ETSI ES 203 119-6 V1.3.1 (2022-05)

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 6: Mapping to TTCN-3

Reference
RES/MTS-1196v1.3.1

Keywords
methodology, model, testing, TTCN-3

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

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 6 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.

1 Scope

The present document specifies how the elements of the Test Description Language (TDL) should be mapped to Testing and Test Control Notation version 3 (TTCN-3) [2]. The intended use of the present document is to serve as the basis for the development of TDL tools. The meta-model of TDL and the meanings of the meta-classes are described in ETSI ES 203 119-1 [1].

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 201 873-1 (V4.13.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language".

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

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

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

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

behaviour function: function used in TTCN-3 code that describes the behaviour of a TDL component instance

TTCNname: name of a TDL meta-model element that is used in the TTCN-3 code

NOTE: A TTCNname of a TDL element follows the syntactical rules of identifiers specified in ETSI ES 201 873-1 [2]. A TTCNname of a TDL element may contain a part that is derived from the TDL name with some prefixes and/or postfixes determined by a naming convention used in the TTCN-3 code.

3.2 Symbols

Void.

3.3 Abbreviations

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

IMS	IP Multimedia Subsystem
MTC	Main Test Component
PTC	Parallel Test Component
SUT	System Under Test
TDL	Test Description Language
TTCN-3	Testing and Test Control Notation version 3

4 Basic Principles

4.1 Introduction

While both TDL and TTCN-3 are standardized languages, there are various ways how TTCN-3 code can be derived from a TDL test description. This may result in different or even incompatible code intended to implement the same test description. Without a standardized mapping of TDL to TTCN-3, there could be a proliferation of different and possibly incompatible tool- and user-specific mappings of TDL test descriptions to executable test cases which can present new challenges to users and tool vendors.

A standardized mapping between the two languages provides a consistent approach for producing executable tests from high level test descriptions specified in TDL. This enables the generation of executable tests from TDL test descriptions in a (semi-) automatic way, and by extension of the re-use of existing test tools and frameworks for test execution. This way, test engineers can concentrate on the specification of test descriptions at a higher level of abstraction, while having a clear expectation of what the resulting test implementation will look like.

4.2 Document Structure

The meta-model of the Test Description Language is specified in ETSI ES 203 119-1 [1]. The present document specifies how the elements of the meta-model of TDL in locally ordered 'TestDescriptions' should be mapped to TTCN-3 code. The mapping of the globally ordered 'TestDescription's is outside the scope of the present document.

The structure of the present document follows the structure of the meta-model specification in ETSI ES 203 119-1 [1]. The clauses 5 to 10 describe the standardized mappings of the meta-model elements with identical clause numbers in ETSI ES 203 119-1 [1]. In each clause, first the description of the mapping of the corresponding meta-model element is described. It may be followed by a Constraints section, if the mapping is provided only with limitations. At the end of a clause an Example clause may exist to illustrate the mapping of the corresponding meta-model element. In the Examples the textual (specified in ETSI ES 203 119-1 [1]) or the graphical (specified in ETSI ES 203 119-3 [3]) notations of TDL can be used.

In some cases the structure of the TTCN-3 code may differ from the structure of the TDL specification or it requires some additional specification in TTCN-3. These special cases are described in clauses 8.2.11, 8.2.12 and 8.3.

At the end of the present document in Annex A several examples illustrate how the TTCN-3 code will look like after the rules of mapping specified in the present document are applied.

4.3 Notational Conventions

4.3.0 General

Elements (e.g. meta-classes, properties, etc.) from the TDL meta-model [1] are typed in between 'single quotes', e.g. 'StructuredDataType' or 'returnType'.

The TTCN-3 code elements (keywords, symbols, etc.) are typed in **`bold Courier New`** font, e.g. **`type port`** or **`{`** .

The TTCN-3 code to be generated is described by production rules, where applicable. The production rules are specified in between << and >> symbols. Inside a production rule, the concatenation between elements of that production rule is specified by a plus (+) symbol.

Iterations over collections of attributes of a metaclass make use of a function collect( ) with the following syntax: *propertyName*.collect(*VariableName* ':' *expression*), where *VariableName* is an alphanumeric word signifying the variable used in the subsequent *expression*, . *propertyName* is a string that shall be the same as the name of a property of a TDL metaclass. The type of this property determines the type of the variable denoted by *VariableName*.

The separator between the elements of an iteration is specified by the concat( ) function.

EXAMPLE 1:

The production rule:

```
type record <<self.name>> {
    << member.collect(m |  m.dataType.name + " " + m.name() ).concat(",")>>
}
```

for this TDL description

```
Type MSG (sessionID of type integer, content of type charstring);
```

will provide the following TTCN-3 code snippet:

```
type record MSG {
    integer sessionID,
    charstring content
}
```

The function select() selects a TDL element with a given value of a property.

EXAMPLE 2:

componentInstance.select(c | c.role = Tester) selects a 'componentInstance' whose 'role' property has a value of 'Tester'.

Other helper functions used in the production rules are collected in clause 4.3.1, while the predefined 'AnnotationType's that can be used to control the TTCN-3 code generation are listed in clause 4.3.2.

4.3.1 Functions used in production rules

- behaviourFunctionInReferencedTD(): returns the name of the behaviour function used in a referenced 'TestDescription' of the same tester component.
- equivalent(): returns the equivalent of the corresponding TDL element. If none of the structural modifications - described in clause 8.3 - on a TDL configuration is to be applied then the element.equivalent() is the element itself, otherwise what is specified in the corresponding sub-clause of clause 8.3.
- getKind(): returns the kind of an 'importedElement' (e.g. **`type`**, **`template`**, **`const`**, **`function`**, etc.) that can be used in a TTCN-3 **`import`** statement.
- toLower(): returns the value of a literal converted to all lowercase characters.
- TTCNname(): returns the name of the corresponding TDL element that will be used in the TTCN-3 code.

4.3.2 Predefined Annotations

A Predefined Annotation is an 'Annotation', whose 'key' is one of the following predefined 'AnnotationTypes'. The Predefined Annotations are used to help the TTCN-3 code generation in cases where the TTCN-3 code to be generated cannot be determined just from the TDL description:

- TTCN3Code: this 'AnnotationType' indicates that the 'body' of the 'Annotation' or of an 'InlineAction' contains a valid TTCN-3 code.

- Value: this 'AnnotationType' indicates that the annotated element shall not be treated as a template or a template type.

4.4 Conformance

For an implementation claiming to conform to this version of the mapping from TDL to TTCN-3, all features specified in the present document and in ETSI ES 203 119-1 [1] shall be implemented consistently with the requirements given in the present document and ETSI ES 203 119-1 [1].

5 Foundation

5.1 Overview

'Package's are mapped to TTCN-3 modules, 'ElementImport's to import statements, while 'Comment's, 'Annotation's, 'AnnotationType's and 'TestObjective's to TTCN-3 comments.

5.2 Mapping of Foundation Elements

5.2.1 Element

This is an abstract metaclass, therefore no mapping is defined.

Naming is different in TDL and in TTCN-3, therefore the names of the 'Element's used in TDL may not be used in TTCN-3. On one hand the set of characters allowed to be used in a TDL name is larger than the set allowed in TTCN-3 and on the other hand a TDL name may be a reserved keyword in TTCN-3. That is why the term TTCNname is introduced. A TTCNname of an 'Element' is the name of the 'Element' that is used in the TTCN-3 code.

A TTCNname may contain a part that is derived from the TDL name with some prefixes and/or postfixes determined by a naming convention used in the TTCN-3 code.

The present document does not specify how a TTCNname is generated from a TDL name. Neither the method how the TDL names are converted to valid TTCN-3 names nor the naming convention to be used in the TTCN-3 code, however the present document recommends a naming convention. The basic assumption of the recommended TTCNname is that it contains a part which is generated from the TDL name and it may be extended by some prefix(es) and/or postfix(es).

NOTE 1: The naming convention used in the present document is only a recommendation, in a concrete tool or implementation a different one may be used.

NOTE 2: In the following clauses the function TTCNname() will be used to get the TTCNname of the corresponding 'Element'.

5.2.2 NamedElement

This is an abstract metaclass, therefore no mapping is defined.

5.2.3 PackageableElement

This is an abstract metaclass, therefore no mapping is defined.

5.2.4 Package

A 'Package' shall be mapped to a module.

```
module <<self.TTCNname()>> {
}
```


For all import: as defined in clause 5.2.5.

NOTE: In TTCN-3 a module cannot contain another module, therefore a contained 'Package' will also be mapped to a "standalone" module. If information about the 'Package' structure needs to be kept in TTCN-3, then use a suitable naming convention.

5.2.5 ElementImport

The 'ElementImport' shall be mapped to **`import`** statement(s).

If the 'importedElement' is empty then an **`import`** … **`all`** statement shall be used:

```
import from <<self.importedPackage.TTCNname()>> all;
```

otherwise for all the 'importedElement' a selected **`import`** statement shall be used:

```
<< importedElement.collect(i | "import from " +" " + self.importedPackage.TTCNname() +" "
+i.getKind() + " " + i.TTCNname() ).concat(";")>>
```

NOTE: How the kind of the 'importedElement' (e.g. **`type`**, **`template`**, **`const`**, **`function`**, etc.) is determined is outside the scope of the present document. For this purpose e.g. an annotation or a naming convention can be used.

5.2.6 Comment

A 'Comment' shall be mapped to a comment:

/* <<self.body>> ***/**

5.2.7 Annotation

If the 'key' of the 'Annotation' is the predefined 'AnnotationType' TTCN3Code, then the 'Annotation' shall be mapped to its 'value' (that is to the TTCN-3 code itself), otherwise it shall be mapped to a comment:

/*

ANNOTATION <<self.key.TTCNname()>>

<<self.value>>

***/**

5.2.8 AnnotationType

'AnnotationType' shall be mapped to a comment:

/*

ANNOTATION TYPE <<self.TTCNname()>>

***/**

If the 'AnnotationType' has an extension:

/*

ANNOTATION TYPE <<self.TTCNname()>> **EXTENDS** <<self.extension.extending.TTCNname()>>

***/**

5.2.9 TestObjective

The 'TestObjective' shall be mapped to a comment:

/*

Test Objective <<self.name>>

Description: <<self.description>>

Objective URI: <<self.objectiveURI>>

***/**

5.2.10 Extension

This metaclass has no dedicated mapping, it is used solely in mapping of other metaclasses.

6 Data

6.1 Overview

Mapping of data definitions can either be done by the explicit 'DataElementMapping's provided by the user or if no 'DataElementMapping' is provided, then the TDL data definitions shall be mapped as they are specified in the following clauses. If there is a 'DataElementMapping' provided for a 'StructuredDataType' then mappings shall be provided for all of its 'Member's.

TDL does not make a distinction if a data instance is a value or a template, while TTCN-3 does. By default, all TDL 'DataInstance's, 'FormalParameter's, 'Variable's and return values of the 'Function's shall be mapped to a TTCN-3 template unless an 'Annotation' with the predefined 'AnnotationType' Value instructs otherwise. The 'PredefinedFunctionCall's and the predefined instances of the 'SimpleDataType' and 'SimpleDataInstance' elements shall be mapped to their TTCN-3 counterparts, while predefined instance 'Second' of 'Time' element shall be mapped to TTCN-3 data type **`float`**.

The 'DataUse's are mapped to their TTCN-3 counterparts, the 'DataInstanceUse' is a usage of a template or a constant; 'SpecialValueUse's are mapped to the TTCN-3 AnyValue (?), AnyValueOr None (*) and the special value omit, respectively; 'FunctionCall's and 'PredefinedFunctionCall's to function calls or operator invocation, 'FormalParameterUse' and 'VariableUse' to usage of a formal parameter or a variable. The inline modification of the 'DataUse's are mapped to a (sequence of) template modification(s).

6.2 Mapping of Data Definition Elements

6.2.1 DataResourceMapping

If 'DataResourceMapping' is provided, then its 'resourceURI' shall be used to locate the resource, where the 'DataElementMapping'(s) can be found.

6.2.2 MappableDataElement

This is an abstract metaclass, therefore no mapping is defined.

6.2.3 DataElementMapping

If a 'DataElementMapping' is provided by the user, then it shall be used for mapping the corresponding data definitions.

6.2.4 ParameterMapping

If there is a 'ParameterMapping' provided by the user for a 'Member' of a 'StructuredDataType' or a 'FormalParameter' of an 'Action' or a 'Function' then it shall be used for mapping the corresponding data definitions.

6.2.5 DataType

This is an abstract metaclass, therefore no mapping is defined.

6.2.6 DataInstance

This is an abstract metaclass, therefore no mapping is defined.

6.2.7 SimpleDataType

If there is a 'DataElementMapping' provided for a 'SimpleDataType' then it shall be used for the mapping, otherwise a 'SimpleDataType' shall be mapped to the TTCN-3 data type **`charstring`**, except for the Predefined Instances of the 'SimpleDataType' Element that shall be mapped according to the rules specified in clause 10.2.

The following TTCN-3 type definition of SimpleType shall be made in the declaration part of the module:

```
type charstring SimpleDataType;
```

For each SimpleDataType:

```
type SimpleDataType <<self.TTCNname()>> ;
```

6.2.8 SimpleDataInstance

A 'SimpleDataInstance' shall be mapped to a **`const`** if an 'Annotation' with the predefined 'AnnotationType' Value instructs this, otherwise it shall be mapped to a **`template`**.

If there is a 'DataElementMapping' provided for a 'SimpleDataInstance' then it shall be used for the mapping, otherwise a 'SimpleDataInstance' shall be mapped to:

```
template <<self.datatype.TTCNname()>> <<self.TTCNname()>> := " <<self.TTCNname()>> ";
```

or if an 'Annotation' with the predefined 'AnnotationType' Value is used, to:

```
const <<self.datatype.TTCNname()>> <<self.TTCNname()>> := " <<self.TTCNname()>> ";
```

6.2.9 StructuredDataType

If there is a 'DataElementMapping' provided for a 'StructuredDataType' then it shall be used for the mapping, otherwise if no 'Constraint' is associated to the 'StructuredDataType', it shall be mapped to a record:

```
type record <<self.TTCNname()>> {
    << allMembers().collect(m |  m.dataType.TTCNname() + " " + m.TTCNname() ).concat(",")>>
}
```

If a 'member' is optional (self.member.isOptional = true) then the TTCN-3 keyword **`optional`** shall be inserted after the TTCNname of that 'member'.

If a 'Constraint' with predefined 'ConstraintType' 'union' is associated to a 'StructuredDataType' then it shall be mapped to a union type:

```
type union <<self.TTCNname()>> {
    << allMembers().collect(m |  m.dataType.TTCNname() + " " + m.TTCNname() ).concat(",")>>
}
```

Constraints

If there is a 'DataElementMapping' provided for a 'StructuredDataType' then a mapping shall be provided for all of its 'Member's.

6.2.10 Member

This metaclass has no dedicated mapping, it is used solely in the mapping of other metaclasses.

6.2.11 StructuredDataInstance

A 'StructuredDataInstance' shall be mapped to a **`const`** if an 'Annotation' with the predefined 'AnnotationType' Value instructs this, otherwise it shall be mapped to a **`template`**.