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

Intellectual Property Rights ................................................................. 5
Foreword ................................................................................................ 5
Modal verbs terminology ................................................................. 5
Introduction ......................................................................................... 5
  1 Scope ............................................................................................ 6
  2 References .................................................................................... 6
    2.1 Normative references .............................................................. 6
    2.2 Informative references ........................................................... 6
  3 Definitions and abbreviations ...................................................... 7
    3.1 Definitions .............................................................................. 7
    3.2 Abbreviations ........................................................................ 7
  4 Basic principles ........................................................................... 7
    4.1 Structured Test Objective Specification ................................. 7
    4.2 Document Structure .............................................................. 8
    4.3 Notational Conventions ......................................................... 8
    4.4 OCL Constraints Requirements ............................................ 8
    4.5 Conformance .......................................................................... 9
  5 Meta-Model Extensions .............................................................. 9
    5.1 Overview ............................................................................... 9
    5.2 Foundation Abstract Syntax and Classifier Description .......... 9
      5.2.1 Entity ............................................................................ 9
      5.2.2 Event ........................................................................ 10
      5.2.3 PICS .......................................................................... 10
    5.3 Test Objective Abstract Syntax and Classifier Description ...... 11
      5.3.1 StructuredTestObjective ............................................... 11
      5.3.2 PICSReference ............................................................ 12
      5.3.3 InitialConditions .......................................................... 12
      5.3.4 ExpectedBehaviour ..................................................... 13
      5.3.5 FinalConditions .......................................................... 13
    5.4 Events Abstract Syntax and Classifier Description ............... 14
      5.4.1 EventSequence ............................................................ 14
      5.4.2 RepeatedEventSequence .............................................. 15
      5.4.3 EventOccurrence .......................................................... 15
      5.4.4 EventOccurrenceSpecification .................................... 16
      5.4.5 EntityReference .......................................................... 17
      5.4.6 EventReference ............................................................ 17
    5.5 Data Abstract Syntax and Classifier Description .................. 18
      5.5.1 Value ........................................................................... 18
      5.5.2 LiteralValue ................................................................. 18
      5.5.3 Content ....................................................................... 19
      5.5.4 LiteralValueReference ............................................... 19
      5.5.5 ContentReference .......................................................... 20
      5.5.6 DataReference .............................................................. 20
    5.6 Event Templates Abstract Syntax and Classifier Description ... 21
      5.6.1 EventSpecificationTemplate ........................................ 21
      5.6.2 EventTemplateOccurrence ......................................... 22
      5.6.3 EntityBinding ............................................................... 23
  6 Graphical Syntax Extensions ...................................................... 23
    6.1 Foundation .......................................................................... 23
    6.1.1 Entity ........................................................................... 23
    6.1.2 Event ........................................................................... 24
    6.1.3 PICS ............................................................................. 24
Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables 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 (https://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.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

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. Full details of the entire series can be found in part 1 [1].

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 graphical syntactic notation for the added constructs. In addition textual syntax examples of the TDL Structured Test Objectives extensions as well as BNF rules for a textual syntax for TDL with the Structured Test Objectives extensions are provided. 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.

NOTE: OMG®, UML®, OCL™ and UTP™ are the trademarks of OMG (Object Management Group). This information is given for the convenience of users of the present document and does not constitute an endorsement by ETSI of the products named.

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 https://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.


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)".
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:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNF</td>
<td>Backus-Naur Form</td>
</tr>
<tr>
<td>EBNF</td>
<td>Extended Backus-Naur Form</td>
</tr>
<tr>
<td>IMS</td>
<td>IP Multimedia Subsystem</td>
</tr>
<tr>
<td>IUT</td>
<td>Implementation Under Test</td>
</tr>
<tr>
<td>OCL</td>
<td>Object Constraint Language</td>
</tr>
<tr>
<td>PICS</td>
<td>Protocol Implementation Conformance Statement</td>
</tr>
<tr>
<td>SUT</td>
<td>System Under Test</td>
</tr>
<tr>
<td>TDL</td>
<td>Test Description Language</td>
</tr>
<tr>
<td>TPLan</td>
<td>Test Purpose Notation</td>
</tr>
</tbody>
</table>

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 may 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 shall be defined in advance as part of a domain description which may 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 lightweight approach, describe data inline without the need to define all data types and instances in advance. Pre-defined data and inline data may be integrated to a certain degree. Inline data may refer to pre-defined data, but pre-defined data shall not refer to inline data.

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 shall 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 OCL Constraints Requirements

In addition to the operations provided by the standard library of OCL and in ETSI ES 203 119-1 [1], the formalized constraints for the present document rely on the following additional operations that serve as reusable shortcuts and shall be provided by TDL semantical analyser implementations for the interpretation of the OCL constraints:

- OclAny getTestObjective (): StructuredTestObjective - applicable on any TDL 'Element', returns the 'StructuredTestObjective' that contains the construct directly or indirectly.
- OclAny contains (object : OclAny): Boolean - applicable on any TDL 'Element', accepts a TDL 'Element' as parameter 'object', returns 'true' if the 'Element' contains the 'object' and 'false' otherwise.
- StructuredTestObjective indexOf (object : OclAny): Integer - applicable on a 'StructuredTestObjective', accepts a TDL 'Element' as parameter 'object', returns the position of the 'object' within the flattened list of all 'Element's directly and indirectly contained within the 'StructuredTestObjective'. The list is flattened according to a depth-first approach.
4.5 Conformance

For an implementation claiming to conform to this extension of the 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 in ETSI ES 203 119-1 [1]. The electronic attachment from annex A in ETSI ES 203 119-1 [1] may 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](image)

**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 to 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

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'. A 'StructuredTestObjective' may optionally reference a 'TestConfiguration' on which the 'StructuredTestObjective' shall be realized. The referenced 'TestConfiguration' provides descriptive information regarding the intended setup for the 'StructuredTestObjective'.

Generalizations

- TestObjective

Properties

- picsReference : PICSReference [*] {ordered}
  An ordered set of 'PICSReferences' to 'PICS'.

- configuration : TestConfiguration [0..1]
  A reference to a 'TestConfiguration'.

Figure 5.2: Structured Test Objective Concepts
• 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'.
  inv: MultiplePICS:
  self.container(). picsReference->size() < 2
  or self.container(). picsReference->forall(p |
  self.container(). picsReference->at(0) = p
  or (not p.comment->isEmpty() 
  and (p.comment->first(). body = 'and' 
  or p.comment->first(). body = 'or')))}

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 RepeatedEventSequence

Semantics

'RepeatedEventSequence' is an 'EventSequence' optionally specifying a number of repetitions or a repetition interval. In case neither the number of repetitions nor the repetition interval is specified, the 'EventOccurrences' defined in the 'RepeatedEventSequence' may occur indefinite number of times with arbitrary frequency. If the 'repetitions' property is defined, the associated 'EventOccurrence's are executed the specified number of times. If the 'interval' property is defined, the associated 'EventOccurrence' are executed repeatedly with the specified time interval.

Generalizations

- EventSequence

Properties

- repetitions : Value [0..1]
  A 'Value' expression that specifies the number of repetitions the 'EventOccurrence's shall be executed.
- interval: Value [0..1]
  A 'Value' expression that specifies the interval between each repeated execution of the 'EventOccurrence's.

Constraints

- Either 'repetitions', or 'interval' or neither shall be specified
  At most one of the optional properties 'repetitions' or 'interval' shall be defined.
  inv: RepetitionOrInterval:
  self.repetitions.oclIsUndefined() or self.interval.oclIsUndefined()
- The 'repetitions' 'Value' shall be countable and positive
  The expression assigned to the 'repetitions' property shall evaluate to a positive and countable 'Value'.
  inv: RepetitionCount:
  This constraint cannot be expressed in OCL.
- The 'interval' 'Value' shall be countable and positive
  The expression assigned to the 'repetitions' property shall evaluate to a positive and countable 'Value'
  inv: RepetitionInterval:
  This constraint cannot be expressed in OCL.

5.4.3 EventOccurrence

Semantics

An 'EventOccurrence' is an 'Element' describing an occurrence of an 'Event' within an 'EventSequence'. 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 a 'SimpleEventSequence' 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

- `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'.

  \[
  \text{inv: MultipleEventOccurrences:}
  \]

  \[
  \text{self.container().oclIsTypeOf(EventSpecificationTemplate)}
  \text{or self.container().events->size() < 2}
  \text{or self.container().events->forAll(o |}
  \text{self.container().events->at(0) = o}
  \text{or (not o.comment->isEmpty()}
  \text{and (o.comment->first().body = 'and'
  \text{or o.comment->first().body = 'or'))})
  \]

5.4.4 EventOccurrenceSpecification

Semantics

An 'EventOccurrenceSpecification' 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 'EventOccurrenceSpecification'. It also includes a 'Value' as an argument describing the details of the 'EventOccurrenceSpecification' such as the data being sent or received, or a state an involved 'Entity' is in.

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'.

ETSÍ
Constraints
There are no constraints specified.

5.4.5 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'. Alternatively, an 'EntityReference' may be used to reference a 'ComponentInstance' of a 'TestConfiguration' instead of an 'Entity'.

Generalizations
- Element

Properties
- entity : Entity [0..1]
  The referenced 'Entity'.
- component : ComponentInstance [0..1]
  The referenced 'ComponentInstance'.

Constraints
- An 'Entity' or a 'ComponentInstance' shall be referenced.
  There shall be a reference to an 'Entity' or a 'ComponentInstance' but not both.
  inv: EntityOrComponentInstance:
    (not self.entity.oclIsUndefined() and self.component.oclIsUndefined())
  or (self.entity.oclIsUndefined() and not self.component.oclIsUndefined())

5.4.6 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

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.
  inv: ContentOrValue:
    self.content->isEmpty() or self.value.oclIsUndefined()

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'.

ETSI
Constraints

- **Referenced 'LiteralValue' visibility**
  Only 'LiteralValue's defined within previous 'EventOccurrence's of the containing 'StructuredTestObjective' may be referenced.
  \[\text{inv: VisibleValue:}\]
  
  \[\text{self.getTestObjective().contains(self.content)}\]
  
  \[\text{and self.getTestObjective().indexOf(self.content) < self.getTestObjective().indexOf(self)}\]

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.
  \[\text{inv: VisibleContent:}\]
  
  \[\text{self.getTestObjective().contains(self.content)}\]
  
  \[\text{and self.getTestObjective().indexOf(self.content) < self.getTestObjective().indexOf(self)}\]

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'.

ETSIT
Constraints

- **'DataUse' restrictions within 'DataReference'**
  Only 'StaticDataUse' may be used directly or indirectly in 'ParameterBinding's of the 'StaticDataUse' within a 'DataReference'.
  inv: DataReferenceContents:
  
  ```
  self.content.oclIsTypeOf(StaticDataUse)
  and self.content.argument->forAll(a | a.dataUse.oclIsKindOf(StaticDataUse))
  and self.content.argument->closure(a | a.dataUse.argument)->forAll(a | a.dataUse.oclIsKindOf(StaticDataUse))
  ```

- **No 'reduction' within 'DataReference'**
  The 'reduction' property of 'StaticDataUse' inherited from 'DataUse' shall not be used within a 'DataReference'.
  inv: DataReferenceReduction:
  
  ```
  self.content.reduction->isEmpty()
  ```

5.6 Event Templates Abstract Syntax and Classifier Description

5.6.1 EventSpecificationTemplate

Semantics

'EventSpecificationTemplate' is a 'PackageableElement' containing a single reusable 'EventOccurrenceSpecification'. An 'EventSpecificationTemplate' may be referenced within an 'EventSequence' by means of an 'EventTemplateOccurrence'.

Generalizations

- PackageableElement

Properties

  A reusable 'EventOccurrenceSpecification'.

Figure 5.5: Event Templates Concepts
5.6.2 EventTemplateOccurrence

**Semantics**

An 'EventTemplateOccurrence' is an 'EventOccurrence' referring to a reusable 'EventSpecificationTemplate' that defines a concrete occurrence of the referenced 'EventSpecificationTemplate' within an 'EventSequence'. Optional 'EntityBinding's may be specified to override some or all of the 'EntityReference' specified in 'EventOccurrenceSpecification' of the referenced 'EventTemplateSpecification' with new 'EntityReference's. Optional 'Value' specification may be specified to overriding the 'Value' specified as argument in 'EventOccurrenceSpecification' of the referenced 'EventTemplateSpecification' with a new 'Value'.

**Generalizations**

- EventOccurrence

**Properties**

- eventTemplate : EventSpecificationTemplate [1]
  The referenced 'EventSpecificationTemplate'.
- entityBinding : EntityBinding [0..*]
  Optional 'EntityBinding's for substituting the 'EntityReference' specified in 'EventOccurrenceSpecification' of the referenced 'EventTemplateSpecification' with new 'EntityReference's.
- occurrenceArgument : Value [0..1]
  Optional 'Value' specification overriding the 'Value' specified as argument in 'EventOccurrenceSpecification' of the referenced 'EventTemplateSpecification'.

**Constraints**

- 'EntityReference' of referenced 'EventSpecificationTemplate'
  If 'EntityBinding's are provided, the 'Entity's or 'ComponentInstance's referenced in the 'templateEntity' properties shall also be referenced by one of the 'EntityReferences' in the 'EventOccurrenceSpecification' of the 'EventSpecificationTemplate' referenced in the 'EventTemplateOccurrence'.
  inv: EntityTemplateOccurrenceConsistency:
    
    self.entityBinding->forAll(b |
      (not b.templateEntity.entity.oclIsUndefined())
      and (b.templateEntity.entity =
        self.eventTemplate.eventSpecification.entityReference.entity))
    or (not b.templateEntity.component.oclIsUndefined())
    and (b.templateEntity.component =
        self.eventTemplate.eventSpecification.entityReference.component)
    or self.eventTemplate.eventSpecification.oppositeEntityReference->exists(e |
      (not b.templateEntity.entity.oclIsUndefined())
      and (e.entity = b.templateEntity.entity))
    or (not b.templateEntity.component.oclIsUndefined())
    and (e.component = b.templateEntity.component())))

Constraints

There are no constraints specified.
5.6.3 EntityBinding

Semantics
An 'EntityBinding' is an 'Element' used for substituting the 'EntityReference' specified in 'EventOccurrenceSpecification' of a 'EventTemplateSpecification' referenced within an 'EventTemplateOccurrence' with new 'EntityReference's.

Generalizations
- Element

Properties
- templateEntity : EntityReference [1]
  An 'EntityReference' describing the 'Entity' referenced in the 'EventOccurrenceSpecification' of the 'EventSpecificationTemplate'.
- occurrenceEntity : EntityReference [1]
  An 'EntityReference' describing the 'Entity' that shall replace the 'EntityReference' referenced in the 'EventOccurrenceSpecification' of the 'EventSpecificationTemplate' in the 'EventTemplateOccurrence'.

Constraints
There are no constraints specified.

6 Graphical Syntax Extensions

6.1 Foundation

6.1.1 Entity

Concrete Graphical Notation

![Entity Notation](image)

Formal Description

context Entity
  ENTITYLABEL := self.name

Constraints
There are no constraints specified.

Comments
No comments.

Example

<table>
<thead>
<tr>
<th>Entity</th>
<th>Entity</th>
<th>Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>IUT</td>
<td>Tester</td>
<td>buffer</td>
</tr>
</tbody>
</table>
6.1.2 Event

Concrete Graphical Notation

```
Event
EVENTLABEL
```

Formal Description

```
context Event
EVENTLABEL := self.name
```

Constraints

There are no constraints specified.

Comments

No comments.

Example

```
Event
  sends
Event
  receives
Event
  is
```

6.1.3 PICS

Concrete Graphical Notation

```
PICS
PICSLABEL
```

Formal Description

```
context PICS
PICSLABEL := self.name
```

Constraints

There are no constraints specified.

Comments

No comments.

Example

```
PICS
  PICS_F1
PICS
  PICS_F2
PICS
  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

class Comment

Qualifier := self.body

AND Qualifier := if self.body = 'and'

or self.body = 'or'

then

self.body

endif

ARTICLE Qualifier := if self.body = 'a'

or self.body = 'an'

or self.body = 'the'

then

self.body

endif

ASSIGNMENT Qualifier := if self.body = 'indicating value'

or self.body = 'set to'

then

self.body

endif

COMMON WORD Qualifier := if self.body = 'after'

or self.body = 'before'

or self.body = 'from'

or self.body = 'of'

or self.body = 'to'

then

self.body

endif

DIRECTION Qualifier := 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

QUANTIFIED Qualifier := 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":"sel f.name":"self.body""'

Constraints

- **Default comment label**
  The Qualified 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

<table>
<thead>
<tr>
<th>TP Id</th>
<th>TestObjectiveNAMELABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Objective</td>
<td>DESCRIPTIONLABEL</td>
</tr>
<tr>
<td>Reference</td>
<td>URIofObjectiveLABEL</td>
</tr>
<tr>
<td>Config Id</td>
<td>&lt;CONFIGLABEL&gt;</td>
</tr>
<tr>
<td>PICS Selection</td>
<td>&lt;PICSSELECTIONLABEL&gt;</td>
</tr>
<tr>
<td>Initial Conditions</td>
<td></td>
</tr>
<tr>
<td>Expected Behaviour</td>
<td></td>
</tr>
<tr>
<td>Final Conditions</td>
<td></td>
</tr>
</tbody>
</table>

Formal Description

```
context StructuredTestObjective

TESTOBJECTIVENAMELABEL ::= self.name
DESCRIPTIONLABEL ::= self.description
URIofObjectiveLABEL ::= self.objectiveURI->newline
CONFIGLABEL ::= self.configuration.name
PICSSELECTIONLABEL ::= foreach p:PICSReference in self.picsReferences p as context in <PICSRÉFERENCELABEL>

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, URIOBJECTIVELABEL, 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

<table>
<thead>
<tr>
<th>TP Id</th>
<th>TP/GONW/FDV/BAH/BV/01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Objective</td>
<td>Check defined values of default Gn parameters in the basic header</td>
</tr>
<tr>
<td>Reference</td>
<td></td>
</tr>
<tr>
<td>Config Id</td>
<td></td>
</tr>
<tr>
<td>PICS Selection</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>with {</td>
</tr>
<tr>
<td>the IUT entity being &quot;in&quot; the initial state</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expected Behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>ensure that {</td>
</tr>
<tr>
<td>when {</td>
</tr>
<tr>
<td>the IUT entity is requested to send a &quot;GUC packet&quot;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>then {</td>
</tr>
<tr>
<td>the IUT entity sends a &quot;GUC packet&quot; containing</td>
</tr>
<tr>
<td>BasicHeader containing</td>
</tr>
<tr>
<td>&quot;version field&quot; indicating value &quot;itsGnProtocolVersion MIB parameter&quot;,</td>
</tr>
</tbody>
</table>
|       "RHL field" indicating value "itsGnDefaultHopLimit MIB parameter"
| ;                       |
| }                      |

| 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

EVENTSEQUENCENAME ::=
  if selfoclIsTypeOf(RepeatedEventSequence) then self as context in <REPEATEDEVENTSEQUENCENAME>
  else if selfoclIsTypeOf(EventSequence) then self as context in <SIMPLEEVENTSEQUENCENAME>
  end

SIMPLEEVENTSEQUENCENAME ::=
  foreach e:EventOccurrence in self.events newline e as context in <EVENTOCCURRENCENAME>
```
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
repeat 2 times { the UE entity sends a "HARQ feedback on the HARQ process" }

6.3.2 RepeatedEventSequence

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 RepeatedEventOccurrence
REPEATEDEVENTSEQUENCELABEL ::= 'repeat'
   if self.intervalIsUndefined() then
      self.repetitions as context in <EVENTARGUMENTLABEL> 'times'
   else
      'every' self.interval as context in <EVENTARGUMENTLABEL>
   endif
   foreach e:EventOccurrence in self.events newline() as context in <EVENTOCCURRENCELABEL> end

Constraints
There are no constraints specified.

Comments
No comments.

Example
repeat 2 times {
   the IUT entity having received a "Beacon information" from the ItsNodeB entity and
   the IUT entity having received any message from the ItsNodeD entity
}
repeat every CBF_MAX {
   the IUT entity saves the "GBC packet" into the CBF buffer entity and
   the IUT entity starts a "contention timer" containing
duration set to CBF_MAX
   ;
   and
   the IUT entity broadcasts the received "GBC packet"
}
6.3.3 EventOccurrence

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

Formal Description

```
context EventOccurrence
EventOccurrenceLabel ::= 
  if selfoclIsTypeOf(EventOccurrenceSpecification) then self as context in <EventOccurrenceSpecificationLabel>
  else if selfoclIsTypeOf(EventTemplateOccurrence) then self as context in <EventTemplateOccurrenceLabel>
  end
```

Constraints
There are no constraints specified.

Comments
No comments.

Example
Not available.

6.3.4 EventOccurrenceSpecification

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 EventOccurrenceSpecification
EventOccurrenceSpecificationLabel ::= [self:comment->first] as context in <AnnotationQualifiers>
  if self:timeLabeloclIsUndefined then
    if not self:timeConstraintoclIsUndefined then
      self:timeConstraint as context in <TimeConstraintLabel>
    endif
  else
    self:timeLabel as context in <TimeLabel>
    if self:timeConstraintoclIsUndefined then
      
    else
      
    endif
  endif
  [self:eventReference 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 <OppositeEntityReferenceLabel> end]
  [foreach c:Comment in self:comment separator'|' c as context in <NoteLabel> end]
```

Constraints
There are no constraints specified.
6.3.5 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]
    if self.component.oclIsUndefined() then
        self.entity.name "entity"
    else
        self.component.name "component"
    endif

OppositeEntityLabel ::= self.comment->at() as context in <DIRECTIONQUALIFIER>
    self.comment->at() as context in <ARTICLEQUALIFIER>
    [foreach c:Comment in self.comment c as context in <QUALIFIER>-end]
    if self.component.oclIsUndefined() then
        self.entity.name "entity"
    else
        self.component.name "component"
    endif
```

Constraints
There are no constraints specified.

Comments
No comments.

Example

```
the IUT entity having received a "Beacon information" from the ItsNodeB entity
(Note 1: "Beacon information may be incomplete")
(.) at time point t1: the IUT entity receives a "message"
(.) at time point t2, (1) 3s after t1: the IUT entity sends an invitation to the ItsNodeD entity
(1) 5s after t1: the IUT entity receives a confirmation from the ItsNodeD entity
```

6.3.6 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'.

Example

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

context EventReference

EVENTREFERENCELABEL ::= [for each 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 selfoclIsTypeOf(DataReference) then self as context in \<DATAREFERENCEARGUMENTLABEL\>
else if selfoclIsTypeOf(LiteralValue) then self as context in \<LITERALVALUEARGUMENTLABEL\>
else if selfoclIsTypeOf(LiteralValueReference) then self as context in \<LITERALVALUEREFERENCEARGUMENTLABEL\>
else if selfoclIsTypeOf(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

\[ \text{literalValue} := \text{self.comment}\to\text{first()} \text{ as context in } \text{<ARTICLEQUALIFIER>}
\]

if not self.dataType.oclIsUndefined() then

\[ '\text{typed}' \]
endif

\[ \text{foreach } \text{cComment } \text{in self.comment } \text{as context in } \text{<QUALIFIER> end} \]

if self.dataType.oclIsUndefined() then

self.name

[\text{containing}'\text{foreach }\text{cContent }\text{in self.content separator[',']} \text{ as context in } \text{<CONTENTLABEL> end} ']
else

self.dataType.name

[\text{containing}'\text{foreach }\text{cContent }\text{in self.content separator[',']} \text{ as context in } \text{<TYPOCONTENTLABEL> end} ']
endif

Constraints

There are no constraints specified.

Comments

No comments.

Example

the "GUC packet"
the (typed) 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 as context in <QUALIFIER> end]
  self.name
  if self.value.oclIsUndefined() then
    ['containing' foreach c:Content in self.content separator '|' as context in CONTENTLABEL end ']
  else
    self.value as context in <VALUE>
  endif

TYPEDCONTENTLABEL ::= [foreach c:Comment in self.comment as context in <QUALIFIER> end]
  self.member.name
  if self.value.oclIsUndefined() then
    ['containing' foreach c:Content in self.content separator '|' as context in TYPEDCONTENTLABEL 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 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 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

CONTENTREFERENCELABEL ::= 'the' 'value' 'contained' 'in'

| foreach c:Comment in self.comment 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

DATAREFERENCELABEL ::= self.comment->first as context in <REFERENCEQUALIFIER>

| self.comment->first as context in <QUANTIFIEDQUALIFIER>

| 'predefined' |

| foreach c:Comment in self.comment as context in <QUALIFIER> end |

self.content as context in <STATICDATASELLABEL>

DATAREFERENCELABEL ::= [self.name]

| self.comment->first as context in <REFERENCEQUALIFIER> |

| foreach c:Comment in self.comment as context in <QUALIFIER> end |

self.content as context in <STATICDATASELLABEL>
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

STATICDATAUSE LABEL ::= if selfoclIsTypeOf(DataInstanceUse) then self as context in<DataInstanceUSE LABEL>
else if selfoclIsTypeOf(AnyValue) then self as context in<ANYVALUE LABEL>
else if selfoclIsTypeOf(AnyValueOrOmitValue) then self as context in<ANYVALUEOROMITVALUE LABEL>
else if selfoclIsTypeOf(OmitValue) then self as context in<OMITVALUE LABEL>
endif

Constraints
There are no constraints specified.

Comments
No comments.

Example
FullHeader
   any Header
   any or omitted
   omitted

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

ANYVALUE LABEL ::= 'any' self.dataType.name

Constraints
There are no constraints specified.
Comments
No comments.

Example
  `any Reader`

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
  ANYVALUEOREMITLABEL ::= 'any' 'or' 'omitted'
```

Constraints
There are no constraints specified.

Comments
No comments.

Example
  `any or omitted`

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
  OMISSLABEL ::= 'omitted'
```

Constraints
There are no constraints specified.

Comments
No comments.

Example
  `omitted`
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:ArgumentSpecific in self.argument separator[', '] 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 as context in <QUALIFIER> end]
    self.dataUse as context in <STATICDATASELABEL>]
```

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

\text{T}IM\text{E}\text{L}AB\text{E}L ::= '\text{.}' 'at' 'time' 'point' self.name

Constraints

There are no constraints specified.

Comments

No comments.

Example

(\text{.}) \text{ 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

\text{T}IM\text{E}\text{C}ON\text{STR}A\text{IN}T ::= '\text{!}'

[\text{each c:Comment in self.comment as context in <QUALIFIERS> end}]

self.timeConstraintExpression.dataInstance.name

Constraints

There are no constraints specified.

Comments

No comments.

Example

(\text{!}) 30s \text{ after t}
(\text{!}) \text{ within 5s of t}
(\text{!}) \text{ during the 5s after t}
6.6 Event Templates

6.6.1 EventSpecificationTemplate

Concrete Graphical Notation

Formal Description

```
context EventSpecificationTemplate
  EventOccurrenceTemplateLABEL ::=
    self.name
  EventOccurrenceTemplateSPECIFICATIONLABEL ::=
    [self.entityReference as context in_entityReference]<EVENTREFERENCELABEL>
    self.eventReference as context in_eventReference><EVENTREFERENCELABEL>
    [foreach e:EntityReference in self.oppositeEntityReference separator'|' as context in_OppositeEntityReference end]
    [foreach c:Comment in self.comment separator'|' as context in_NoteLabel end]
```

Constraints

There are no constraints specified.

Comments

No comments.

Example

```
EventOccurrenceTemplate
ReceiveBeacon

the IUT entity having received a "Beacon information" from the ItsNodeB entity
```

6.6.2 EventTemplateOccurrence

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 EventTemplateOccurrence
  EventTemplateLABEL ::=
    [self.comment->first| as context in_QUALIFIER]
    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
    endif
```

Constraints

There are no constraints specified.

Comments

Optionally, an 'EventTemplateOccurrence' may be visually represented as the content of the referenced 'EventSpecificationTemplate's 'EventOccurrenceSpecification', where bound 'EntityReference's from the 'EventOccurrenceSpecification' in the 'EventSpecificationTemplate' shall be substituted by the 'EntityReference's provided in the 'EventTemplateOccurrence'. Similarly, the argument from the 'EventOccurrenceSpecification' in the 'EventSpecificationTemplate' shall substituted by the argument provided in the 'EventTemplateOccurrence'.

Example

```plaintext
event ReceiveBeacon occurs
   (. at time point t1 : event ReceiveBeacon occurs
   (!) 30s after t1 : event ReceiveBeacon occurs
   event ReceiveBeacon occurs with {
      the ItsNodeB entity replaced by an ItsNodeC entity
   }
   event ReceiveBeacon occurs with {
      argument replaced by a "Beacon confirmation"
   }
   event ReceiveBeacon occurs with {
      the ItsNodeB entity replaced by an ItsNodeC entity
      argument replaced by a "Beacon confirmation"
   }
```

6.6.3 EntityBinding

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

```plaintext
context EntityBinding

ENTITY_BINDING_LABEL ::= self.templateEntity as context in <ENTITYREFERENCELABEL>
   'replaced' 'by'
   self.occurrenceEntity as context in <ENTITYREFERENCELABEL>
```

Constraints

There are no constraints specified.

Comments

No comments.
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
    
    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"
    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"
    Initial conditions with {
      the UE entity "in" the "E-UTRA RRC_CONNECTED state"
    }
  }
```
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 "IMS, clause 4.5.1" {
//a possible specification of the test objectives from clause 4.5.1 in [i.3]
//some interpolation has been applied to fit into the overall framework and concrete syntax
//of the present document

Domain{
  entities:
  - UE_A
  - UE_B
  - IMS_B
  ;
  events :
  - sends
  - receives
  ;
}

Test Purpose {
  TP Id  TP_IMS_4002_1
  Test objective ""
  Reference "ETSI TS 124 229 [1], clause 4.2A, paragraph 1",
  "ts_18601102v030101p.pdf::4.5.1.1 (CC 1)"

Expected behaviour ensure that {
  when {
    the UE_A entity sends a MESSAGE
    containing Message_Body_Size indicating value greater than 1 300 bytes;
    to the UE_B entity
  }
  then {
    the IMS_B entity receives the MESSAGE
    containing Message_Body_Size indicating value greater than 1 300 bytes;
  }
}
}
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>
<thead>
<tr>
<th>::=</th>
<th>is defined to be</th>
</tr>
</thead>
<tbody>
<tr>
<td>abc</td>
<td>the non-terminal symbol abc</td>
</tr>
<tr>
<td>abc xyz</td>
<td>abc followed by xyz</td>
</tr>
<tr>
<td>abc</td>
<td>alternative (abc or xyz)</td>
</tr>
<tr>
<td>abc</td>
<td>0 or 1 instance of abc</td>
</tr>
<tr>
<td>abc+</td>
<td>1 or more instances of abc</td>
</tr>
<tr>
<td>abc</td>
<td>0 or more instances of abc</td>
</tr>
<tr>
<td>'a'-'z'</td>
<td>all characters from a to z</td>
</tr>
<tr>
<td>(...)</td>
<td>denotes a textual grouping</td>
</tr>
<tr>
<td>'abc'</td>
<td>the terminal symbol abc</td>
</tr>
<tr>
<td>;</td>
<td>production terminator</td>
</tr>
<tr>
<td>\</td>
<td>the escape character</td>
</tr>
</tbody>
</table>

B.2 Production Rules

```
Package ::= 'Package' Identifier '{'
            { ElementImport }
            [ 'Domain' '{'
              [ 'pics' ':' { PICS }+ ';' ]
              [ 'entity' 'types' ':' { EntityType }+ ';' ]
              [ 'entities' ':' { Entity }+ ';' ]
              [ 'events' ':' { Event }+ ';' ]
              [ 'event' 'templates' ':' { EventOccurrenceTemplate }+ ';' ]
            }
            'Data' '{'
            { StructuredDataType }
            { StructuredDataInstance } '}'
            [ 'Configuration' '{'
              [ GateType ]
              [ ComponentType ]
              [ TestConfiguration ] }' ]
            { StructuredTestObjective }
            ( Group ) '}' ;

ElementImport ::= 'import'
                { 'all' | { Identifier | ( ',' Identifier ) } }
                'from' Identifier ';' ;

Group ::= 'Group' Identifier '{'
       { ElementImport }
       { StructuredTestObjective }
       ( Group ) '}' ;

PICS ::= '-' Identifier [ ( 'Qualifier' ) ] ;
PICSReference ::= [ AndOrQualifier ] Identifier ;
EntityType ::= 'abc' Identifier ;
```

ETSI
Entity ::= '-' Identifier

EventType ::= '-' Identifier

Annotation ::= Identifier

Event ::= '-' Identifier

EventOccurrenceTemplate ::= '-' Identifier '{' EventSpecification '}

EventSpecification ::= EntityReference EventReference Argument

StructuredTestObjective ::= 'Test Purpose' '{' 'TP Id' Identifier ['Test objective' Identifier ] ['Reference' Identifier '{' 'Config Id' Identifier ] ['PICS Selection' PICSReference { PICSReference } ] ['Expected behaviour'] ['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 ::= RepeatedEventSequence | SimpleEventSequence

RepeatedEventSequence ::= 'repeat' [ ( 'every' | IterationValue ) | ( IterationValue | 'times' ) ] '{' FirstEventOccurrence { EventOccurrence } '}'

FirstEventOccurrence ::= FirstEventOccurrenceSpecification | FirstEventTemplateOccurrence

FirstEventOccurrenceSpecification ::= [ ( TimeLabel | ( ( ',' | TimeConstraint ) | ':' ) ) TimeConstraint ] EntityReference EventReference Argument

FirstEventTemplateOccurrence ::= [ ( TimeLabel | ( ( ',' | TimeConstraint ) | ':' ) ) TimeConstraint ] 'event' Identifier 'occurs'

with '{' EntityBinding { ',' EntityBinding } ['argument' 'replaced' 'by' Argument ] '}'

EventOccurrence ::= EventOccurrenceSpecification | EventTemplateOccurrence

EventOccurrenceSpecification ::= [ ( TimeLabel | ( ( ',' | TimeConstraint ) | ':' ) ) TimeConstraint ] EntityReference EventReference Argument

EventTemplateOccurrence ::= AndOrQualifier [ ( TimeLabel | ( ( ',' | TimeConstraint ) | ':' ) ) TimeConstraint ] 'event' Identifier 'occurs'

with '{' EntityBinding { ',' EntityBinding } ['argument' 'replaced' 'by' Argument ] '}'

EntityBinding ::= EntityReference 'replaced' 'by' EntityReference

Note ::= ('Note' NumberAsIdentifier ':' Identifier ')

EventOccurrenceSpecification ::= AndOrQualifier [ ( TimeLabel | ( ( ',' | TimeConstraint ) | ':' ) ) TimeConstraint ] EntityReference EventReference Argument

EventTemplateOccurrence ::= AndOrQualifier [ ( TimeLabel | ( ( ',' | TimeConstraint ) | ':' ) ) TimeConstraint ] 'event' Identifier 'occurs'

with '{' EntityBinding { ',' EntityBinding } ['argument' 'replaced' 'by' Argument ] '}'

Note ;
TimeLabel ::= '(.)' 'at' 'time' 'point' Identifier ;

TimeConstraint ::= '(!)' { Qualifier }
    TimeConstraintQualifier ( Qualifier | CommonWordQualifier | ArticleQualifier )
    TimeConstraintExpression ':' ;

TimeConstraintExpression ::= ConstraintTimeLabelUse | ConstraintDataInstanceUse ;

ConstraintDataInstanceUse ::= Identifier | NumberAsIdentifier ;

ConstraintTimeLabelUse ::= Identifier ;

TimeConstraintQualifier ::= { 'before' | 'after' | 'during' | 'within' } ;

EntityReference ::= ArticleQualifier ( ( Identifier 'entity' ) | ( Identifier 'component' ) ) ;

OppositeEntityReference ::= DirectionQualifier ArticleQualifier ( ( Identifier 'entity' ) | ( Identifier 'component' ) ) ;

EventReference ::= { Qualifier | CommonWordQualifier } Identifier ;

Argument ::= LiteralValueAsArgument
    TypedLiteralValueAsArgument
    DataReferenceAsArgument
    ContentReferenceAsArgument
    LiteralValueReferenceArgument ;

Value ::= LiteralValue
    DataReference
    ContentReference
    LiteralValueReference ;

TypedValue ::= TypedLiteralValue
    DataReference
    ContentReference
    LiteralValueReference ;

IterationValue ::= IterationLiteralValue | IterationDataReference ;

TypedLiteralValueAsArgument ::= ( ArticleQualifier | QuantifiedQualifier ) 'typed'
    ( Qualifier )
    Identifier
    [ 'containing'
        TypedDataContent ( ',' TypedDataContent ) ';' ] ;

TypedLiteralValue ::= AssignmentQualifier
    ( Qualifier )
    Identifier
    [ 'containing'
        TypedDataContent ( ',' TypedDataContent ) ';' ] ;

TypedDataContent ::= { Qualifier }
    Identifier
    [ 'containing'
        TypedDataContent ( ',' TypedDataContent ) ';' ] ;

LiteralValueAsArgument ::= ( ArticleQualifier | QuantifiedQualifier )
    ( Qualifier )
    Identifier
    [ 'containing'
        DataContent ( ',' DataContent ) ';' ] ;

LiteralValue ::= AssignmentQualifier
    ( Qualifier )
    Identifier
    [ 'containing'
        DataContent ( ',' DataContent ) ';' ] ;

IterationLiteralValue ::= ( Identifier | NumberAsIdentifier )
    [ 'containing'
        DataContent ( ',' DataContent ) ';' ] ;

IterationDataReference ::= RepetitionDataInstanceUse ;

DataContent ::= { Qualifier }
    Identifier
    [ 'containing'
        DataContent ( ',' DataContent ) ';' ] ;

Identifier ::= STRING | ID ;

Qualifier ::= Identifier | NumberAsIdentifier ;

CommonWordQualifier ::= 'before'
    'after'
    'from'
    'to'
    'of' ;

ArticleQualifier ::= 'a'
    'an'
    'the' ;

QuantifiedQualifier ::= 'all'
    'any'
    'few' ;
AssignmentQualifier ::= 'indicating value' | 'set to';
AndOrQualifier ::= 'and' | 'or';
DirectionQualifier ::= 'by';
| 'in'
| 'into'
| 'from'
| 'to';
ReferenceQualifier ::= 'corresponding to'
| 'derived from'
| 'carrying'
| 'contained in'
| 'associated with';
DataInstanceUse ::= ( Identifier | NumberAsIdentifier )
[ 'containing'
ParameterBinding { ',' ParameterBinding } ';'];
RepetitionDataInstanceUse ::= Identifier | NumberAsIdentifier;
StaticDataUse ::= DataInstanceUse
| AnyValue
| AnyValueOrOmit
| OmitValue;
AnyValue ::= 'any' Identifier;
AnyValueOrOmit ::= 'any' 'or' 'omitted';
OmitValue ::= 'omitted';
ParameterBinding ::= 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;
TestConfiguration ::= 'Test Configuration'
Identifier
[ 'containing'
ComponentInstance { ComponentInstance } ]
Connection { Connection };
ComponentInstance ::= ComponentInstanceRole
| 'component' Identifier 'of' 'type' Identifier;
Connection ::= 'connection' 'between' G
terReference 'and' GateReference;
GateReference ::= 'Identifier '.' Identifier;
GateType ::= 'Interface' 'Type' Identifier
[ 'accepts' Identifier { ',' Identifier } ];
ComponentType ::= 'Component' 'Type' Identifier;
'with' { Timer } { Variable } { GateInstance } ';' ;

Timer ::= 'timer' Identifier ;
Variable ::= 'variable' Identifier 'of' 'type' Identifier ;
GateInstance ::= 'gate' Identifier 'of' 'type' Identifier ;
ComponentInstanceRole ::= ( 'SUT' | 'Tester' ) ;
ID ::= ( [ 'a'-'z' | 'A'-'Z' | '_' ] { 'a'-'z' | 'A'-'Z' | '_' | '0'-'9' | '/' } ) ;
INT ::= { '0'-'9' }+ ;
DQ ::= *** ;
SQ ::= *** ;
STRING ::= ( ( DQ |
| ( 'b' | 't' | 'n' | 'f' | 'r' | 'u' | '"' |
| *** | '"' ) ) |
| ( '\\' | DQ ) ) |
| SQ |
| ( 'b' | 't' | 'n' | 'f' | 'r' | 'u' |
| *** | *** | '"' ) ) |
| ( '\\' | SQ ) ) ;
ML_COMMENT ::= ( '/*' '*/' ) ;
SL_COMMENT ::= ( '//' ( '\n' | '\r' ) [ [ '\r' ] '\n' ] ) ;
WS ::= | ' ' |
| '\t' |
| '\r' |
| '\n' ;
## History

<table>
<thead>
<tr>
<th>Document history</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.1.1</td>
</tr>
<tr>
<td>V1.2.1</td>
</tr>
<tr>
<td>V1.3.1</td>
</tr>
<tr>
<td>V1.3.1</td>
</tr>
</tbody>
</table>