::=   Identifier
                                  AssignmentQualifier
                                  { Qualifier }
                                  StaticDataUse ;
ContentReference            ::=   ReferenceQualifier
                                  'the' 'value' 'contained in'
                                  { Qualifier }
                                  Identifier ;
LiteralValueReference       ::=   ReferenceQualifier
                                  'the' 'value' 'of'
                                  { Qualifier }
                                  Identifier ;
ContentReferenceAsArgument  ::=   'the' 'value' 'contained in' { Qualifier } Identifier ;
LiteralValueReferenceArgument ::= 'the' 'value' 'of' { Qualifier } Identifier ;
DataReference               ::=   Identifier
                                  ReferenceQualifier
                                  { Qualifier }
                                  StaticDataUse ;
DataReferenceAsArgument     ::=   ( ArticleQualifier | QuantifiedQualifier )
                                  '(predefined)'
                                  { Qualifier }
                                  StaticDataUse ;
NumberAsIdentifier          ::=   ['-'] INT [ '.' INT ] ;
StructuredDataType          ::=   'type' Identifier [ 'with' Member { ',' Member } ] ';' ;
Member                      ::=   Optional Identifier 'of' 'type' Identifier ;
Optional                    ::=   'optional' ;
StructuredDataInstance      ::=   Identifier
                                  ( Identifier | NumberAsIdentifier )
                                  [ 'containing'
                                    MemberAssignment { ',' MemberAssignment } ] ';' ;
MemberAssignment            ::=   Identifier AssignmentQualifier StaticDataUse ;
ID                          ::=   ( ['^'] ( 'a'-'z' | 'A'-'Z' | '_' ) { 'a'-'z' | 'A'-'Z' |
                                  '_' | '0'-'9' | '/' } ) ;
INT                         ::=   '0'-'9' ;
```

```
SQ                              ::=   '"' ;
DQ                              ::=   "'" ;
STRING                          ::=   ( ( DQ | { ( '\\' | ( 'b' | 't' | 'n' | 'f' | 'r' | 'u' |
                                      '"' | "'" | '\\' ) ) | ( '\\' | DQ ) } | DQ ) | ( SQ | { (
                                      '\\' | ( 'b' | 't' | 'n' | 'f' | 'r' | 'u' | '"' | "'" |
                                      '\\' ) ) | ( '\\' | SQ ) } | SQ ) ) ;
ML_COMMENT                      ::=   ( '/*' '*/') ;
SL_COMMENT                      ::=   ( '//' ( '\\n' | '\\r' ) [ ['\\r'] '\\n' ] ) ;
WS                              ::=   { ' '
                                       | '\\t'
                                       | '\\r'
                                       | '\\n' }+ ;
```

# History

| Document history | | |
|---|---|---|
| V1.1.0 | April 2015 | Membership Approval Procedure MV 20150619: 2015-04-20 to 2015-06-19 |
| V1.1.1 | June 2015 | Publication |
| | | |
| | | |
| | | |