Final draft ETSI ES 203 119-3 V1.5.1 (2022-03)

Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 3: Exchange Format

Reference
RES/MTS-TDL1193v151

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

ETSI

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

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

Siret N° 348 623 562 00017 - 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 final draft ETSI Standard (ES) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS), and is now submitted for the ETSI standards Membership Approval Procedure.

The present document is part 3 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 the exchange format of the Test Description Language (TDL) in the form of an XML Schema derived from the TDL meta-model [1]. The intended use of the present document is to serve as the specification of the format used for exchange of model instances and tool interoperability between TDL-compliant tools.

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] OMG®: "OMG Meta Object Facility™ (MOF) Core Specification", Version 2.4.2, formal/2014-04-03.

NOTE: Available at http://www.omg.org/spec/MOF/2.4.2/.

[3] OMG®: "OMG XML Metadata Interchange™ (XMI) Specification", Version 2.4.2, formal/2014-04-04.

NOTE: Available at http://www.omg.org/spec/XMI/2.4.2/.

[4] W3C® Recommendation 26 November 2008: "Extensible Markup Language (XML) 1.0 (Fifth Edition)".

NOTE: Available at http://www.w3.org/TR/REC-xml/.

[5] Recommendation ITU-T X.667: "Information technology - Procedures for the operation of object identifier registration authorities: Generation of universally unique identifiers and their use in object identifiers".

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 following terms apply:

TDL XMI document: XMI document that represents the serialization of a TDL model

TDL XMI Schema: XMI Schema that describes valid TDL XMI documents

XMI document: XML document that represents the serialization of a MOF model

XMI Schema: XML Schema definition that describes valid XMI documents

3.2 Symbols

Void.

3.3 Abbreviations

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

EBNF	Extended Backus-Naur Form
MOF	Meta-Object Facility™
TDL	Test Description Language
URI	Uniform Resource Identifier
UUID	Universal Unique Identifier
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
XSD	XML Schema Definition

4 Basic principles

4.1 Introduction

XMI is the serialization format for persistence and interchange of TDL models. XMI stands for XML Metadata Interchange and describes a unified way to serialize, persist, exchange and de-serialize MOF-based models [2]. The XMI specification [3] describes both production rules to create an XMI Schema and an XMI document for a MOF model.

The present document describes the production rules for both a TDL XMI Schema according to the TDL meta-model definition and TDL XMI documents for the serialization of TDL models. A TDL XMI Schema is useful to validate whether a TDL XMI document complies with the serialization rules specified in the present document. A complete validation of the represented TDL model cannot be performed solely with the TDL XMI Schema due to additional semantics introduced by MOF and XMI compared to XML Schema. Validation of TDL XMI documents shall be done in a two-step approach:

- Lexical (syntactical) validation based on the TDL XMI Schema: this validation step assures that XMI documents abide by the serialization rules for TDL models as described by the present document.
- Semantical validation based on the TDL meta-model definition: while deserializing TDL XMI documents, the modelling tool shall perform semantics checks based on the specification of the TDL meta-model and the additional XMI information.

The lexical validation is optional, since it only serves the purpose of syntactically assuring that a TDL XMI document is valid in terms of structure. Semantical validation is always required to ensure that the XMI document is a valid serialization of a TDL model. Due to the nature of the lexical validation, it is possible to produce valid TDL XMI documents with respect to the TDL XMI Schema but invalid TDL XMI documents with respect to the TDL meta-model specification.

4.2 Document Structure

The present document defines the exchange format for TDL model instances by means of a TDL XMI schema. It is structured as follows:

- Clause 5 "TDL XMI Schema" describes the rules that are applied for the production of a valid TDL XMI Schema according to TDL meta-model.
- Clause 6 "TDL XMI Document Serialization" describes the production rules that shall be applied when serializing TDL models. This clause contains a number of examples to comprehensibly illustrate the serialization of various TDL elements.
- Annex A specifies the canonical TDL XMI Schema based on the current TDL meta-model specification [1].

4.3 Notational Conventions

For the scope of the present document, the following notational conventions apply:

- Elements from the TDL meta-model or MOF model or XMI model are typeset in italic, e.g. the property *is Optional* of *Member* has the default value *false*.
- Elements from XML or XML Schema are typeset in the monospaced font Courier, e.g. the `complexType` declaration.

4.4 Conformance

TDL XMI documents shall be valid and well formed as defined by the XML recommendation [4].

TDL XMI documents shall be syntactically valid according the TDL XMI Schema described in the present document.

TDL XMI documents shall be semantically valid according the TDL meta-model specification.

5 TDL XMI Schema

5.1 TDL XMI Schema Production Rules

5.1.1 Overview

The TDL XMI Schema production rules refine the general XMI Schema derivation rules [3]. The TDL XMI Schema production rules unify the way in which TDL XMI documents shall be represented. Since XMI is more expressive than XML Schema in some parts which are relevant for MOF-based models, the following rules clarify how a TDL XMI Schema copes with the higher expressiveness of XMI compared to XML Schema:

- Multiple inheritance: Multiple inheritance is not required for the TDL XMI Schema.
- Identification: Element identification is based on the XMI attribute `ID` as specified in the XMI specification [3]. The value of an element's `ID` shall be derived according to the algorithm for generating universally unique identifier (UUID) specified in Recommendation ITU-T X.667 [5].
- Differences: TDL XMI Schema does not support the XMI elements *Difference*, *Add*, *Replace*, *Delete*.
- Bidirectionality: Bidirectional associations (i.e. association ends that have an opposite association end that can be directly navigated from each other) are resolved into two associations.
- The TDL XMI Schema described in the present document should not be used for the generation of the MOF-based TDL meta-model.

The structure of the TDL XMI Schema document complies with the XMI specification and contains the following concepts:

- An XML version processing instruction with an encoding character set information, e.g. `<?XML version="1.0" encoding="UTF-8">`.
- Any other valid XML processing instructions.
- An XML schema element, i.e. the root of the XML Schema document.
- An import XML element for the XMI namespace.
- Any other valid import XML elements to other referenced XML Schema documents.
- Declarations of concepts for the TDL meta-model.

5.1.2 TDL metaclass Element

The TDL XMI Schema element (root element of the TDL XMI Schema definition) has to comply with the following rules:

- The namespace declaration of XML Schema shall be present and set to `xsd="http://www.w3.org/2001/XMLSchema"`
- The namespace declaration of XMI shall be present and set to `xmi="http://www.omg.org/XMI"`
- The namespace declaration of the TDL meta-model shall be present and set to `tdl="http://www.etsi.org/spec/TDL/1.6.1"`
- The target namespace declaration shall be the same as the namespace declaration for the TDL meta-model

5.1.3 Import statements

For the definition of import statements, the following rules shall be applied:

- An import of the fixed declarations that are mandatory for every XMI schema shall be given as first import statement after the XMI Schema element. These declarations are in the namespace `http://www.omg.org/spec/XMI/20131001`. For further information about the fixed declarations of the XMI schema element, refer to the XMI specification [3], clause "XMI Document and Schema Design Principles".
- Import elements for other XML Schema documents are allowed as long as they do not contradict with the concepts provided by the TDL meta-model or rules specified for the TDL XMI Schema document.

5.1.4 Representation of the TDL meta-model

The declaration of the concepts contained in the TDL meta-model shall comply with the following rules:

- **Metaclasses** are represented as a combination of the XML Schema elements `complexType` and `element` where `complexType` specifies the metaclass structure and `element` its concrete use in a certain situation. The use of `element` is required for expressing containment relationships.
- **Abstract metaclasses** are represented similar to concrete metaclasses, but with `abstract="true"` set in the `complexType` declaration.
- **Enumerations** are represented as XML Schema `simpleType` elements. A restriction is used with `base` set to XML Schema `xsd:NCName`.
- **Metaclass properties in general** represent properties that are typed by MOF primitive types (e.g. *String*, *Integer*, etc.), TDL enumerations, or TDL metaclasses. They are either represented as an XML Schema `element` or `attribute`, depending on the multiplicity of the metaclass property, the type of the metaclass property, and whether the type of property is contained within in the same XMI file as the property or not. In either case, the `name` shall be set to the property's *name*. The *type* shall be set to the XML Schema type declaration corresponding to the property's type.
- **Metaclass properties that refer to metaclasses in a different XMI file** are always serialized as nested elements. These properties are represented as XML schema elements `element`. Such an `element` declaration is embedded in a `choice` model group with `minOccurs="0"` and `maxOccurs="unbounded"` attributes within the `complexType` declaration representing the containing metaclass of the property.
- **Metaclass properties of simple type** represent properties that are typed by MOF primitive types (e.g. *String*, *Integer* etc.) or TDL enumerations. They are represented either as an XML Schema `element` or `attribute`, depending on the multiplicity of the metaclass property. The MOF primitive type to XML Schema simple type mapping is defined as follows:
  - *MOF::String* -> xsd:string
  - *MOF::Integer* -> xsd:integer
  - *MOF::Boolean* -> xsd:Boolean
  - *MOF::Real* -> xsd:decimal
- **Metaclass properties of simple type with an upper bound of 1** are represented as `attribute` elements of the `complexType` declaration representing the property's containing metaclass. If the lower value of the metaclass property is 1 (i.e. the property value is mandatory), the `attribute`'s attribute `use="required"` shall be set.

- **Metaclass properties of simple type with an upper bound greater than 1** are represented as `element` elements. Such `element` declarations are embedded in a `choice` model group with `minOccurs="0"` and `maxOccurs="unbounded"` attributes within the `complexType` declaration representing the containing metaclass of the property. Mixed use of both an `attribute` and an `element` elements for the same metaclass property is not permitted.

- **Metaclass properties of metaclass type composing metaclasses** represent properties of a TDL metaclass that are typed by another TDL metaclass with compositional relationship among those metaclasses. These properties are represented as XML schema `element` elements. Such `element` declarations are embedded in a `choice` model group with `minOccurs="0"` and `maxOccurs="unbounded"` attributes within the `complexType` declaration representing the containing metaclass of the property.

- **Metaclass properties of metaclass type referencing metaclasses** represent properties of a metaclass that are typed by another TDL metaclass with a non-compositional relationship among those metaclasses. These properties are represented as XML Schema `attribute` elements, or `element` elements if the referenced metaclass is contained in a different XMI file. The `type` attribute shall be set to `xsd:string` in case of `attribute` elements, and to the `complexType` declaration in case of `element` elements. Referencing `attribute`s that represent collections shall list the *xmi:id* of each referenced metaclass in a white-space separated list.

- **Single inheritance** is represented by the XML Schema extension with `base` set to the qualified name of the general metaclass.

- **Multiple inheritance** is not permitted by XML Schema and is not required for the TDL XMI Schema production.

- **Associations** of the TDL meta-model are not represented in the TDL XMI Schema.

5.1.5 Specifics on the TDL metaclass *Element*

The TDL metaclass *Element* represents the single root metaclass for any other TDL metaclass. As such, it declares both XMI-specific and TDL-specific `attribute` elements which are inherited by all other TDL metaclasses. This clause describes the rules that shall be applied to the TDL metaclass *Element:*

- The *Element* `complexType` declaration contains the XMI-specific `attribute` elements for object identity, i.e.:
```
<xsd:attribute ref="xmi:id"/>
<xsd:attributeGroup ref="xmi:ObjectAttribs"/>
```
The `attribute` group `xmi:ObjectAttribs` provide additional `attribute` elements from the XMI namespace to each TDL element. These `attribute` elements grouped under the umbrella of the above-mentioned name are used for the XMI object identity via `xmi:id`, type identification via `xmi:type`, and cross-document references via `xmi:href`.

- The *Element* `complexType` declaration contains the XMI:Extension element in its structure, i.e.:
```
<xsd:choice
  …
  <xsd:element ref="xmi:Extension"/>
</xsd:choice>
```
Both the XMI-specific concepts for object identity and general extensibility are inherited by any other metaclass definition of the TDL meta-model, since every metaclass in the TDL meta-model is either an immediate or a transitive subclass of the TDL metaclass *Element.*

5.2 Examples

5.2.1 TDL metaclass *Element*

The example below shows the declaration of the TDL metaclass *Element*. *Element* has a special meaning in the meta-model, since it represents the superclass of all other TDL metaclasses.

TDL meta-model representation

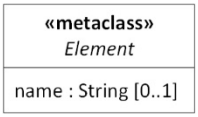

Figure 1: TDL Element metaclass definition (simplified)

XML Schema representation

```
<xsd:complexType abstract="true" name="Element">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <xsd:element name="comment" type="tdl:Comment"/>
      <xsd:element name="annotation" type="tdl:Annotation"/>
      <xsd:element ref="xmi:Extension"/>
    </xsd:choice>
    <xsd:attribute ref="xmi:id"/>
    <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
    <xsd:attribute name="name" type="xsd:string">
</xsd:complexType>

<xsd:element name="Element" type="tdl:Element"/>
```

5.2.2 Enumerations

The example below shows the XML Schema representation of enumerations defined in the TDL meta-model.

TDL meta-model representation

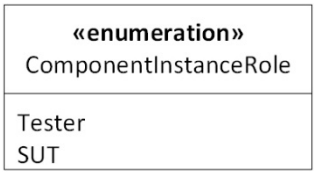

Figure 2: TDL enumeration ComponentInstanceRole example

XML Schema representation

```
<xsd:simpleType name="ComponentInstanceRole">
   <xsd:restriction base="xsd:NCName">
      <xsd:enumeration value="SUT"/>
      <xsd:enumeration value="Tester"/>
    </xsd:restriction>
</xsd:simpleType>
```

5.2.3 Metaclasses

The example below demonstrates the representation of several aspects related to more complex metaclass definitions. It shows both the definition of simple type properties (the *role* property of the metaclass *ComponentInstance*), compositional associations, as well as reference associations. Simple type property declarations and reference associations do not differ from each other in the corresponding XML Schema representation except by their type.

TDL meta-model representation

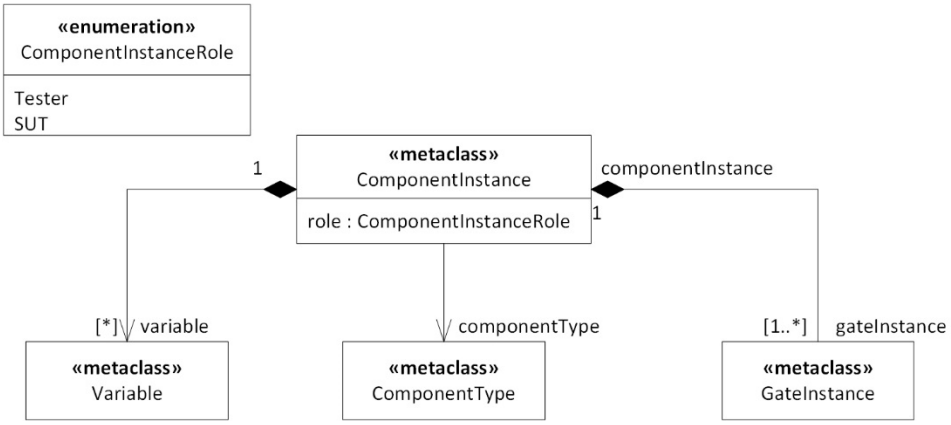

Figure 3: Examples on composition and non-compositional associations

XML Schema representation

```
<xsd:complexType name="ComponentInstance">
    <xsd:complexContent>
      <xsd:extension base="tdl:Element">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="gateInstance" type="tdl:GateInstance"/>
          <xsd:element name="variable" type="tdl:Variable"/>
        </xsd:choice>
        <xsd:attribute name="componentType" type="tdl:ComponentType">
        <xsd:attribute name="role" type="tdl:ComponentInstanceRole">
      </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="ComponentInstance" type="tdl:ComponentInstance"/>
```

5.2.4 Multiple inheritance

Multiple inheritance is not required by the current version of the TDL meta-model.

6 TDL XMI Document Serialization

6.1 TDL XMI Document Production Rules

6.1.1 TDL XMI Document Header Structure

The TDL XMI Document (i.e. the serialization of an instance model of the TDL meta-model) shall comply syntactically with the TDL XMI Schema definition. Since the concept space and validation capabilities of XML Schema are not sufficient to guarantee validity of XMI Documents with respect to a MOF meta-model, an additional set of rules for the construction of XMI Documents is required.

The TDL XMI Document header structure shall include all the information required to build a valid TDL XMI Document without considering the serialized model. Thus, the header structure described in this clause shall be identical for all TDL XMI Documents. However, it may be just a subset of the entire header structure, e.g. due to user-specific extensions. User-specific extensions are not described in the present document in great detail. The XMI specification [3] provides further information on how to manage user-defined extensions to XMI Documents.

The production rules for TDL XMI Document header structures are defined as follows:

- An XML version processing instruction with an encoding character set information, e.g.
  `<?XML version="1.0" encoding="UTF-8">`
  Any other valid XML processing instructions

- A XMI element, i.e. the root of the TDL XMI Document
- The `xmi:version` attribute shall be set to `2.0`
- The XMI element specifies at least the following namespace declarations:
```
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:tdl="http://www.etsi.org/spec/TDL/1.6.1"
```

6.1.2 TDL XMI Document Content Structure

The TDL XMI Document content structure specifies the TDL model serialization format. Each serialized TDL model represents a valid instance of the TDL meta-model at a given point in time. *Valid* means that the serialized TDL XMI Document shall be both:

- successfully validated against the TDL XMI Schema (as specified in the present document) by any XML Schema validator, and
- successfully deserialized by a tool that has implemented the TDL meta-model (as specified in the TDL specification [1]) in accordance to the semantics of a MOF model.

In addition to the validity rules, the following rules shall apply for any content structure of a TDL XMI Document:

- Derived values are not serialized, they shall be calculated from other values present in the model upon deserialization.
- Properties representing the opposite end of a compositional association (i.e. the property referring to the container metaclass) are not serialized.
- Properties whose values in the instance model are the same as the default defined in the abstract syntax are not serialized.
- Properties with an upper bound greater than one are always serialized as nested `element` of the corresponding `complexType` of the property's metaclass, regardless of whether these properties are typed by a metaclass or a primitive type, and whether they represent compositional or non-compositional references.
- The predefined `attribute` element `xmi:type` imported from the XMI namespace shall always be used to indicate the concrete type of the serialized MOF object. This is not required for the root element of the TDL model, since its type can be inferred. The value of the `xmi:type` element shall match with the qualified name of the metaclass of the referenced TDL element instance, i.e. the prefix of the imported namespace, such as *tdl*, followed by the name of the TDL element, such as *Package*, separated by a colon.

6.1.3 Cross-document element references

With the `attribute` group `xmi:LinkAttribs` XMI enables the referencing of elements across documents. The XMI specification [3] provides different mechanisms for this capability (see clause 7.6.2 of [3] "Linking Attributes in the XMI specification"). While reusing the very same `attribute` group for the definition of the TDL XMI Schema, the TDL Document production rules restrict cross-document references to be expressed by means of the `href attribute` element only.

Element linking with `href` shall only be applied for cross-document references, even if the XMI specification allows in principle to link elements in the same document via `href`. Therefore, the only valid format of such a `href attribute` element is `"tdl_xmi_file#someid"`, with `tdl_xmi_file` representing a URI reference to a serialized TDL XMI Document (the file suffix may vary from `".tdl"` to `".xmi"` since this is technology-dependent) and `someid` representing the `xmi:id` of the referenced element.

For further information on cross-document linking using the `href attribute`, refer to EBNF production rule 2m in clause 9.5 of the XMI specification [3].

6.2 Examples

6.2.1 TDL XMI Document Header Structure Example

This example shows the minimal information for valid TDL XMI Documents that shall be identical for each serialized TDL model.

```
<?XML version="1.0" encoding="UTF-8">
<tdl:Package
  xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:tdl="http://www.etsi.org/spec/TDL/1.6.1">
  <!—- any TDL XMI Document Content -->
</tdl:Package>
```

6.2.2 TDL XMI Document Content Structure Examples

6.2.2.1 Instances of TDL metaclasses

This example shows the serialization of an instance of the metaclass *Package*. It has no content and represents the root Package of the model. The `xmi:type` declaration is not required here, because the type of the element can be precisely inferred from the XMI element declaration.

Concrete syntax and model instance representation

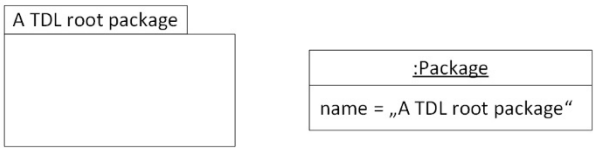

Figure 4: Concrete syntax (left) and model instance representation of a TDL root Package (right)

XMI Document representation

```
<tdl:Package xmi:id="f5d8e27c714049d4913e71661658558c" name="A TDL root package"/>
```

6.2.2.2 Compositional references to TDL metaclasses

This example shows the serialization of compositional references to instances of TDL metaclasses as complex content of the containing XMI element. In addition, it demonstrates the omission of the `attribute` element that represents the containing instance. This information can be precisely inferred from the XMI element that contains the nested XMI elements.

Concrete syntax and underlying model representation

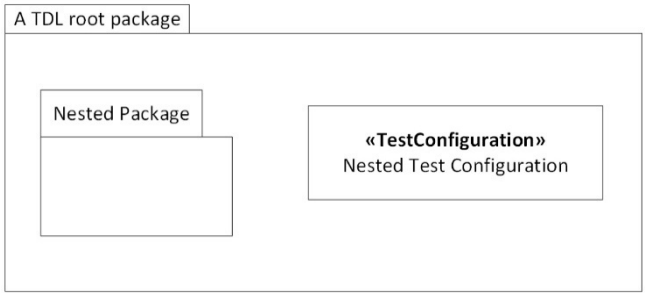

Figure 5: Concrete syntax representation of compositional references example

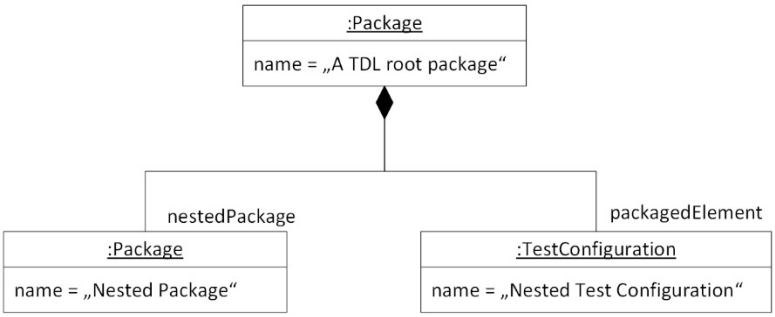

Figure 6: Model instance representation for compositional references example

XMI Document representation

```
<tdl:Package xmi:id="f5d8e27c714049d4913e71661658558c" name="A TDL root package">
  <nestedPackage xmi:id="6e973c47caf84687851ea7bbf8551e93" xmi:type="tdl:Package" name="Nested
  Package"/>
  <packagedElement xmi:id="58923a3a160649aaa37f3d8d228ec53c" xmi:type="tdl:TestConfiguration"
  name="Nested Test Configuration"/>
</tdl:Package>
```

6.2.2.3 Non-compositional references to TDL metaclasses

This example shows the serialization of non-compositional references to instances of TDL metaclasses as `attribute` elements of the referencing XMI element.

Concrete syntax and underlying model representation

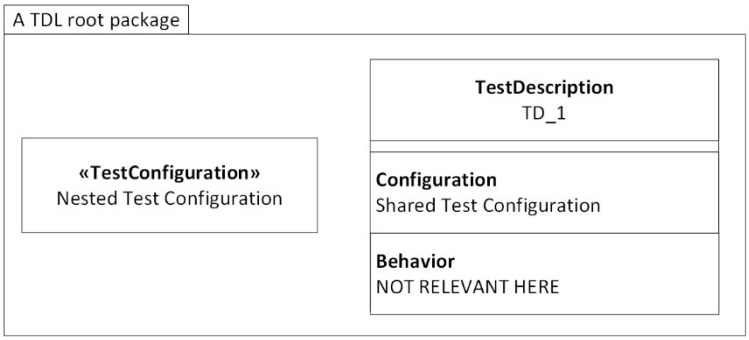

Figure 7: Concrete syntax representation of non-compositional references example

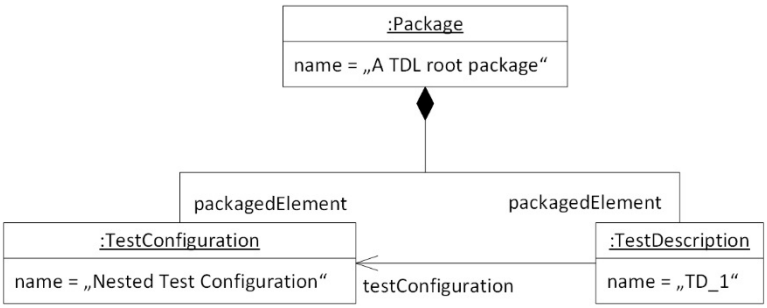

Figure 8: Model instance representation of non-compositional references example

XMI Document representation

```
<tdl:Package xmi:id="f5d8e27c714049d4913e71661658558c" name="A TDL root package">
  <packagedElement xmi:id="58923a3a160649aaa37f3d8d228ec53c" xmi:type="tdl:TestConfiguration"
  name="Nested Test Configuration">
    <!-- content of TestConfiguration -->
  </packagedElement>
  <packagedElement xmi:id="c0776b9fa8df49599196e3154d895d80" xmi:type="tdl:TestDescription"
  name="TD_1" testConfiguration="58923a3a160649aaa37f3d8d228ec53c">
```

```
    <!-- content of TestDescription -->
  </packagedElement>
</tdl:Package>
```

6.2.2.4 Instance of TDL metaclasses with simple type properties

This example shows the serialization of simple type properties of TDL metaclasses, i.e. properties of TDL metaclasses that are typed either a MOF primitive type or TDL enumeration.

Concrete syntax and underlying model representation

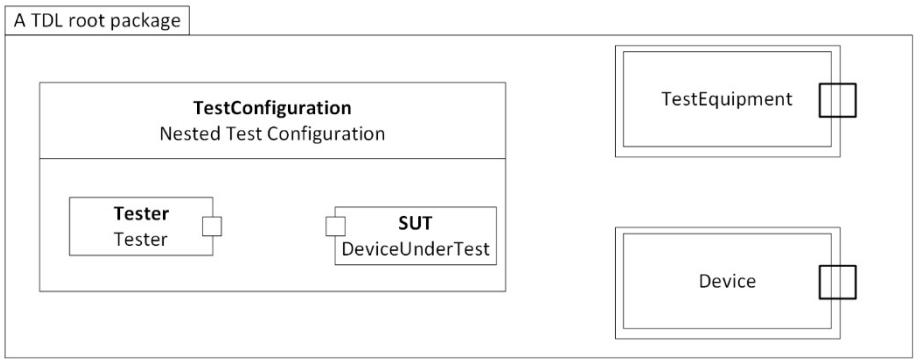

Figure 9: Concrete syntax representation of simple type properties example

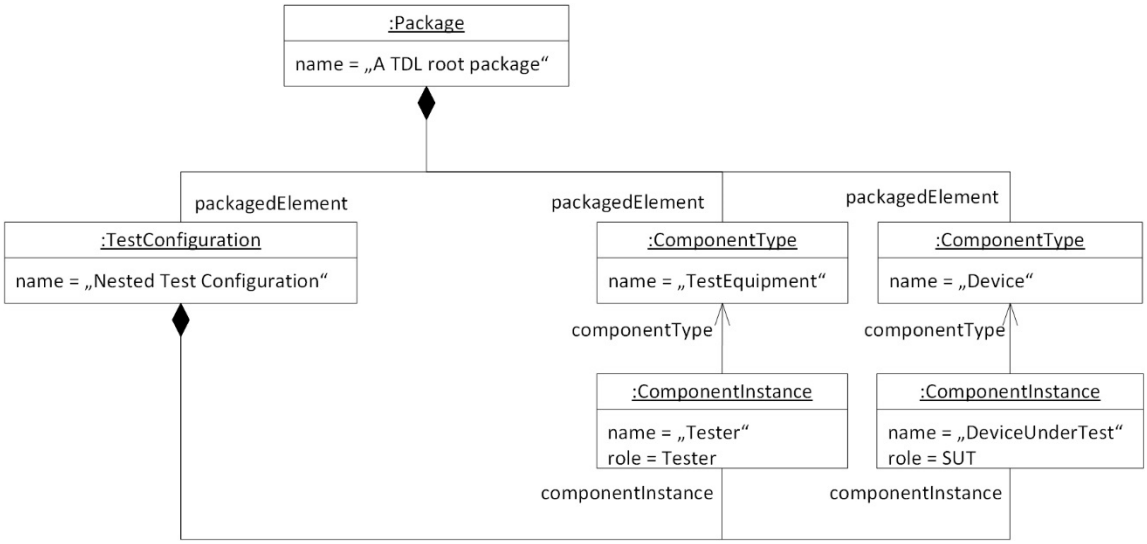

Figure 10: Instance model representation of simple type properties example

XMI Document representation

```
<tdl:Package xmi:id="f5d8e27c714049d4913e71661658558c" name="A TDL root package">
  <packagedElement xmi:id="a87c7d462c734d94ad277f4f33b63f37" xmi:type="tdl:ComponentType"
  name="TestEquipment" >
    <!-- content of ComponentType -->
  </packagedElement>
  <packagedElement xmi:id="38fc40c72e1c4abaac9c66d3ab29120e" xmi:type="tdl:ComponentType"
  name="Device" >
    <!-- content of ComponentType -->
  </packagedElement>
  <packagedElement xmi:id="58923a3a160649aaa37f3d8d228ec53c" xmi:type="tdl:TestConfiguration"
  name="TestConfiguration">
    <componentInstance xmi:id=" 931c879e1f6a4abb9aadb43fec029e48" xmi:type="tdl:ComponentInstance"
    componentType=" a87c7d462c734d94ad277f4f33b63f37" role="Tester" name="Tester"/>
    <componentInstance xmi:id=" 0272f014ed1e40d48458f883b8795bd7" xmi:type="tdl:ComponentInstance"
    componentType="38fc40c72e1c4abaac9c66d3ab29120e" role="SUT" name="DeviceUnderTest"/>
  </packagedElement>
</tdl:Package>
```

6.2.2.5 Handling of default values for properties

This example shows the handling of default values for serialization. As specified in clause 6.2.2, a value of a property is not serialized if the value is identical with the default value declared for this property in the abstract syntax. In the above-mentioned example an instance of the *ComponentInstance* is shown with its property *role* set to the *Tester*. Since the TDL meta-model does not declare a default value for this property, the given value needs to be serialized.

XMI Document representation

```
<componentInstance xmi:id=" 931c879e1f6a4abb9aadb43fec029e48" xmi:type="tdl:ComponentInstance"
componentType=" a87c7d462c734d94ad277f4f33b63f37" role="Tester" name="Tester"/>
<packagedElement xmi:id="931c879e1f6a4abb9aadb43fec029e48" xmi:type="tdl:SimpleDataInstance"
name="AnInstance"/>
```

The following example shows the definition of a *StructuredDataType* with a mandatory *Member*. Since the property *isOptional* is set to *false* by default in the corresponding metaclass specification, the literal *false* shall not be serialized, if the user does not explicitly change the value to *true*. That means, whenever the user models non-optional (*optional* implicitly set to *false*) *Members*, the value *false* does not need serialization.

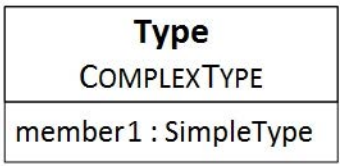

Figure 11: Definition of a TDL StructuredDataType

The corresponding object model is defined as follows:

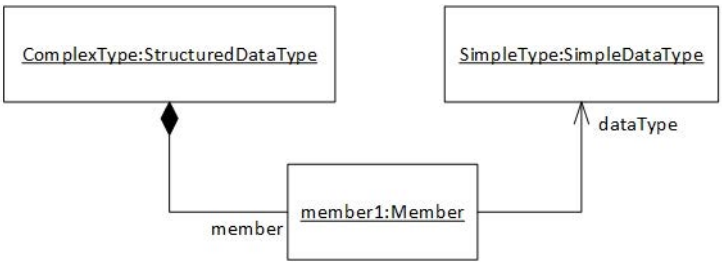

Figure 12: Object model showing the definition of a StructuredDataType

XMI Document representation

```
<packagedElement xmi:id="931c879e1f6a4abb9aadb43fec029e48" xmi:type="tdl:StructuredDataType"
name="ComplexType"/>
  <member xmi:id="931c879e1f6a4abb9aadb43fec029e49" xmi:type="tdl:SimpleDataType" name="member1"/>
</packagedElement>
```

6.2.2.6 Cross-document references

XMI allows splitting up the output of the serialization of a model into multiple files. The main application scenario of XMI cross-document references is when two different models, which are serialized into different XMI files, have to work with each other.

Figure 13 denotes a scenario in which the TDL Package *Package1* declares an import statement to the TDL Package *Package2*.

Figure 13: Linking between two XMI files

Figure 14 represents the corresponding object model (the object model).

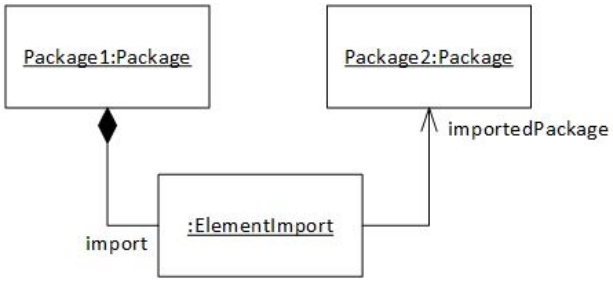

Figure 14: Object model for linking example

Both models shall be serialized into different XMI files. The corresponding XMI Document serializations for both models are:

File 1 (containing *Package1*)

```
<tdl:Package xmi:id="f5d8e27c714049d4913e71661658558c" name="Package1">
  <import xmi:id="a87c7d462c734d94ad277f4f33b63f37" xmi:type="tdl:ElementImport">
    <importedPackage xmi:type="tdl:Package" href="file2.xmi#shortID">
  </import>
</tdl:Package>
```

File 2 (containing *Package2*)

```
<tdl:Package xmi:id=" f12c5a522c235d94ad2634f4e34b63f32" name="Package1">
```

Annex A (normative): Technical Representation of the TDL XMI Schema

The technical representation of the TDL XMI schema is included as an electronic attachment es_20311903v010501m0.zip which accompanies the present document. See the readme contained in the zip file for details.

Annex B (informative): Bibliography

- ETSI ES 203 119-2 (V1.5.1): "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 2: Graphical Syntax".
- ETSI ES 203 119-4 (V1.5.1): "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 4: Structured Test Objective Specification (Extension)".
- ETSI ES 203 119-7 (V1.3.1): "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 7: Extended Test Configurations (Extension)".

History

Document history		
V1.1.1	June 2015	Publication
V1.2.1	September 2016	Publication
V1.3.1	May 2018	Publication
V1.4.1	August 2020	Publication
V1.5.1	March 2022	Membership Approval Procedure MV 20220527: 2022-03-28 to 2022-05-27