ETSI ES 203 119-7 V1.3.1 (2022-05)

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 7: Extended Test Configurations

Reference
RES/MTS--TDL1197v131

Keywords
language, MBT, methodology, testing

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights 5
Foreword 5
Modal verbs terminology 5
Introduction 5
1 Scope 6
2 References 6
2.1 Normative references 6
2.2 Informative references 6
3 Definition of terms, symbols and abbreviations 7
3.1 Terms 7
3.2 Symbols 7
3.3 Abbreviations 7
4 Basic Principles 7
4.1 Extended Test Configurations 7
4.2 Document Structure 8
4.3 Notational Conventions 8
4.4 Element Operations 8
4.5 Conformance 8
5 Meta-Model Extensions 8
5.1 Overview 8
5.2 ExtendedTestConfiguration 9
5.3 TestConfigurationInstance 9
5.4 TestConfigurationOperation 10
5.5 ComponentReference 10
5.6 ExtendedGateReference 11
5.7 ComponentMerge 11
5.8 ComponentAlias 12
5.9 ComponentHide 12
5.10 ReassignRole 12
6 Graphical Syntax Extensions 13
6.1 ExtendedTestConfiguration 13
6.2 TestConfigurationInstance 13
6.3 TestConfigurationOperation 14
6.4 ComponentReference 14
6.5 ComponentMerge 14
6.6 ComponentAlias 15
6.7 ComponentHide 15
6.8 ReassignRole 15
7 Exchange Format Extensions 15
8 Textual Syntax Extensions 16
8.1 ExtendedTestConfiguration 16
8.2 TestConfigurationInstance 16
8.3 TestConfigurationOperation 17
8.4 ComponentReference 17
8.5 ExtendedGateReference 17
8.6 ComponentMerge 18
8.7 ComponentAlias 18
8.8 ComponentHide 18
8.9 ReassignRole 19
8.10 Connection 19

Annex A (informative): Examples.....20
A.0 Overview.....20
A.1 Test Configuration Instantiation.....20
A.2 Test Configuration Operations.....20
A.3 Component Merging.....22
History.....23

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 7 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

This extension package to TDL introduces additional features for the specification of extended test configurations by reusing existing test configurations. Existing test configurations can be instantiated within an extended test configuration. By means of test configuration operations, the test configuration instances can be modified within an extended test configuration, without affecting the original test configuration specification that is instantiated.

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

1 Scope

The present document defines extensions to the Test Description Language (TDL) to support the re-use of test configurations.

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.

[1] ETSI ES 203 119-1 (V1.6.1): "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 1: Abstract Syntax and Associated Semantics".

[2] ETSI ES 203 119-2 (V1.5.1): "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 2: Graphical Syntax".

[3] ETSI ES 203 119-3 (V1.5.1): "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 3: Exchange Format".

[4] ETSI ES 203 119-8 (V1.1.1): "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 8: Textual Syntax".

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.

Not applicable.

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:

component reference: reference to a unique component instance in an extended test configuration

extended gate reference: extension to gate reference that makes it possible to specify gate references from different component instances in a unique manner within an extended test configuration

extended test configuration: specification of a test configuration which includes a set test configuration instances and test configuration operations, as well as additional component instnaces and connections

flattened test configuration: test configuration resulting from the transformation of an extended test configuration into a test configuration that includes all the component instances and connections from the instantiated test configurations after applying the test configuration operations, as well as additional component instances and connections defined within the extended test configuration

test configuration instance: instantiation of an existing test configuration

test configuration operation: operation on a component instance in an extended test configuration

3.2 Symbols

Void.

3.3 Abbreviations

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

OCL™	Object Constraint Language
SUT	System Under Test
TDL	Test Description Language

4 Basic Principles

4.1 Extended Test Configurations

Re-use of test configurations with the capability to modify a test configuration as part of the re-use is an essential feature for managing larger test specifications in TDL. This extension for the specification of extended test configurations in TDL provides the necessary capabilities for instantiating existing test configuration within an extended test configuration, as well as modifying the instantiated test configurations by means of test configuration operations. Extended test configurations are intented for higher-level specification of reusable test configurations. An extended test configuration shall be transformed into a "*flattened*" test configuration in order to be used in a test description. The flattened test configuration shall contain all the component instances and connections from the instantiated test configurations after applying the test configuration operations, as well as additional component instances and connections defined within the extended test configuration.

4.2 Document Structure

The present document defines the composite test configuration extensions for TDL comprising:

- Meta-model extensions describing additional concepts required for the specification of extended test configurations (clause 5).
- Concrete syntax extension describing corresponding shapes for the representation of the additional concepts (clause 6).
- An informative annex with examples (annex A).

4.3 Notational Conventions

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

The abstract syntax specification and the classifier descriptions follow the notational conventions defined in clause 4.5 of Abstract Syntax and Associated Semantics [1]. The concrete grahical syntax notation specification follows the notational conventions described in clause 4.5 of the Graphical Syntax [2]. The concrete textual notation follows the notational conventions described in clause 4.3 of the Textual Syntax [4].

4.4 Element Operations

The formalized constraints for the present document rely on operations provided by the standard library of OCL and in ETSI ES 203 119-1 [1].

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 extended test configuration concepts are defined within a single package in the TDL meta-model. The additional concepts are "self-contained" in that a specification that relies on them shall be transformed into a test configuration that does not make any use of the additional concepts before using the test configuration in a test description.

5.2 ExtendedTestConfiguration

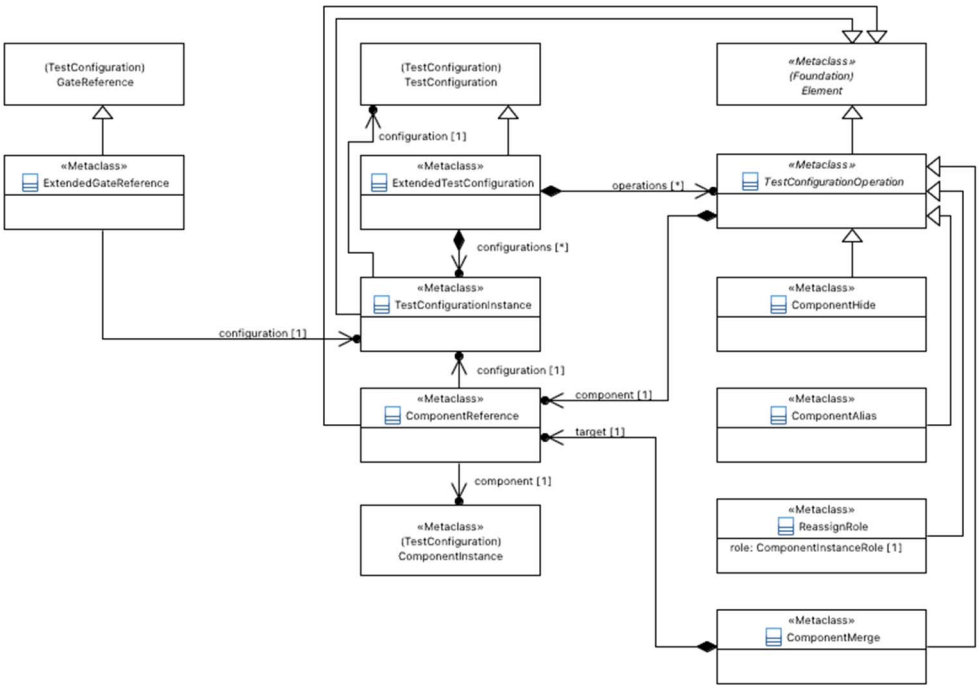


Figure 5.2.1: Extended test configuration specification concepts

Semantics

An 'ExtendedTestConfiguration' is a refinement of 'TestConfiguration' that contains the 'TestConfigurationInstance's and 'TestConfigurationOperation's enabling the reuse of existing 'TestConfiguration's. The 'TestConfigurationOperation's shall be applied in the specified order.

Generalization

- TestConfiguration.

Properties

- configurations: TestConfigurationInstance [0..*]
  The instantiated 'TestConfiguration's.

- operations: TestConfigurationsOperation [0..*]
  The 'TestConfigurationOperation's for the refinement of the instantiated 'TestConfiguration's.

Constraints

- There are no constraints specified.

5.3 TestConfigurationInstance

Semantics

A 'TestConfigurationInstance' represents an instantiation of an existing 'TestConfiguration' All 'ComponentInstance's and 'Connection's of the instantiated 'TestConfiguration' shall be replicated.

Generalization

- Element.

Properties

- configuration: TestConfiguration [1]
  A reference to the instantiated 'TestConfiguration'.

Constraints

- There are no constraints specified.

5.4 TestConfigurationOperation

Semantics

An abstract super-class for any concrete operation on 'ComponentInstances' within an 'ExtendedTestConfiguration'.

Generalization

- Element.

Properties

- component: ComponentReference ([1])
  A reference to the 'ComponentInstance' on which the operation shall be applied.

Constraints

- There are no constraints specified.

5.5 ComponentReference

Semantics

A 'ComponentReference' is a target of a 'TestConfigurationOperation'. It allows 'ComponentInstance's within an 'ExtendedTestConfiguration' to be referenced in unique manner, where multiple instances of the same 'TestConfiguration' would otherwise create ambiguity.

Generalization

- Element.

Properties

- component: ComponentInstance [1]
  The 'ComponentInstance' that the 'ComponentReference' refers to.
- configuration: TestConfigurationReference [0..1]
  The 'TestConfigurationInstance' that the 'ComponentReference' refers to.

Constraints

- There are no constraints specified.

5.6 ExtendedGateReference

Semantics

An extension to 'GateReference' enabling the specification of 'GateReferences' of different 'ComponentInstance's in different 'TestConfigurationInstance's in a unique manner.

Generalization

- GateReference.

Properties

- configuration: TestConfigurationReference [0..1]
  The 'TestConfigurationInstance' that the 'ExtendedGateReference' refers to.

Constraints

- There are no constraints specified.

5.7 ComponentMerge

Semantics

A 'ComponentMerge' enables two 'ComponentInstance's of the same 'ComponentType' to be merged into one where the target 'ComponentInstance' shall inherit the 'Connection's of the source 'ComponentInstance' specified by means of the 'component' property, while keeping the role of the target 'ComponentInstance'.

Generalization

- TestConfigurationOperation.

Properties

- target: ComponentReference [1]
  A reference to the target 'ComponentInstance' which the 'ComponentInstance' shall be merged into.

Constraints

- **No self-merging**
  A 'ComponentInstance' shall not be merged with itself, i.e. the source and target 'ComponentInstance's specified by means of the 'ComponentReference's shall be different.
  inv: **NoSelfMerge**:

  ```
  not (self.component.component = self.target.component and
      self.component.configuration = self.target.configuration)
  ```

- **Conforming 'ComponentType's**
  The 'ComponentInstance' specified by means of the target 'ComponentReference's shall have a 'ComponentType' which conforms to the 'ComponentType' of the source 'ComponentReference'.
  inv: **ComponentMergeType**:

  ```
  self.target.component.type.conformsTo(self.component.component.type)
  ```

5.8 ComponentAlias

Semantics

A 'ComponentAlias' is a 'TestConfigurationOperation' that enables 'ComponentInstance' from an instantiated 'TestConfiguration' to be renamed.

Generalization

- TestConfigurationOperation.

Properties

- There are no properties specified.

Constraints

- **Mandatory name**
  The 'name' property of the 'ComponentAlias' shall be set and it shall not be an empty String.
  inv: **AliasMandatoryName**:

  not self.name.oclIsUndefined() and self.name.size() > 0

5.9 ComponentHide

Semantics

A 'ComponentHide' is a 'TestConfigurationOperation' enabling the hiding of a 'ComponentInstance' from an instantiated 'TestConfiguration'.

Generalization

- TestConfigurationOperation.

Properties

- There are no properties specified.

Constraints

There are no constraints specified.

5.10 ReassignRole

Semantics

A 'ReassignRole' is a 'TestConfigurationOperation' that enables the re-assignment of the role of a 'ComponentInstance' from an instantiated 'TestConfiguration'.

Generalization

- TestConfigurationOperation.

Properties

- role: ComponentInstanceRole [1]
  The new role of the referenced 'ComponentInstance'.

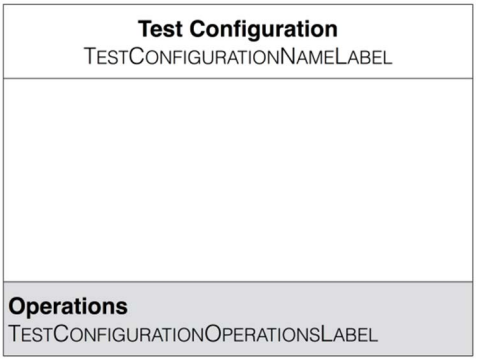
Constraints

- There are no constraints specified.

6 Graphical Syntax Extensions

6.1 ExtendedTestConfiguration

Concrete Graphical Notation



Formal Description

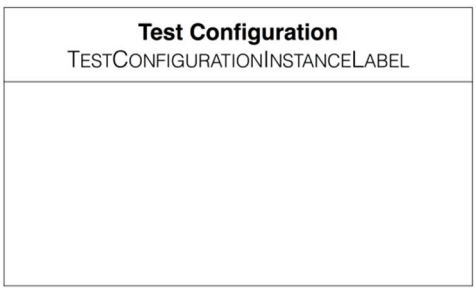
context ExtendedTestConfiguration
c ::=self.name
TESTCONFIGURATIONOPERATIONSLABEL ::= **foreach** o:TestConfigurationOperation **in** self.operations
o **as context in** <TESTCONFIGURATIONOPERATIONLABEL>
end

Comments

The elements of the *ExtendedTestConfiguration* shall be placed into the middle empty compartment. The compartment containing **Operations** is optional (that is it can be omitted). If the optional compartment is present, its content shall also be present.

6.2 TestConfigurationInstance

Concrete Graphical Notation



Formal Description

context ExtendedTestConfiguration
TESTCONFIGURATIONINSTANCENAMELABEL ::= self.name ':' self.configuration.name

Comments

The elements of the instantiated *TestConfiguration* shall be placed into the lower empty compartment.

6.3 TestConfigurationOperation

Concrete Graphical Notation

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

Formal Description

context TestConfigurationOperation

TESTCONFIGURATIONOPERATIONLABEL ::= **if** self**.oclIsTypeOf(**ComponentHide**) then** self **as context in** <COMPONENTHIDELABEL>
else if self**.oclIsTypeOf(**ComponentAlias**) then** self **as context in** <COMPONENTALIASLABEL>
else if self**.oclIsTypeOf(**ReassignRole**) then** self **as context in** <REASSIGNROLELABEL>
else if self**.oclIsTypeOf(**ComponentMerge**) then** self **as context in** <COMPONENTMERGELABEL>
endif

Comments

No comments.

6.4 ComponentReference

Concrete Graphical Notation

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

Formal Description

context ComponentReference

COMPONENTREFERENCELABEL ::= self.configuration.name '.' self. component.name

Comments

No comments.

6.5 ComponentMerge

Concrete Graphical Notation

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

Formal Description

context ComponentMerge

COMPONENTMERGELABEL ::= 'merge' self.component **as context in** <COMPONENTREFERENCELABEL>
'into' self.target **as context in** <COMPONENTREFERENCELABEL>

Comments

No comments.

6.6 ComponentAlias

Concrete Graphical Notation

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

Formal Description

context ComponentAlias

COMPONENTALIASLABEL ::= 'rename' self.component **as context in** <COMPONENTREFERENCELABEL> 'to' self.name

Comments

No comments.

6.7 ComponentHide

Concrete Graphical Notation

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

Formal Description

context ComponentHide

COMPONENTHIDELABEL ::= 'hide' self.component **as context in** <COMPONENTREFERENCELABEL>

Comments

No comments.

6.8 ReassignRole

Concrete Graphical Notation

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

Formal Description

context ReassignRole

REASSIGNROLELABEL ::= 'reassign' self.component **as context in** <COMPONENTREFERENCELABEL> 'to' self **as context in** <COMPONENTROLELABEL>

COMPONENTROLELABEL ::= **if** self.role = ComponentInstanceRole::SUT **then** 'SUT' **else** 'Tester' **endif**

Comments

No comments.

7 Exchange Format Extensions

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

8 Textual Syntax Extensions

8.1 ExtendedTestConfiguration

Concrete Textual Notation

```
ExtendedTestConfiguration returns tc::ExtendedTestConfiguration:
  AnnotationCommentFragment
  'Configuration' name=Identifier
  'with'
  BEGIN
    configurations+=TestConfigurationInstance (',' configurations+=TestConfigurationInstance)*
    (',' operations+=TestConfigurationOperation)*
    (',' componentInstance+=ComponentInstance)*
    (',' connection+=ExtendedConnection)*
  END
;
```

Comments

No Comment.

Examples

```
Configuration ClientServerInIMS with
  //instantiated test configurations
  BasicClientServer as CS,
  IMSEndToEnd as IMS,

  //test configuration operations
  CS::client as SUT,
  CS::server as proxyServer,
  IMS::IMS_B as Tester,
  IMS::IBCF_A merged into IMS::IBCF_A,
  IMS::USER_A hidden,

  //additional components
  node FW_A as Tester,
  node FW_B as Tester,

  //additional connections
  connect CS::client::sg to IMS::UE_A::g,
  connect CS::server::sg to IMS::UE_B::g,
  connect CS::client::sg to FW_A::g,
  connect CS::server::sg to FW_B::g
```

8.2 TestConfigurationInstance

Concrete Textual Notation

```
TestConfigurationInstance returns tc::TestConfigurationInstance:
  AnnotationCommentFragment
  configuration=[tdl::TestConfiguration|Identifier]
  'as' name=Identifier
;
```

Comments

No Comment.

Examples

```
  BasicClientServer as c1
```

```
    IMSEndToEnd as c2
```

8.3 TestConfigurationOperation

Concrete Textual Notation

```
TestConfigurationOperation returns tc::TestConfigurationOperation:
  ComponentMerge
  | ComponentHide
  | ReassignRole
  | ComponentAlias
;
```

Comments

No Comment.

Examples

Void.

8.4 ComponentReference

Concrete Textual Notation

```
ComponentReference returns tc::ComponentReference:
  configuration=[tc::TestConfigurationInstance]
  '::' component=[tdl::ComponentInstance|Identifier]
;
```

Comments

No Comment.

Examples

```
    c1::server
    c2::firewall
    c2::client
```

8.5 ExtendedGateReference

Concrete Textual Notation

```
ExtendedGateReference returns tc::ExtendedGateReference:
  (name=GRIdentifier '=')?
  configuration=[tc::TestConfigurationInstance|Identifier]
  '::'
  component=[tdl::ComponentInstance|Identifier]
  '::'
  gate=[tdl::GateInstance|Identifier]
;
```

Comments

It is recommended to use named 'ExtendedGateReference's for better readability and compatibility.

Examples

```
    cs=c1::server::sg
    fw=c2::firewall::sg
```

c2::client::sg

8.6 ComponentMerge

Concrete Textual Notation

```
ComponentMerge returns tc::ComponentMerge:
  AnnotationCommentFragment
  component=ComponentReference
  'merged' 'into' target=ComponentReference
;
```

Comments

No Comment.

Examples

c1::client **merged into** c2::client

8.7 ComponentAlias

Concrete Textual Notation

```
ComponentAlias returns tc::ComponentAlias:
  AnnotationCommentFragment
  component=ComponentReference
  'as' name=Identifier
;
```

Comments

No Comment.

Examples

c1::server **as** proxyServer

8.8 ComponentHide

Concrete Textual Notation

```
ComponentHide returns tc::ComponentHide:
  AnnotationCommentFragment
  component=ComponentReference 'hidden'
;
```

Comments

No Comment.

Examples

c2::server **hidden**

8.9 ReassignRole

Concrete Textual Notation

```
ReassignRole returns tc::ReassignRole:
  AnnotationCommentFragment
  component=ComponentReference
  'as' role=ComponentInstanceRole
;
```

Comments

No Comment.

Examples

```
c1::client as SUT
c2::client as Tester
```

8.10 Connection

Concrete Textual Notation

```
ExtendedConnection returns tdl::Connection:
  AnnotationCommentFragment
  'connect'
  endPoint+=(GateReference | ExtendedGateReference)
  'to' endPoint+=(GateReference | ExtendedGateReference)
  WithNameFragment?
;
```

Comments

The alternative derivation is used in the respective context of 'ExtendedTestConfiguration's. The derivation includes 'ExtendedGateReference's as 'endPoint's of 'Connection's.

Examples

```
connect proxy::sg to firewall::sg
connect cs=c1::server::sg to fw=firewall::sg
connect c1::server::sg to c2::client::sg
```

Annex A (informative): Examples

A.0 Overview

This annex provides several examples to illustrate the use of extended test configurations by means of the graphical syntax. The first example in clause A.1 illustrates the instantiation of an existing test configuration. The second example in clause A.2 illustrates the application of the component hide, role reassignment, and component alias operations. The third example in clause A.3 illustrates the application of the component merge operation.

A.1 Test Configuration Instantiation

In this example, an example test configuration 'defaultTC' which will be instantiated and reused multiple times subsequently is shown in Figure A.1.1. An extended test configuration 'compositeTC' which features two instances 'source' and 'target' of the test configuration 'defaultTC' and the resulting test configuration after the flattening transformation are illustrated in Figure A.1.2.

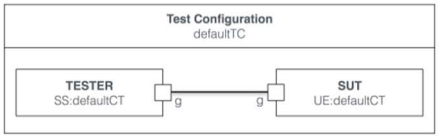


Figure A.1.1: An example test configuration which will be reused

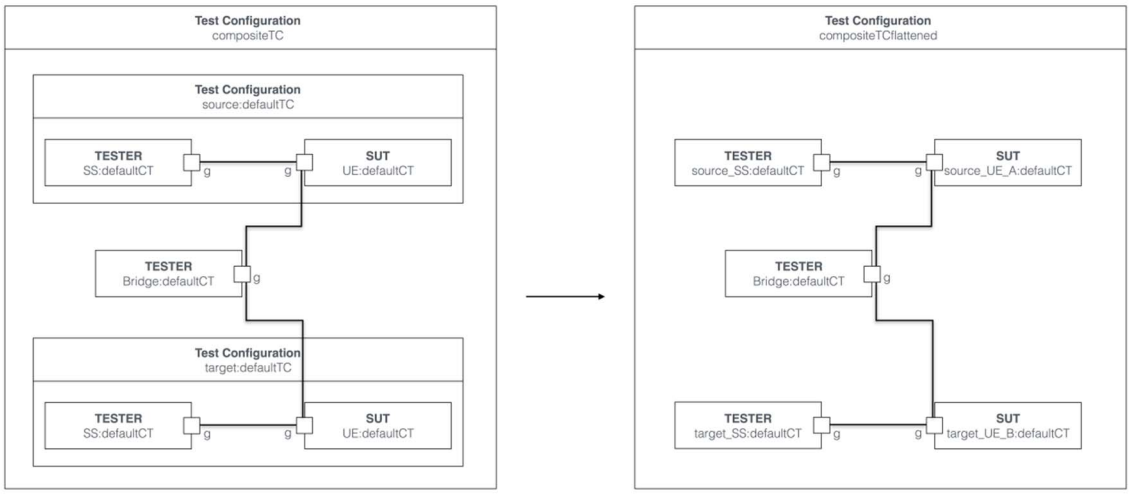


Figure A.1.2: Extended test configuration with test configuration instances and flattening

A.2 Test Configuration Operations

In this example, the extended test configuration 'compositeTC' from Figure A.1.2 is refined further by applying the component hide, component alias and role reassignment operations. The extended test configuration resulting from the application of the test configuration operations is illustrated in Figure A.2.1. The corresponding test configuration after the flattening transformation is illustrated in Figure A.2.2.

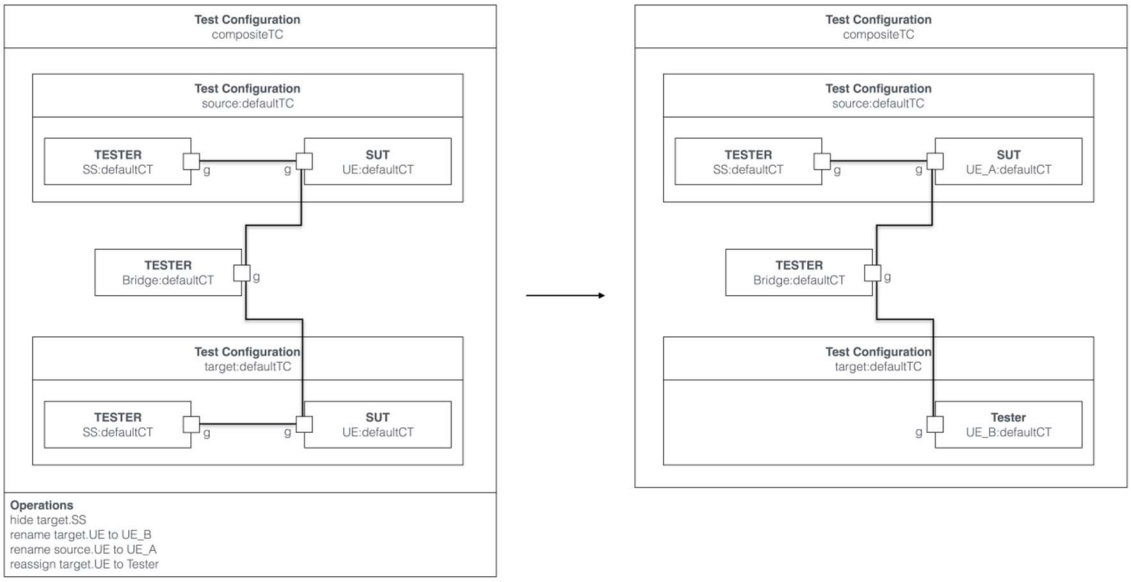


Figure A.2.1: Extended test configuration with operations and resulting test configuration

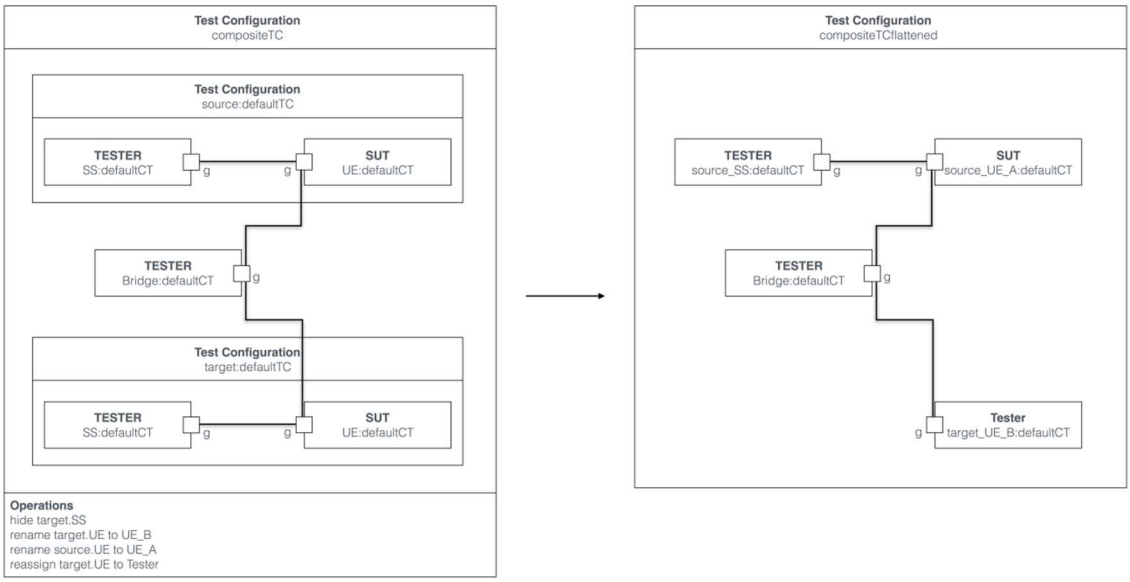


Figure A.2.2: Extended test configuration with operations and resulting flattened test configuration

A.3 Component Merging

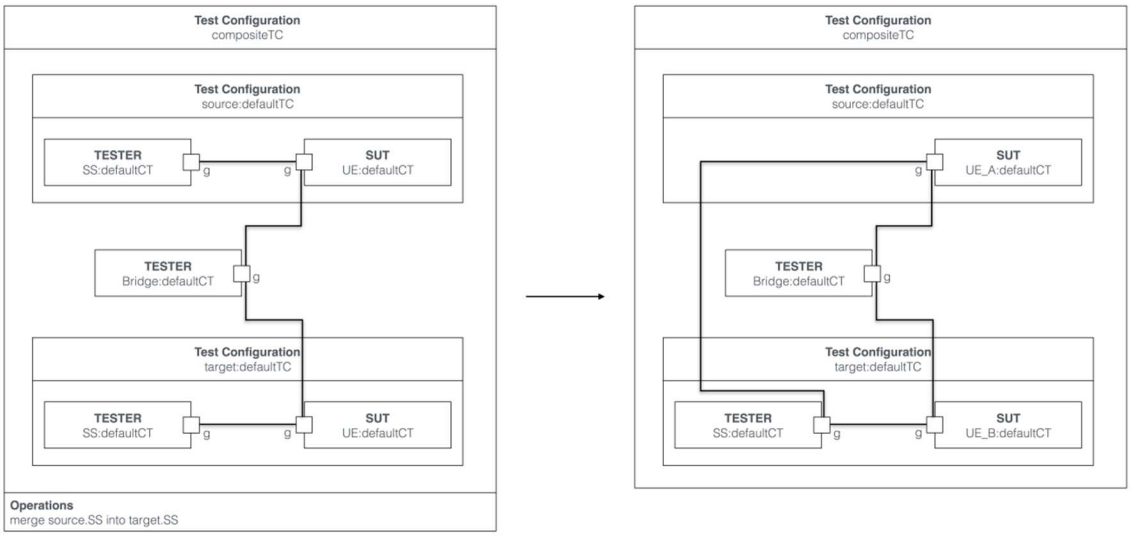


Figure A.3.1: Extended test configuration with merging and resulting test configuration

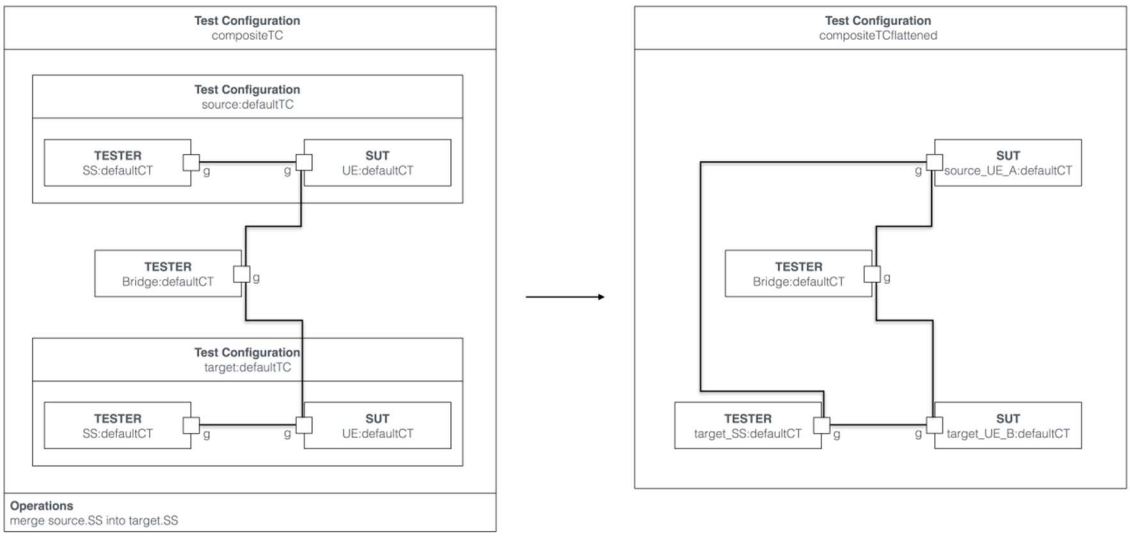


Figure A.3.2: Extended test configuration with merging and resulting flattened test configuration

In this example, the extended test configuration 'compositeTC' from Figure A.1.2 is refined further by applying the component merge operation. The extended test configuration resulting from the application of the test configuration operations is illustrated in Figure A.3.1. The corresponding test configuration after the flattening transformation is illustrated in Figure A.3.2.

History

Document history		
V1.1.1	May 2018	Publication
V1.2.1	August 2020	Publication
V1.3.1	March 2022	Membership Approval Procedure MV 20220527: 2022-03-28 to 2022-05-27
V1.3.1	May 2022	Publication