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

Intellectual Property Rights ........................................................................................................ 6

Foreword ....................................................................................................................................... 6

Modal verbs terminology ............................................................................................................. 6

Introduction .................................................................................................................................. 6

1 Scope ........................................................................................................................................ 8

2 References .................................................................................................................................. 8
2.1 Normative references ........................................................................................................... 8
2.2 Informative references ....................................................................................................... 8

3 Definition of terms, symbols and abbreviations .................................................................... 9
3.1 Terms ................................................................................................................................... 9
3.2 Symbols ............................................................................................................................... 9
3.3 Abbreviations ...................................................................................................................... 9

4 Basic principles ....................................................................................................................... 10
4.1 Structured Test Objective Specification ............................................................................ 10
4.2 Document Structure ........................................................................................................... 10
4.3 Notational Conventions .................................................................................................... 11
4.4 Element Operations .......................................................................................................... 11
4.5 Conformance ...................................................................................................................... 11

5 Meta-Model Extensions ........................................................................................................... 11
5.1 Overview ............................................................................................................................ 11
5.2 Foundation Abstract Syntax and Classifier Description .................................................... 12
5.2.1 Entity .......................................................................................................................... 12
5.2.2 Event ......................................................................................................................... 12
5.2.3 PICS .......................................................................................................................... 13
5.3 Test Objective Abstract Syntax and Classifier Description ................................................ 14
5.3.1 StructuredTestObjective ......................................................................................... 14
5.3.2 PICSReference ........................................................................................................ 15
5.3.3 InitialConditions ....................................................................................................... 16
5.3.4 ExpectedBehaviour ................................................................................................... 16
5.3.5 FinalConditions ........................................................................................................ 16
5.4 Events Abstract Syntax and Classifier Description ............................................................. 17
5.4.1 EventSequence ......................................................................................................... 17
5.4.2 RepeatedEventSequence ........................................................................................ 18
5.4.3 EventOccurrence ....................................................................................................... 18
5.4.4 EventOccurrenceSpecification ............................................................................... 19
5.4.5 EntityReference ....................................................................................................... 20
5.4.6 EventReference ........................................................................................................ 20
5.5 Data Abstract Syntax and Classifier Description ................................................................. 21
5.5.1 Value ......................................................................................................................... 21
5.5.2 LiteralValue ............................................................................................................... 21
5.5.3 Content ..................................................................................................................... 22
5.5.4 LiteralValueReference ............................................................................................ 22
5.5.5 ContentReference .................................................................................................... 23
5.5.6 DataReference .......................................................................................................... 23
5.6 Event Templates Abstract Syntax and Classifier Description .............................................. 24
5.6.1 EventSpecificationTemplate .................................................................................... 24
5.6.2 EventTemplateOccurrence ...................................................................................... 25
5.6.3 EntityBinding .......................................................................................................... 25
5.7 Structured Test Objective Variants Abstract Syntax and Classifier Description ..................... 26
5.7.1 StructuredTestObjectiveVariant ............................................................................ 26
5.7.2 Variants .................................................................................................................... 27
5.7.3 VariantBinding ......................................................................................................... 27
5.8 Predefined TDL Model Instances ....................................................................................... 28
5.8.1 Overview .................................................................................................................. 28
5.8.2 Predefined Instances of the 'AnnotationType' Element .............................................. 28
5.8.2.1 Initial conditions .................................................................................................... 28
5.8.2.2 Expected behaviour .............................................................................................. 28
5.8.2.3 Final conditions ........................................................................................................ 28
5.8.2.4 when ..................................................................................................................... 28
5.8.2.5 then ...................................................................................................................... 28
5.8.2.6 Test Purpose Description ..................................................................................... 28

6  Graphical Syntax Extensions ............................................................................................. 29
6.1 Foundation ..................................................................................................................... 29
6.1.1 Entity ....................................................................................................................... 29
6.1.2 Event ....................................................................................................................... 29
6.1.3 PICS .......................................................................................................................... 30
6.1.4 Comment ................................................................................................................... 30
6.2 Test Objective ................................................................................................................. 32
6.2.1 StructuredTestObjective .......................................................................................... 32
6.3 Events ............................................................................................................................ 34
6.3.1 EventSequence ........................................................................................................ 34
6.3.2 RepeatedEventSequence ......................................................................................... 34
6.3.3 EventOccurrence ...................................................................................................... 35
6.3.4 EventOccurrenceSpecification ............................................................................... 35
6.3.5 EntityReference ....................................................................................................... 36
6.3.6 EventReference ........................................................................................................ 37
6.4 Data .................................................................................................................................. 37
6.4.1 Value ......................................................................................................................... 37
6.4.2 LiteralValue .............................................................................................................. 38
6.4.3 Content ...................................................................................................................... 38
6.4.4 LiteralValueReference .............................................................................................. 39
6.4.5 ContentReference .................................................................................................... 39
6.4.6 DataReference .......................................................................................................... 40
6.4.7 StaticDataUse .......................................................................................................... 41
6.4.8 AnyValue .................................................................................................................. 41
6.4.9 AnyValueOrOmit ..................................................................................................... 42
6.4.10 OmitValue .............................................................................................................. 42
6.4.11 DataInstanceUse ................................................................................................... 43
6.4.12 ParameterBinding ................................................................................................. 43
6.5 Time .................................................................................................................................. 44
6.5.1 TimeLabel ................................................................................................................ 44
6.5.2 TimeConstraint ....................................................................................................... 44
6.6 Event Templates .............................................................................................................. 45
6.6.1 EventSpecificationTemplate ..................................................................................... 45
6.6.2 EventTemplateOccurrence ...................................................................................... 45
6.6.3 EntityBinding .......................................................................................................... 46
6.7 Variants .......................................................................................................................... 47
6.7.1 StructuredTestObjectiveVariant .............................................................................. 47
6.7.2 Variants ................................................................................................................... 48
6.7.3 VariantBinding ........................................................................................................ 48

7  Exchange Format Extensions ............................................................................................ 48
8  Textual Syntax Extensions .................................................................................................. 49
8.0 Terminals ....................................................................................................................... 49
8.1 Foundation ..................................................................................................................... 49
8.1.1 Entity ....................................................................................................................... 49
8.1.2 Event ....................................................................................................................... 50
8.1.3 PICS .......................................................................................................................... 50
8.1.4 Comment ................................................................................................................... 50
8.1.5 AnnotationType ....................................................................................................... 51
8.1.6 Annotation ............................................................................................................... 52
8.1.7 PackageableElement ............................................................................................... 53
8.1.8 Element .................................................................................................................... 53
8.2 Test Objective ................................................................................................................ 53
Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are 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 Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, 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.

DECT™, PLUGTESTS™, UMTS™ and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. 3GPP™ and LTE™ are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. oneM2M™ logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. GSM® and the GSM logo are trademarks registered and owned by the GSM Association.

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.

NOTE: The use of underline (additional text) and strikethrough (deleted text) highlights the differences between base document and extended documents.
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".
3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI ES 203 119-1 [1], ETSI ES 203 119-2 [2], ETSI ES 203 119-3 [3], ETSI ES 203 119-8 [4] 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 Symbols

Void.

3.3 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
- OCL: Object Constraint Language™
- 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 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 light weight 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.

Test objective variants may be included in a structured test objective to describe additional test objectives derived from the structured test objective by specialising or overriding data elements and meta information.

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).
- Graphical concrete syntax extension describing corresponding shapes for the representation of the additional concepts (clause 6).
- Exchange format extension describing corresponding representation of the additional concepts (clause 7).
- Textual concrete syntax extension describing corresponding derivations for the representation of the additional concepts (clause 8).
- 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], ETSI ES 203 119-2 [2], and ETSI ES 203 119-8 [4].

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 for the graphical syntax extensions follows the notational conventions described in clause 4.5 of the Graphical Syntax [2]. The concrete syntax notation specification for the textual syntax extensions follows the basic principles described in clause 4.3 of the Textual Syntax [4] and the general rules described in clause 5 of the Textual Syntax [4].

4.4 Element Operations

The following operations shall be provided in an implementation of the TDL-TO extension to the TDL meta-model in order to ensure the semantic integrity of TDL-TO models, in addition to the operations defined for the TDL meta-model in ETSI ES 203 119-1 [1]. The operations are also used as reusable shortcuts for the specification of the formalized constraints and are required for their interpretation, in addition to the operations provided by the standard library of OCL:

- 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.

- OclAny `getNotes()`: Set<Comment> - applicable on any TDL 'Element', returns a set of all named 'Comment's that are contained within the 'Element'.

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'.

A 'StructuredTestObjective' may include 'Variants' which define new 'StructuredTestObjective's based on this 'StructuredTestObjective'.

Generalizations

- TestObjective
Properties

- `picsReference` : `PICSReference [*]` (ordered)
  An ordered set of 'PICSReferences' to 'PICS'.
- `configuration` : `TestConfiguration [0..1]`
  A reference to a 'TestConfiguration'.
- `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'.
- `variants` : `Variants [0..1]`
  Container for 'StructuredTestObjectiveVariant's.

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'. A 'Comment' with body containing 'not' may be used to indicate that the referenced 'PICS' is to be negated.

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

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'.

ETSI
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'.
  inv: `MultipleEventOccurrences`

```plaintext
self.container().oclIsTypeOf(EventSpecificationTemplate)
or self.container().events->size() < 2
or self.container().events->forall(o | self.container().events->at(0) = o
or (not o.comment->isEmpty()
and (o.comment->first().body = 'and'
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'.

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

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'.

Figure 5.4: Data Concepts
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'.


Constraints

- **Referenced 'LiteralValue' visibility**
  Only 'LiteralValue's defined within previous 'EventOccurrence's of the containing 'StructuredTestObjective' may be referenced.
  
  **inv: VisibleValue:**
  
  self.getTestObjective().contains(self.content) 
  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.
  
  **inv: VisibleContent:**
  
  self.getTestObjective().contains(self.content) 
  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'.
Constraints

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

5.6 Event Templates Abstract Syntax and Classifier Description

5.6.1 EventSpecificationTemplate

\textbf{Figure 5.5: Event Templates Concepts}

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

- \texttt{eventSpecification : EventOccurrenceSpecification [1]}
  A reusable 'EventOccurrenceSpecification'.
Constraints

There are no constraints specified.

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))))

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.

5.7 Structured Test Objective Variants Abstract Syntax and Classifier Description

5.7.1 StructuredTestObjectiveVariant

Figure 5.6: Structured Test Objective Variant Concepts

Semantics

A 'StructuredTestObjectiveVariant' is a refinement of 'TestObjective' defined on the basis of a 'StructuredTestObjective'. In addition to the 'description' and 'objectiveURI' properties inherited from 'TestObjective', a 'StructuredTestObjectiveVariant' may include 'PICSReference's and 'VariantBinding's identifying the substitutions to be applied to derive the 'StructuredTestObjectiveVariant'.

Generalizations

- TestObjective
Properties

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

- bindings: VariantBinding [*] {ordered}
  A set of 'VariantBinding's specifying 'Value's that override corresponding 'Value's within the 'StructuredTestObjective' from which the 'StructuredTestObjectiveVariant' is derived.

Constraints

There are no constraints specified.

5.7.2 Variants

Semantics

'Variants' is an 'Element' contained in a 'StructuredTestObjective', serving as a container for a non-empty set of 'StructuredTestObjectiveVariant's.

Generalizations

- Element

Properties

- variants : StructuredTestObjectiveVariant' [1..*] {ordered}
  A non-empty set of 'StructuredTestObjectiveVariant's.

Constraints

There are no constraints specified.

5.7.3 VariantBinding

Semantics

'VariantBinding' is an 'Element' used to specify the substitutions in the associated 'StructuredTestObjective' in order to describe a 'StructuredTestObjectiveVariant'.

Generalizations

- Element

Properties

- value : Value [1]
  A value defined in the associated 'StructuredTestObjective' that is to be substituted.

- boundTo : Value [1]
  The value to be used in the 'StructuredTestObjectiveVariant'.
Constraints

- **Referenced 'Value' of 'VariantBinding'**
  
  If the 'value' property references a 'LiteralValue' or 'Content' element, the referenced element shall be contained in the 'StructuredTestObjective' containing the 'VariantBinding'.

  \[
  \text{inv: VariantBindingValues:} \]
  \[
  \begin{align*}
  &\text{self.value.oclIsTypeOf(LiteralValueReference) implies} \\
  &\text{self.value.oclAsType(LiteralValueReference).content.getTestObjective()} = \text{self.getTestObjective()} \\
  &\text{and} \\
  &\text{or self.value.oclIsKindOf(ContentReference) implies} \\
  &\text{self.value.oclAsType(ContentReference).content.getTestObjective()} = \text{self.getTestObjective()} \\
  \end{align*}
  \]

5.8 Predefined TDL Model Instances

5.8.1 Overview

This clause lists the extensions to predefined element instances for various meta-model elements that shall be a part of a standardcompliant TDL implementation.

5.8.2 Predefined Instances of the 'AnnotationType' Element

5.8.2.1 Initial conditions

The predefined 'AnnotationType' 'Initial conditions' is the 'key' of an 'Annotation' that may be attached to an 'EventSequence' or a 'CompoundBehaviour' element. The 'value' of the shall be left unspecified.

5.8.2.2 Expected behaviour

The predefined 'AnnotationType' 'Expected behaviour' is the 'key' of an 'Annotation' that may be attached to an 'EventSequence' or a 'CompoundBehaviour' element. The 'value' of the shall be left unspecified.

5.8.2.3 Final conditions

The predefined 'AnnotationType' 'Final conditions' is the 'key' of an 'Annotation' that may be attached to an 'EventSequence' or a 'CompoundBehaviour' element. The 'value' of the shall be left unspecified.

5.8.2.4 when

The predefined 'AnnotationType' 'Final conditions' is the 'key' of an 'Annotation' that may be attached to a 'CompoundBehaviour' element. The 'value' of the shall be left unspecified.

5.8.2.5 then

The predefined 'AnnotationType' 'Final conditions' is the 'key' of an 'Annotation' that may be attached to a 'CompoundBehaviour' element. The 'value' of the shall be left unspecified.

5.8.2.6 Test Purpose Description

The predefined 'AnnotationType' 'Test Purpose Description' is the 'key' of an 'Annotation' that may be attached to a 'TestDescription' element. The 'value' of the shall be left unspecified.
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

```
Entity
ENTITylabel

Entity
IUT

Entity
TESTER

Entity
BUFFER
```

6.1.2 Event

Concrete Graphical Notation

Formal Description

```context Event
EVENTylabel := 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

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

NOTQualifier ::= if self.body = 'not'
then
 self.body
endif

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 ::= if not self.name.oclIsUndefined then """Note self.name ': self.body ''"""endif
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

<table>
<thead>
<tr>
<th>Test Objective</th>
<th>TestObjectiveNameLabel</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

**Initial Conditions**

- `INITIALCONDITIONSLabel`
- `EXPECTEDBEHAVIOURLABEL`
- `FINALCONDITIONSLabel`
- `NOTESLabel`

**Tests Purpose**

<table>
<thead>
<tr>
<th>Test Purpose</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>Configuration</td>
<td>&lt;ConfigLabel&gt;</td>
</tr>
<tr>
<td>PICS Selection</td>
<td>&lt;PICSSelectionLabel&gt;</td>
</tr>
<tr>
<td>Initial Conditions</td>
<td>INITIALCONDITIONSLabel</td>
</tr>
<tr>
<td>Expected Behaviour</td>
<td>EXPECTEDBEHAVIOURLABEL</td>
</tr>
<tr>
<td>Final Conditions</td>
<td>FINALCONDITIONSLabel</td>
</tr>
</tbody>
</table>

**Formal Description**

```plaintext
context StructuredTestObjective

TESTOBJECTIVENAMELABEL := self.name
DESCRIPTIONLABEL := self.description
URIOfObjectiveLabel := self.objectiveURI->newline

PICSSelectionLabel := foreach p:PICSReference in self.picsReferences as context in <PICSREFERENCELABEL>

INITIALCONDITIONSLabel := self.initialConditions.conditions as context in <EVENTSEQUENCENAMELABEL>

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

ETSI
self.expectedBehavior.thenClause as context in <EVENTSEQUENCE>=""
endif
""

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

NOTESLABEL ::= foreach p:Comment in self.getNotes() p as context in <NOTELABEL> end

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, the FINALCONDITIONSLABEL, and the NOTESLABEL 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 mandatory only if the related compartments are displayed, otherwise they may be hidden. The compartment with the NOTESLABEL shall contain only named 'Comment's.

In the alternate notation shown above, all compartments except the TestObjective compartment are optional and only
displayed if the respective model elements are defined. Named 'Comment's are displayed as floating shapes (inherited from ETSI ES 203 119-2 [2]).

Example

<table>
<thead>
<tr>
<th>TP Id</th>
<th>TP/GCONW/FTUV/BATV/BVI/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>

```
with {
  the IUT entity being "in" the initial state
}
```

```
Expected Behaviour

ensure that {
  when {
    the IUT entity is requested to send a "GUC packet"
  }
  then {
    the IUT entity sends a "GUC packet" containing
    BasicHeader containing
      "version field" indicating value "itsGnProtocolVersion MIB parameter" ,
      "RHL field" indicating value "itsGnDefaultHopLimit MIB parameter"
    ;
  }
} 
```
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

```plaintext
context EventSequence
EventSequenceLABEL ::= if self.oclIsTypeOf(RepeatedEventSequence) then self as context in <REPEATEDEVENTSEQUENCELABEL> else if self.oclIsTypeOf(EventSequence) then self as context in <SIMPLEEVENTSEQUENCELABEL> endif

SIMPLEEVENTSEQUENCELABEL ::= 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

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

```plaintext
context RepeatedEventOccurrence
RepeatedEventSEQUENCELABEL ::= "repeat"
if self.interval.oclIsUndefined() then
  self.repetitions as context in <EVENTARGUMENTLABEL> "times"
else
  every self.interval as context in <EVENTARGUMENTLABEL>
endif

every e:EventOccurrence in self.events newline e as context in <EVENTOCCURRENCELABEL> end
```

Constraints

There are no constraints specified.

Comments

No comments.
Example

\[
\text{repeat 2 times} \{  \\
\text{the IUT entity having received a "Beacon information" from the ItsNodeB entity and}  \\
\text{the IUT entity having received any message from the ItsNodeD entity}  \\
\}\]

\[
\text{repeat every CBF_MAX} \{  \\
\text{the IUT entity saves the "GBC packet" into the CBF buffer entity and}  \\
\text{the IUT entity starts a "contention timer" containing}  \\
\text{duration set to CBF_MAX}  \\
\text{and}  \\
\text{the IUT entity broadcasts the received "GBC packet"}  \\
\}\]

6.3.3 EventOccurrence

Concrete Graphical Notation

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

Formal Description

context EventOccurrence

\[
\text{EVENT_OCCURRENCE_LABEL} := \begin{align*}
\text{if } \text{self.oclIsTypeOf(EventOccurrenceSpecification)} \text{ then } \text{self as context in } <\text{EVENT_OCCURRENCE_SPECIFICATION_LABEL}> \\
\text{else if } \text{self.oclIsTypeOf(EventTemplateOccurrence)} \text{ then } \text{self as context in } <\text{EVENT_TEMPLATE_OCCURRENCE_LABEL}> \\
\text{endif}
\end{align*}
\]

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

\[
\text{EVENT_OCCURRENCE_SPECIFICATION_LABEL} ::= \begin{align*}
\text{self.comment->first as context in } <\text{ANDQUALIFIER}>  \\
\text{if not self.oclIsUndefined then}  \\
\text{self.timeLabel as context in } <\text{TIME_LABEL}>  \\
\text{endif}  \\
\text{else}  \\
\text{self.timeLabel as context in } <\text{TIME_LABEL}>  \\
\text{if not self.oclIsUndefined then}  \\
\text{endif}
\end{align*}
\]
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

6.3.5 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 as context in QUALIFIER end]
if self.component.oclIsUndefined() then
  self.entity.name 'entity'
else
  self.component.name 'component'
endif

OppositeEntityLABEL ::= self.comment->at(1) as context in DIRECTIONQUALIFIER
self.component->at(1) as context in ARTICLEQUALIFIER
[foreach c:Comment in self.comment 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
from the ItsNodeB component
in the location service buffer entity, for the ItsNodeB component

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'.

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 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**

```
LiteralValueLabel ::= self.comment->[first][ as context in <ArticleQualifier>
  if not self.dataType.oclIsUndefined() then
    'typed'
  endif
  [foreach c:Comment in self.comment c as context in <Qualified>]
  if self.dataType.oclIsUndefined() then
    self.name
    [containing'foreach c:Content in self.content separator',' as context in <ContentLabel> end']
  else
    self.dataType.name
    [containing'foreach c:Content in self.content separator',' as context in <TypedContentLabel> end']
  endif
```

```
LiteralValueLabel ::= self.comment->[first][ as context in <AssignmentQualifier>
  [foreach c:Comment in self.comment c as context in <Qualified>]
  self.name
  [containing'foreach c:Content in self.content separator',' as context in <ContentLabel> end']
```

```
LiteralValueLabel ::= self.comment->[first][ as context in <AssignmentQualifier>
  [foreach c:Comment in self.comment c as context in <Qualified>]
  self.name
  [containing'foreach c:Content in self.content separator',' as context in <ContentLabel> end']
```

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 c as context in <QUALIFIER> end]

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

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

self.member.name

if self.value.oclIsUndefined() then
  ['containing' foreach c:Content in self.content separator ';' c as context in <TYPECONTENTLABEL> 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

LITERALREFERENCEARGUMENTLABEL ::= "the 'value' of"

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

self.content.name

LITERALREFERENCELABEL ::= 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

DATABASEARGUMENTLABEL :=

[if self.content->first] as context in<REFERENCEQUALIFIER>
[if self.content->first] as context in<QUALIFIER>
'predefined'

[foreach c:Comment in self.comment c as context in<QUALIFIER> end]
self.content as context in<STATE>DATAUSELABEL>

DATABASELABEL :=

[if self.name]
self.content->first as context in<REFERENCEQUALIFIER>

[foreach c:Comment in self.comment c as context in<QUALIFIER> end]
self.content as context in<STATE>DATAUSELABEL>

DATABASEBINDINGLABEL :=

'predefined'

[foreach c:Comment in self.comment c as context in<QUALIFIER> end]
self.content as context in<STATE>DATAUSELABEL>
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 <DATAINSTANCEUS ELABEL>
else if self.oclIsTypeOf(AnyValue) then self as context in <ANYVALUELABEL>
else if self.oclIsTypeOf(AnyValueOrOmitValue) then self as context in <ANYVALUEOROMITVALUELABEL>
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 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

ANYVALUELABEL := '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
AnyValueOrOmitLABEL ::= '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
OmitValueLABEL ::= '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

\[
\text{context } \text{DataInstanceUse} \\
\text{DATAVALUELABEL} := \text{self.dataInstance.name} \\
\text{[containing] } \\
\text{foreach a:ParameterBinding in self.argument separator[', '] as context in <PARAMETERBINDINGLABEL> end}
\]

Constraints

There are no constraints specified.

Comments

No comments.

Example

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

6.4.12 ParameterBinding

Concrete Graphical Notation

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

Formal Description

\[
\text{context } \text{ArgumentSpecification} \\
\text{PARAMETERBINDINGLABEL} := \text{self.parameter.name} \\
\text{self.comment->first[ ] as context in <ASSIGNMENTQUALIFIER> } \\
\text{foreach c:Comment in self.comment cas context in <QUALIFIER> end} \\
\text{self.dataUse as context in <STATICDATUSLABEL>}
\]

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
TIMELABEL ::= "\.|" '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 as context in <QUALIFIER> COMMONWORDQUALIFIER ARTICLEQUALIFIER> end]
self.timeConstraintExpression.dataInstance.name

Constraints

There are no constraints specified.

Comments

No comments.

Example

(1) 30s after t
(1) within 5s of t
(1) during the 5s after t
6.6 Event Templates

6.6.1 EventSpecificationTemplate

Concrete Graphical Notation

Formal Description

```plaintext
class EventSpecificationTemplate

context EventSpecificationTemplate

EVENTOCCURRENCETEMPLATELABEL := self.name

EVENTOCCURRENCETEMPLATESPECIFICATIONLABEL := [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 <OPPOSETITLELABEL> end]

[foreach c:Comment in self.comment separator(',') e as context in <NOTELABEL> end]
```

Constraints

There are no constraints specified.

Comments

No comments.

Example

```
Event Occurrence Template
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

```plaintext
class EventTemplateOccurrence

context EventTemplateOccurrence

EVENTTEMPLATEOCCURRENCELABEL := [self.comment->first as context in <ANNOTQUALIFIER>]

if self.timeLabel.oclIsUndefined() then
    if not self.timeConstraint.oclIsUndefined() then
        self.timeConstraint as context in <TIMECONSTRAINTLABEL>
    end
else
    self.timeLabel as context in <TIMELABEL>
end
```

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
    ENTITYBINDINGLABEL ::= self.templateEntity as context in "ENTITYREFERENCELABEL"
    replaced "by"
    self.occurrenceEntity as context in "ENTITYREFERENCELABEL"
```

Constraints

There are no constraints specified.

Comments

No comments.
Example

the ItsNodeB entity replaced by an ItsNodeC entity
the ITS_B component replaced by an ITS_C component

6.7 Variants

6.7.1 StructuredTestObjectiveVariant

Concrete Graphical Notation

<table>
<thead>
<tr>
<th>TP Id</th>
<th>Description</th>
<th>Reference</th>
<th>PICS</th>
<th>VARIANT_BINDING_LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARIANT_NAME_LABEL</td>
<td>DESCRIPTION_LABEL</td>
<td>URI_OF_OBJECTIVE_LABEL</td>
<td>&lt;PICS_SELECTION_LABEL&gt;</td>
<td>Binding Value Label</td>
</tr>
<tr>
<td>VARIANT_NOTES_LABEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Formal Description

context StructuredTestObjectiveVariant

VARIANT_NAME_LABEL := self.name

VARIANT_NOTES_LABEL := foreach p:Comment in self.getNotes() p as context in <NOTE_LABEL> end

Constraints

There are no constraints specified.

Comments

The 'StructuredTestObjectiveVariant's for a 'StructuredTestObjective' are represented as a table-like shape, where each 'StructuredTestObjectiveVariant' is represented as an individual row. Only the first column containing the names of the variants is always present in the shape. All other columns depend on the specification of the variants. If the variants specify the 'description', the 'objectiveURI', or the 'picsReference' properties, the corresponding columns and labels shall be shown. If 'VariantBinding's are specified, corresponding columns shall be added for each value that is bound with the label for the corresponding value as the heading and the label for the value that it is bound to in the corresponding column cell within for row corresponding to the variant. Different variants may specify different properties or values, in which case the corresponding cells for the unspecified properties and values shall be empty. The compartment with the VARIANT_NOTES_LABEL shall contain only named 'Comment's, aggregated from all 'StructuredTestObjectiveVariant's.

Example

<table>
<thead>
<tr>
<th>TP Id</th>
<th>Description</th>
<th>ID</th>
<th>HTTP_STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP_RESOURCE_GET_200v1</td>
<td>&quot;Read full contents of a resource with a valid ID&quot;</td>
<td>VALID_ID</td>
<td>200 OK</td>
</tr>
<tr>
<td>TP_RESOURCE_GET_404v2</td>
<td>&quot;Read contents of a resource with a non-existent ID returns 404&quot;</td>
<td>NONEXISTENT_ID</td>
<td>404 Not found</td>
</tr>
</tbody>
</table>
6.7.2 Variants

Concrete Graphical Notation
There is no shape associated with this element. It serves as a container for the individual 'StructuredTestObjectiveVariant's which are represented as rows within the same graphical representation.

Formal Description
No formal description.

Constraints
There are no constraints specified.

Comments
No comments.

Example
No example.

6.7.3 VariantBinding

Concrete Graphical Notation
There is no shape associated with this element. Instead, it is represented as labels within the context of a 'StructuredTestObjectiveVariant'.

Formal Description

\[
\text{context VariantBinding} \\
\text{VARIANT_BINDING_LABEL ::= self.content.name} \\
\text{_BINDING_VALUE_LABEL ::= if self.oclIsTypeOf(DataReference) then self as context in <DATA_REFERENCE_BINDING_LABEL> else if self.oclIsTypeOf(LiteralValue) then self as context in <LITERAL_VALUE_BINDING_LABEL> endif}
\]

Constraints
There are no constraints specified.

Comments
No comments.

Example
No examples.

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.
8 Textual Syntax Extensions

8.0 Terminals

In addition to the terminals and context-specific 'pseudo-terminal' data rules specified in clause 5.5 of ETSI ES 203 119-8 [4], for the extension to the textual syntax, the following context-specific 'pseudo-terminal' data rules are defined:

Assignments:
  'indicating' 'value'
  | 'set' 'to'

References:
  'corresponding' to
  | 'derived' from
  | 'carrying'
  | 'contained' in
  | 'associated' with

InitialBlockName:
  'Initial' 'conditions'

ExpectedBlockName:
  'Expected' 'behaviour'

FinalBlockName:
  'Final' 'conditions'

TestPurposeDescriptionName:
  'Test' 'Purpose' 'Description'

When:
  'when'

Then:
  'then'

KIdentifier:
  ID | 'sends' | 'receives' | 'triggers' | 'in'

8.1 Foundation

8.1.1 Entity

Concrete Textual Notation

Entity returns to Entity:
  AnnotationCommentFragment
  'entity' name=identifier

Comments

No Comment.
8.1.2 Event

Concrete Textual Notation

Event returns to Entity:
  AnnotationCommentFragment
  "Event" name=Identifier

Comments

No Comment.

Examples

Event sent
Event received
Event started
Event stopped
Event ready

8.1.3 PICS

Concrete Textual Notation

PICS returns to PICS:
  AnnotationCommentFragment
  "PICS" name=Identifier

Comments

No Comment.

Examples

PICS R1
PICS Radio
PICS Wifi

8.1.4 Comment

Concrete Textual Notation

Qualifier returns id::Comment:
  body=(Identifier | NIdentifier)

NotQualifier returns id::Comment:
  body="not"

AndOrQualifier returns id::Comment:
  body="and" | body="or"
ArticleQualifier returns tdl::Comment:
  body= 'a' | body= 'an' | body= 'the'
;
AssignmentQualifier returns tdl::Comment:
  body= Assignments
;
CommonWordQualifier returns tdl::Comment:
  body= 'before'
  | body= 'after'
  | body= 'from'
  | body= 'to'
  | body= 'of'
;
DirectionQualifier returns tdl::Comment:
  body= 'by'
  | body= 'in'
  | body= 'into'
  | body= 'from'
  | body= 'to'
;
QuantifiedQualifier returns tdl::Comment:
  body= 'all'
  | body= 'any'
  | body= 'few'
  | body= 'multiple'
  | body= 'no'
  | body= 'only'
  | body= 'several'
  | body= 'some'
;
ReferenceQualifier returns tdl::Comment:
  body= References
;
TimeConstraintQualifier returns tdl::Comment:
  body= 'before'
  | body= 'after'
  | body= 'during'
  | body= 'within'
;
Comments

The alternative derivations are used in the respective contexts. The derivations represent 'Comment's with pre-defined 'body' contents.

Examples

Void.

8.1.5 AnnotationType

Concrete Textual Notation

AnnotationType returns tdl::AnnotationType:
  "Annotation" name = |
    Identifier
    | InitialBlockName
    | ExpectedBlockName
    | FinalBlockName
    | TestPurposeDescriptionName
    | When
Comments

In addition to identifiers, dedicated names for the predefined 'AnnotationType's can be specified (even containing spaces). The rules for these dedicated names are externalised for reuse.

The alternative derivations are used in the respective contexts. The derivations represent pre-defined 'AnnotationType's which are used for 'Annotation's in specific contexts, such as specific 'EventSequence's or 'Block's.

Examples

```
<table>
<thead>
<tr>
<th>Annotation</th>
<th>Initial conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Expected behaviour</td>
</tr>
<tr>
<td>Annotation</td>
<td>Final conditions</td>
</tr>
<tr>
<td>Annotation</td>
<td>Test Purpose Description</td>
</tr>
</tbody>
</table>

8.1.6 Annotation

Concrete Textual Notation

InitialConditionsAnnotation returns tdl::Annotation:
key=[tdl::AnnotationType|InitialBlockName]
;

ExpectedBehaviourAnnotation returns tdl::Annotation:
key=[tdl::AnnotationType|ExpectedBlockName]
;

FinalConditionsAnnotation returns tdl::Annotation:
key=[tdl::AnnotationType|FinalBlockName]
;

TestPurposeDescriptionAnnotation returns tdl::Annotation:
key=[tdl::AnnotationType[TestPurposeDescriptionName]
;

WhenAnnotation returns tdl::Annotation:
key=[tdl::AnnotationType[When]]
;

ThenAnnotation returns tdl::Annotation:
key=[tdl::AnnotationType[Then]]
;
```

Comments

The alternative derivations are used in the respective contexts. The derivations represent 'Annotation's using pre-defined 'AnnotationType's in specific contexts, such as specific 'EventSequence's or 'Block's.

Examples

```
@communication
Event sent
@communication
Event received
```
8.1.7 PackageableElement

Concrete Textual Notation

```
PackageableElement returns tdl::PackageableElement:
  super |
  | Entity
  | Event
  | PICS
  | StructuredTestObjective
  | EventSpecificationTemplate
  | TestPurposeDescription

```

Comments

The alternatives for the elements from the extension defined in the present document are in addition to the base specification for the concrete syntax of 'PackageableElement', indicated here by 'super'.

Examples

Void.

8.1.8 Element

Concrete Textual Notation

```
fragment InitialConditionsFragment returns tdl::Element:
  annotation+=InitialConditionsAnnotation

fragment ExpectedBehaviourFragment returns tdl::Element:
  annotation+=ExpectedBehaviourAnnotation

fragment FinalConditionsFragment returns tdl::Element:
  annotation+=FinalConditionsAnnotation
```

Comments

The alternative derivations are used in the respective contexts. The derivations represent fragments for 'Annotation's using pre-defined 'AnnotationType's in specific contexts, such as specific 'EventSequence's or 'Block's.

Examples

Void.

8.2 Test Objective

8.2.1 StructuredTestObjective

Concrete Textual Notation

```
StructuredTestObjective returns tdl::StructuredTestObjective:
  "TestPurpose" name=Identifier
  BEGIN
  | "Objective": description=EString |
  | "Reference": objective URI=EString [ , objective URI=EString ] * |
  | "Configuration": configuration=[tdl::TestConfiguration | Identifier] |
  | "PICS": picsReference+=FirstPICSReference [ picsReference+=PICSReference | picsReference+=PICSReference | ] * |
  | initialConditions=InitialConditions |
```
8.2.2 PICSReference

Concrete Textual Notation

FirstPICSReference returns to: PICSReference:
|comment+=NotQualifier|?
pics=[to::PICS|Identifier]

Examples

PICS: Wifi or Radio
PICS: Wifi and Radio or R1
PICS: Radio

8.2.3 InitialConditions

Concrete Textual Notation

InitialConditions returns to: InitialConditions:
InitialConditionsFragment
when condition=EventSequence

Comments

No Comment.

Examples

PICS: Wifi or Radio
PICS: Wifi and Radio or R1
PICS: Radio
Comments
No Comment.

Examples

\[
\text{Initial conditions}
\begin{array}{l}
\text{with}
\text{the IUT entity having sent an empty request}
\end{array}
\]

8.2.4 Expected Behaviour

Concrete Textual Notation

Expected behaviour returns to ExpectedBehaviour:

\[
\text{ExpectedBehaviourFragment}
\begin{array}{l}
\text{'ensure' 'that'}
\end{array}
\]

\[
\text{BEGIN}
\begin{array}{l}
\text{when } \text{whenClause=} \text{EventSequence}
\text{then } \text{thenClause=} \text{EventSequence}
\text{END}
\end{array}
\]

\[
\text{\text{thenClause=} \text{EventSequence}}
\]


Comments
No Comment.

Examples

\[
\text{Expected behaviour ensure that}
\text{\begin{array}{l}
\text{when}
\text{\begin{array}{l}
\text{the IUT entity has received a connection closure}
\text{then}
\text{the IUT entity closes the connection}
\end{array}}
\end{array}}
\]

8.2.5 Final Conditions

Concrete Textual Notation

FinalConditions returns to: FinalConditions:

\[
\text{FinalConditionsFragment}
\begin{array}{l}
\text{'with' 'condition'=} \text{EventSequence}
\end{array}
\]


Comments
No Comment.

Examples

\[
\text{Final conditions with}
\text{the IUT entity being stopped}
\]
8.3 Events

8.3.1 EventSequence

Concrete Textual Notation

EventSequence returns to: EventSequence:
  BEGIN
  | RepeatedEventSequence | SimpleEventSequence |
  END

SimpleEventSequence returns to: EventSequence:
  events+=FirstEventOccurrence | events+=EventOccurrence |

Comments

No Comment.

Examples

- the IUT entity being connected and
- the IUT entity having sent an empty request and
- the IUT entity having received a valid response

8.3.2 RepeatedEventSequence

Concrete Textual Notation

RepeatedEventSequence returns to: RepeatedEventSequence:
  'repeat'
  | 'every' interval=LiteralOrDataReferenceAsBinding |
  | repetitions=LiteralOrDataReferenceAsBinding 'times' |
  ?
  BEGIN
  events+=FirstEventOccurrence
  events+=EventOccurrence |
  END

Comments

No Comment.

Examples

Expected behavior ensure that when repeat 5 times the Client entity has sent an empty request then the Client entity shall be blocked
8.3.3 EventOccurrence

Concrete Textual Notation

FirstEventOccurrence returns to EventOccurrence:
  FirstEventOccurrenceSpecification | FirstEventTemplateOccurrence

EventOccurrence returns to EventOccurrence:
  EventOccurrenceSpecification | EventTemplateOccurrence

fragment EventTimingSuffix returns to EventOccurrence:
  With
  BEGIN
  EventTimeLabelFragment?
  EventTimeConstraintFragment?
  END

fragment EventTimeLabelFragment returns to EventOccurrence:
  timeLabel=TimeLabel

fragment EventTimeConstraintFragment returns to EventOccurrence:
  LBrace timeConstraint+=TimeConstraint (',' timeConstraint+=TimeConstraint)* RBrace

Comments
No Comment.

Examples

the IUT entity having sent a connection request

the IUT entity having sent an empty request with
  ts=now
  \{[@ts <= 2]\}

8.3.4 EventOccurrenceSpecification

Concrete Textual Notation

FirstEventOccurrenceSpecification returns to EventOccurrenceSpecification:
  EventOccurrenceSpecificationFragment

EventOccurrenceSpecification returns to EventOccurrenceSpecification:
  ElementAndOrPrefix
  EventOccurrenceSpecificationFragment

fragment EventOccurrenceSpecificationFragment returns to EventOccurrenceSpecification:
  entityReference=EntityReference?
  eventReference=EventReference
  eventArgument=Argument?
  (oppositeEntityReference+=OppositeEntityReference
   (',' oppositeEntityReference+=OppositeEntityReference)*
  )?
  EventTimingSuffix?
  [comment+=Comment]*

Comments
No Comment.
Examples

the IUT entity having sent a booking request

the IUT entity having sent a connection request to the SS entity

8.3.5 EntityReference

Concrete Textual Notation

EntityReference returns to::EntityReference:

EntityReferenceFragment

; OppositeEntityReference returns to::EntityReference:

   comment+=DirectionQualifier

   EntityReferenceFragment

; fragment EntityReferenceFragment returns to::EntityReference:

   comment+=ArticleQualifier

   comment+=Qualifier*

   | [entity=[to::Entity|Identifier]'entity']

   | [component=[to::ComponentInstance|Identifier]'component']

;

Comments

No Comment.

Examples

the UE entity having sent a registration request

the server component having sent a registration request

the client component having sent a registration request to the server component

the client component having sent a registration request to the IUT entity

8.3.6 EventReference

Concrete Textual Notation

EventReference returns to::EventReference:

   [comment+=Qualifier | comment+=CommonWordQualifier | comment+=NotQualifier]*

   event=[to::Event|Identifier]

;

Comments

No Comment.

Examples

having sent
being connected
has been stopped
opening
closing
having closed
8.4 Data

8.4.1 Value

Concrete Textual Notation

Value returns to::Value:
| LiteralValue
| DataReference
| ContentReference
| LiteralValueReference

Argument returns to::Value:
| LiteralValueAsArgument
| DataReferenceAsArgument
| ContentReferenceAsArgument
| LiteralValueReferenceAsArgument

LiteralOrDataReferenceAsBinding returns to::Value:
| LiteralValueAsBinding
| DataReferenceAsBinding

fragment ValueReferenceFragment returns to::Value:
| comment+=NotQualifier?
| comment+=ReferenceQualifier

Comments
No Comment.

Examples
Void.

8.4.2 LiteralValue

Concrete Textual Notation

LiteralValue returns to::LiteralValue:
| comment+=NotQualifier?
| comment+=AssignmentQualifier
| LiteralValueFragment

LiteralValueAsArgument returns to::LiteralValue:
| comment+=ArticleQualifier | comment+=QuantifiedQualifier
| LiteralValueFragment

LiteralValueAsBinding returns to::LiteralValue:
| LiteralValueFragment

fragment LiteralValueFragment returns to::LiteralValue:
| comment+=Qualifier*
| name=Identifier | name=NIdenifier | name=EString
| containing
| BEGIN
| content+=DataContent (',' content+=DataContent)*
| END
|?
Comments

No Comment.

Examples

the IUT entity having sent a booking request containing
date set to today,
days set to 3,
preferences containing
class set to sedan,
transmission set to automatic,
extra insurance not set to desired

8.4.3 Content

Concrete Textual Notation

DataContent returns to::Content:
  comment+=NotQualifier?
  comment+=Qualifier*
    | [name=Identifier | name=NIdentifier]
  | [containing'
    BEGIN
      content+=DataContent | '[' content+=DataContent ']
    END
  | value=Value

Comments

Specification of the optional 'member' property is not supported in the current version of the present document.

Examples

containing
date set to today,
days set to 3,
preferences containing
class set to sedan,
transmission set to automatic,
extra insurance not set to desired

8.4.4 LiteralValueReference

Concrete Textual Notation

LiteralValueReference returns to::LiteralValueReference:
  ValueReference Fragment
  LiteralValueReference Fragment

LiteralValueReference Argument returns to::LiteralValueReference:
  LiteralValueReference Fragment

Binding LiteralValueReference returns to::LiteralValueReference:
  content=[to::LiteralValue | Identifier] | content=[to::LiteralValue | NIdentifier]

fragment LiteralValueReference Fragment returns to::LiteralValueReference:
  "the 'value' of"?
  comment+=ArticleQualifier?
  comment+=Qualifier*
8.4.5 ContentReference

Concrete Textual Notation

```
ContentReference returns to::ContentReference:
  ValueReferenceFragment
  ContentReferenceFragment

ContentReferenceAsArgument returns to::ContentReference:
  ContentReferenceFragment

BindingContentReference returns to::ContentReference:
  content=[to::Content | Identifier]
  fragment ContentReferenceFragment returns to::ContentReference:
    'the' 'value' 'contained' 'in'
    comment+=ArticleQualifier?
    comment+=Qualifier*
    content=[to::Content | Identifier]
```

Comments

No Comment.

Examples

```
the IUT entity having received a booking confirmation containing
days corresponding to the value of the booking request
```

8.4.6 DataReference

Concrete Textual Notation

```
DataReference returns to::DataReference:
  ValueReferenceFragment
  DataReferenceFragment

DataReferenceAsArgument returns to::DataReference:
  [comment+=ArticleQualifier]
  [comment+=QuantifiedQualifier]
  [predefined]'
  DataReferenceFragment
```

Comments

No Comment.

Examples

```
the IUT entity having received a booking confirmation containing
date corresponding to the value contained in the booking request date
```
8.5 Time

8.5.1 TimeLabel

Concrete Textual Notation

Void.

Comments

The concrete syntax is inherited from ETSI ES 203 119-8 [4].

Examples

Void.

8.5.2 TimeConstraint

Concrete Textual Notation

Void.

Comments

The concrete syntax is inherited from ETSI ES 203 119-8 [4].

Examples

Void.
8.6 Event Templates

8.6.1 EventSpecificationTemplate

Concrete Textual Notation

EventSpecificationTemplate returns to EventSpecificationTemplate:
  AnnotationCommentFragment
  "template" name=Identifier
  BEGIN
  eventSpecification=FirstEventOccurrenceSpecification
  END;

Comments

No Comment.

Examples

Template cancellation
  the IUT entity having sent a cancellation request

Template any_job_request
  having sent a job request

8.6.2 EventTemplateOccurrence

Concrete Textual Notation

FirstEventTemplateOccurrence returns to EventTemplateOccurrence:
  EventTemplateOccurrenceFragment

EventTemplateOccurrence returns to EventTemplateOccurrence:
  ElementAndOrPrefix
  EventTemplateOccurrenceFragment

fragment EventTemplateOccurrenceFragment returns to EventTemplateOccurrence:
  'event'
  eventTemplate=[to: EventSpecificationTemplate [Identifier]
  'occurs' | 'with'
  BEGIN
  EventTimeLabelFragment?
  EventTimeConstraintFragment?
  entityBinding+=EntityBinding (',' entityBinding+=EntityBinding)*?
  |'argument' 'replaced' 'by' occurrenceArguments=Argument|?
  |'template' 'replaced' 'by' occurrenceArguments=Argument|?
  END

Comments

No Comment.

Examples

event cancellation occurs

event cancellation occurs with
  the IUT entity replaced by the UE
  argument replaced by a cancellation request containing
8.6.3 EntityBinding

Concrete Textual Notation

EntityBinding returns to EntityBinding:
  templateEntity=EntityReference
  'replaced' 'by'
  occurrenceEntity=EntityReference ;

Comments

No Comment.

Examples

*the IUT entity replaced by the UE entity*

8.7 Variants

8.7.1 StructuredTestObjectiveVariant

Concrete Textual Notation

Variant returns to TestObjectiveVariant:
  'Variant' name=Identifier
  BEGIN
    'Objective': description=EString ?
    'Reference': objectiveURI+=EString ( ',' objectiveURI+=EString )* ?
    'PICS': picsReference+=FirstPICSReference (picsReference+=PICSReference )* ?
    'Bindings'
      BEGIN
        bindings+=VariantBinding ( ',' bindings+=VariantBinding )* ?
      END
    ?
    WithCommentFragment?
  END ;

Comments

No Comment.

Examples

Test Purpose TP_RESOURCE_GET
  Objective: "Read full contents of a resource with an ID"
  Reference: "Clause 4.3.2.4", "Clause 4.3.2.6"
  Expected behaviour ensure that
    when
      the Server entity receives a vGET request containing
      uri indicating value '/resource/' ,
      id set to ID
    then
      the Server entity sends a HTTP response containing
      status set to HTTP_STATUS
  Variant TP_RESOURCE_GET_200v1
    Objective: "Read full contents of a resource with a valid ID"
    Bindings
      value ID set to VALID_ID,
      value HTTP_STATUS set to 200 OK
Variant TP_RESOURCE_GET_404v2

Objective: "Read contents of a resource with a non-existent ID returns 404"

Bindings
value ID set to NONEXISTENT_ID,
value HTTP_STATUS set to 404 Not found

8.7.2 Variants

Concrete Textual Notation

Variants returns to::Variants:

variants+=Variant+
WithCommentFragment?

Comments
No Comment.

Examples
Void.

8.7.3 VariantBinding

Concrete Textual Notation

VariantBinding returns to::VariantBinding:

VariantBindingValue

VariantBindingAttribute

VariantBindingPredefined

VariantBindingValue returns to::VariantBinding:

'value' value=BindingLiteralValueReference
comment+=AssignmentQualifier
boundTo=LiteralOrDataReferenceAsBinding
WithCommentFragment?

VariantBindingAttribute returns to::VariantBinding:

'attribute' value=BindingContentReference
comment+=AssignmentQualifier
boundTo=LiteralOrDataReferenceAsBinding
WithCommentFragment?

VariantBindingPredefined returns to::VariantBinding:

'predefined' 'value' value=BindingDataReference
comment+=AssignmentQualifier
boundTo=DataReferenceAsBinding
WithCommentFragment?

Comments
No Comment.

Examples
Void.
8.8 Behaviour

8.8.1 TestDescription

Concrete Textual Notation

```plaintext
TestPurposeDescription returns tdl::TestDescription:
  annotation += TestPurposeDescriptionAnnotation
  name = Identifier
  [LParen formalParameter += FormalParameter *, FormalParameter +] RParen ?
BEGIN
  TDObjectiveFragment? 'Configuration:' testConfiguration=
  (tdl::TestConfiguration | Identifier)
  (behaviourDescription = TPDBehaviourDescription)
END;
```

Comments

The alternative derivation is used to enable 'TestDescription's to be used as containers for the specification of test purposes. The alternative derivation provides a structure comprising a 'CompoundBehaviour' containing the 'Block's and 'Behaviour's annotated with pre-defined 'AnnotationType'. The overall structure resembles a 'StructuredTestObjective' while the actual contents are 'Behaviour's rather than 'EventOccurrence's. The semantics and constraints for 'TestDescription's still apply. The pre-defined 'Annotation's may be used to indicate potential incompleteness in the specification. In contrast to 'StructuredTestObjective's, it is not possible to reference 'PICS' and 'TestObjective's need to defined separately and can be referenced. 'TestDescription's specified with the alternative derivation can be represented by means of the inherited derivation, the opposite is not necessarily true.

Examples

```
Test Purpose Description TPD1
  Objective: TPD_Illustration
  Configuration: basic
  Initial conditions
  with
    client::sg sends "ring, ring" to server::sg
  Expected behaviour
  ensure that
    when
      client::sg sends "hi" to server::sg
        client::sg sends "how are you?" to server::sg
        then
          server::sg sends "hi back" to client::sg
          server::sg sends "we are fine" to client::sg
    Final conditions
  with
    client::sg sends "ok, thx, bye" to server::sg
```

8.8.2 BehaviourDescription

Concrete Textual Notation

```plaintext
TPDBehaviourDescription returns tdl::BehaviourDescription:
  behaviour = TPDCcompoundBehaviour
```

Comments

The alternative derivation is used in the context of 'TestDescription's used as containers for the specification of test purposes. Only a 'CompoundBehaviour' containing the 'Block' and 'Behaviour's annotated with pre-defined 'AnnotationType' is allowed in this context.
Examples
Void.

8.8.3 CompoundBehaviour

Concrete Textual Notation

```
TPD CompoundBehaviour returns tdl::CompoundBehaviour:
    block = TPDBlock
;
InitialConditionsBehaviour returns tdl::CompoundBehaviour:
    annotation += InitialConditionsAnnotation
    'with' block = Block
;
ExpectedBehaviourBehaviour returns tdl::CompoundBehaviour:
    annotation += ExpectedBehaviourAnnotation
    'ensure' 'that'
    | BEGIN
    |   block = WhenThenBlock
    | END
| block = Block
;
FinalConditionsBehaviour returns tdl::CompoundBehaviour:
    annotation += FinalConditionsAnnotation
    'with' block = Block
;
WhenBehaviour returns tdl::CompoundBehaviour:
    annotation += WhenAnnotation
    block = Block
;
ThenBehaviour returns tdl::CompoundBehaviour:
    annotation += ThenAnnotation
    block = Block
;
```

Comments

The alternative derivations are used in the context of 'TestDescription's used as containers for the specification of test purposes. The 'CompoundBehaviour's annotatated with the pre-defined 'AnnotationType's outline the structure of the 'TestDescription'. The inherited derivations for 'CompoundBehaviour' may be used within the 'Block's inside the outlined structure.

Examples
Void.

8.8.4 Block

Concrete Textual Notation

```
TPDBlock returns tdl::Block:
    | behaviour += InitialConditionsBehaviour?
    | behaviour += ExpectedBehaviourBehaviour?
    | behaviour += FinalConditionsBehaviour?
;
```
WhenThenBlock returns tdl::Block:
    behaviour += WhenBehaviour
    behaviour += ThenBehaviour

Comments

The alternative derivations are used in the context of TestDescription's used as containers for the specification of test purposes. The 'guard' property is syntactically excluded from the alternative derivations.

Examples

Void.
Annex A (informative):
Examples

A.0 Overview

This annex provides several examples to illustrate how the different elements of the Textual Syntax for the Structured Test Objective Specification extension of TDL can be used and demonstrates the applicability of the extension in different areas. The examples showcase the indentation-based textual syntax variant.

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 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.

```plaintext
Package Example3GPP

Entity UE
Event in
Event sends
Event receives
Event performs
Event send

Test Purpose TP_7.1.3.1.1
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_7.1.3.1.2
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 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 ETSITS 186 011-2 [i.3] into the 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.

```plaintext
Package ExampleIMS
  Import all from Standard
  Import all from TO

// 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

Entity UE_A
Entity UE_B
Entity IMS_B
Event sends
Event receives

Test Purpose TP_IMS_4002_1
  Objective: "
  Reference: 'ETSITS 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 1300 bytes
    to the UE_B entity
  then
    the IMS_B entity receives the MESSAGE containing
      Message_Body_Size indicating value greater than 1300 bytes
```
Annex B (informative):
Examples in Legacy Textual Syntax

B.0 Overview

This annex provides several examples to illustrate how the different elements of the Structured Test Objective Specification extension of TDL can be used with the legacy informative textual syntax.

The specification of the legacy textual syntax for the additional concepts in the Structured Test Objective extension as well as minimal set of required TDL concepts to facilitate the specification and representation of StructuredTestObjective's can be found in the TDL Open Source Project (TOP) [i.4]. 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.

NOTE: This annex is deprecated and will be removed in future editions of the present document in favour of the standardised textual syntax in ETSI ES 203 119-8 [4] and the extensions to it specified in clause 8 of the present document. The latest specification of the legacy textual syntax is available in the TDL Open Source Project (TOP) [i.4].

B.1 A 3GPP Test Objective in Legacy 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 legacy 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"
}
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"
  }
}

B.2 An IMS Test Objective in Legacy 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 legacy 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 C (informative):
Legacy Textual Syntax BNF Production Rules

C.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.

NOTE: This annex is deprecated and will be removed in future editions of the present document in favour of the standardised textual syntax in ETSI ES 203 119-8 [4] and the extensions in clause 8 of the present document. The latest specification of the legacy textual syntax is available in the TDL Open Source Project (TOP) [i.4].

C.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 C.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>

Table C.1: Syntax definition conventions used

C.2 Production Rules

<table>
<thead>
<tr>
<th>Package</th>
<th>::=</th>
<th>'Package' Identifier '{'</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>}</td>
<td>'ElementImport')</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ 'Domain' '{'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ 'pics' ':' [ PICS ]+ ';' ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ 'entity' 'types' ':' [ EntityType ]+ ';' ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ 'entities' ':' [ Entity ]+ ';' ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ 'event' 'types' ':' [ EventType ]+ ';' ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ 'events' ':' [ Event ]+ ';' ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ 'event' 'templates' ':'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ EventOccurrenceTemplate ]+ ';' ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ 'Data' ':'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ DataElement ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ 'Configuration' ':'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ GateType ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ ComponentType ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ TestConfiguration ] ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ StructuredTestObjective ]</td>
</tr>
<tr>
<td></td>
<td>]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DataElement</th>
<th>::=</th>
<th>DataType</th>
<th>DataInstance</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DataType</th>
<th>::=</th>
<th>SimpleDataType</th>
<th>StructuredDataType</th>
</tr>
</thead>
</table>

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

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

PICS ::= 'PICS' Identifier [ ( 'Qualifier' ) ] ;
FirstPICSReference ::= [ NotQualifier ] Identifier ;
PICSReference ::= [ AndOrQualifier ] [ NotQualifier ] Identifier ;
EntityType ::= "EntityType" Identifier [ ( 'Annotation' { ',' 'Annotation' } ) ] ;
Event ::= "Event" Identifier ;
Annotation ::= Identifier ;
EventOccurrenceTemplate ::= 'EventOccurrenceTemplate' Identifier [ 'EventSpecification' ] ;
EventSpecification ::= EntityReference EventReference Argument
StructuredTestObjective ::= 'StructuredTestObjective' '{' 'TP Id' Identifier ['Test objective' Identifier ] ['Reference' Identifier ] ['Config Id' Identifier ] ['PICS Selection' FirstPICSReference [ PICSReference ] ] [ InitialConditions ] [ ExpectedBehaviour ] [ FinalConditions ] '}' ;
InitialConditions ::= "Initial conditions'
'with' '{' EventSequence '}' ;
ExpectedBehaviour ::= 'Expected behaviour'
'ensure that' '{' 'when' '{' EventSequence '}' 'then' '{' EventSequence '}' '}' ;
PartialExpectedBehaviour ::= 'Expected behaviour'
'ensure that' '{' EventSequence '}' ;
FinalConditions ::= "Final conditions'
'with' '{' EventSequence '}' ;
EventSequence ::= RepeatedEventSequence | SimpleEventSequence ;
SimpleEventSequence ::= FirstEventOccurrence { EventOccurrence } ;
RepeatedEventSequence ::= 'repeat' [ ( 'every' | IterationValue ) | ( IterationValue | 'times' ) ] '{' FirstEventOccurrence { EventOccurrence } '}' ;
FirstEventOccurrence ::= FirstEventOccurrenceSpecification | FirstEventTemplateOccurrence ;
EventOccurrence ::= EventOccurrenceSpecification | EventTemplateOccurrence ;
FirstEventOccurrenceSpecification ::= [ ( TimeLabel | ( ( ',' TimeConstraint ) | ':' ) ) TimeConstraint ]
EntityReference EventReference Argument
StructuredTestObjective ::= [ ( Note ) ];
FirstEventTemplateOccurrence ::= [ TimeLabel | ( ( ',' TimeConstraint ) | ':' ) ]
'event' Identifier 'occurs'
'with' [ ( EntityBinding [ ( 'argument' 'replaced' 'by' Argument ] ) ] [ Note ] ;
EntityBinding ::= EntityReference 'replaced' 'by' EntityReference ;
EventOccurrenceSpecification ::= AndOrQualifier [ ( TimeLabel | ( ( ',' TimeConstraint ) | ':' ) ) TimeConstraint ]
EntityReference EventReference
Argument
  [ OppositeEntityReference
    [ ',' OppositeEntityReference ] ]
  ( Note ) ;
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 ::= ConstraintTimeLabelUse | ConstraintDataInstanceUse ;
ConstraintDataInstanceUse ::= Identifier | NumberAsIdentifier ;
ConstraintTimeLabelUse ::= Identifier ;
TimeConstraintQualifier ::= { 'before' | 'after' | 'during' | 'within' ) ;
EntityReference ::= ArticleQualifier
  ( Qualifier )
  [ ( Identifier 'entity' )
    | ( Identifier 'component' ) ] ;
OppositeEntityReference ::= DirectionQualifier
  ArticleQualifier
  ( Qualifier )
  [ ( Identifier 'entity' )
    | ( Identifier 'component' ) ] ;
EventReference ::= { Qualifier | CommonWordQualifier | NotQualifier }
  Identifier ;
Argument ::= LiteralValueAsArgument
  TypedLiteralValueAsArgument
  Value
  DataReference
  ContentReference
  LiteralValueReferenceArgument ;
Value ::= LiteralValue
  DataReference
  ContentReference
  LiteralValueReference ;
TypedValue ::= TypedLiteralValue
  DataReference
  ContentReference
  LiteralValueReference ;
IterationValue ::= IterationLiteralValue | IterationDataReference ;
TypedLiteralValueAsArgument ::= ( ArticleQualifier | QuantifiedQualifier )
  'typed'
  Identifier
  [ 'containing'
    TypedDataContent { ',' TypedDataContent } ':' ] ;
TypedLiteralValue ::= [ NotQualifier ]
  AssignmentQualifier
  ( Qualifier )
  Identifier
  [ 'containing'
    TypedDataContent { ',' TypedDataContent } ':' ] ;
TypedDataContent ::= [ NotQualifier ]
  [ AssignmentQualifier
    ( Qualifier )
    Identifier
    [ 'containing'
      TypedDataContent { ',' TypedDataContent } ':' ]
    TypedList ] ;
LiteralValueAsArgument ::= ( ArticleQualifier | QuantifiedQualifier )
  ( Qualifier )
  Identifier
  [ 'containing'
    DataContent { ',' DataContent } ':' ] ;
LiteralValue ::= [ NotQualifier ]
  AssignmentQualifier
  ( Qualifier )
  Identifier
  [ 'containing'
    DataContent { ',' DataContent } ':' ] ;
IterationLiteralValue ::= IterationDataInstanceUse ;
IterationDataReference ::= IterationDataInstanceUse ;
DataContent ::= [ NotQualifier ]
  [ AssignmentQualifier
    ( Qualifier )
    Identifier
    [ 'containing'
      DataContent { ',' DataContent } ':' ]
    TypedList ] ;
( 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';
 NotQualifier ::= 'not';
 AndOrQualifier ::= 'and' | 'or';
 DirectionQualifier ::= 'by' | 'in' | 'into' | 'for' | '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
[ NotQualifier ] AssignmentQualifier
( Qualifier )
 StaticDataUse;
 ContentReference ::= [ NotQualifier ] ReferenceQualifier
'the' 'value' 'contained in'
( Qualifier )
 Identifier;
 LiteralValueReference ::= [ NotQualifier ]
ReferenceQualifier
'of'
( Qualifier )
 Identifier;
 ContentReferenceAsArgument ::= 'the' 'value' 'of'
( Qualifier )
 Identifier;
 LiteralValueReferenceAsArgument ::= 'the' 'value' 'of'
( Qualifier )
 Identifier;
 DataReference ::= Identifier
[ NotQualifier ] ReferenceQualifier
( Qualifier )
 StaticDataUse;
 DataReferenceAsArgument ::= ( ArticleQualifier | QuantifiedQualifier )
'(predefined)'
( Qualifier )
 StaticDataUse;
 NumberAsIdentifier ::= ['-'] INT [ '.' INT ];
SimpleDataType ::= 'type' Identifier ';' ;
StructuredDataType ::= 'type' Identifier

  'with' Member [ ',', Member ] ';' ;
Member ::= [ Optional ] Identifier 'of' 'type' Identifier ;
Optional ::= 'optional' ;
SimpleDataInstance ::= Identifier

  ( Identifier | NumberAsIdentifier ) ';' ;
StructuredDataInstance ::= Identifier

  ( Identifier | NumberAsIdentifier )

  'containing' MemberAssignment { ',', MemberAssignment } ';' ;
MemberAssignment ::= Identifier [ NotQualifier ]

  AssignmentQualifier StaticDataUse ;
TestConfiguration ::= 'Test Configuration'

  Identifier

  'containing' ComponentInstance [ ComponentInstance ]
Connection [ Connection ] ';' ;
ComponentInstance ::= ComponentInstanceRole

  'component' Identifier 'of' 'type' Identifier ;
Connection ::= 'connection' 'between'

  GateReference '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 ) ]

  | SQ ) ;
ML_COMMENT ::= (/' */ *'/ ) ;
SL_COMMENT ::= ( '/' ( '\n' | '\r' ) [ [ '/\r' ] '\n' ] ) ;
WS ::= ( ' '

  \s

  '\n' ) ;

ETS
## History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.1.1</td>
<td>June 2015</td>
<td>Publication</td>
</tr>
<tr>
<td>V1.2.1</td>
<td>September 2016</td>
<td>Publication</td>
</tr>
<tr>
<td>V1.3.1</td>
<td>May 2018</td>
<td>Publication</td>
</tr>
<tr>
<td>V1.4.1</td>
<td>August 2020</td>
<td>Publication</td>
</tr>
<tr>
<td>V1.5.1</td>
<td>March 2022</td>
<td>Membership Approval Procedure MV 20220527: 2022-03-28 to 2022-05-27</td>
</tr>
<tr>
<td>V1.5.1</td>
<td>May 2022</td>
<td>Publication</td>
</tr>
</tbody>
</table>