Test Assertions Part 2 - Test Assertion Markup Language Version 1.0

Committee Specification Draft 03 / Public Review Draft 03~~Draft 02~~

08 September 2011

~~30 August 2010~~

Specification URIs:

This v~~V~~ersion:

http://docs.oasis-open.org/tag/taml/v1.0/csprd03/testassertionmarkuplanguage-1.0-csprd03.pdf (Authoritative)~~d02/testassertionmarkuplanguage-1.0-cd-02.html~~
http://docs.oasis-open.org/tag/taml/v1.0/csprd03/testassertionmarkuplanguage-1.0-csprd03.html~~d02/testassertionmarkuplanguage-1.0-cd-02.odt~~
http://docs.oasis-open.org/tag/taml/v1.0/csprd03/testassertionmarkuplanguage-1.0-csprd03.odt~~d02/testassertionmarkuplanguage-1.0-cd-02.pdf (Authoritative)~~

Previous v~~V~~ersion:

http://docs.oasis-open.org/tag/taml/v1.0/cs01/testassertionmarkuplanguage-1.0-cs-01.pdf (Authoritative)~~d01/testassertionmarkuplanguage-1.0-cd-01.html~~
http://docs.oasis-open.org/tag/taml/v1.0/cs01/testassertionmarkuplanguage-1.0-cs-01.html~~d01/testassertionmarkuplanguage-1.0-cd-01.odt~~
http://docs.oasis-open.org/tag/taml/v1.0/cs01/testassertionmarkuplanguage-1.0-cs-01.odt~~d01/testassertionmarkuplanguage-1.0-cd-01.pdf (Authoritative)~~

Latest v~~V~~ersion:

http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0.pdf (Authoritative)~~html~~
http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0.html~~odt~~
http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0.odt~~pdf (Authoritative)~~

Technical Committee:

OASIS Test Assertions Guidelines (TAG) TC

Chair~~(s)~~:

Jacques Durand (jdurand@us.fujitsu.com), Fujitsu Limited

Editors:

Stephen D. Green (stephengreenubl@gmail.com), Individual
Jacques Durand (jdurand@us.fujitsu.com), Fujitsu Limited

Additional Work Product artifacts:

This prose specification is one component of a Work Product which also includes:
XML schema: taml/v1.0/csprd03/xsd/testAssertionMarkupLanguage.xsd

Related work:

~~Patrick Curran~~
~~Jacques Durand~~

~~**Editor(s):**~~

~~Stephen D Green~~

~~**Related Work:**~~

This specification is related to:

- Test Assertions Part 1 - Test Assertions Model Version 1.0. Latest version. http://docs.oasis-open.org/tag/model/v1.0/testassertionsmodel-1.0.html
- Test Assertions Guidelines Version 1.0. Latest version. http://docs.oasis-open.org/tag/guidelines/v1.0/testassertionsguidelines.html

Declared XML namespaces:

~~OASIS TAG TC - Test Assertions Model - Version 1.0~~

~~OASIS TAG TC - Test Assertions Guidelines - Version 1.0~~

~~**Declared XML Namespace(s):**~~

http://docs.oasis-open.org/ns/tag/taml-201002/

Abstract:

This document defines an XML vocabulary for representing test assertions aligned with the Test Assertions Model~~efines a markup for writing test assertions~~.

Status:

This document was last revised or approved by the OASIS Test Assertions Guidelines (TAG) TC on the above date. The level of approval is also listed above. Check the "Latest version" ~~This document was last revised or approved by the TAG TC on the above date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved Version"~~ location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/tag/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/tag/ipr.php).

Citation format:

When referencing this specification the following citation format should be used:

[TAML]

Test Assertions Part 2 - Test Assertion Markup Language Version 1.0. 08 September 2011. OASIS Committee Specification Draft 03 / Public Review Draft 03. http://docs.oasis-open.org/tag/taml/v1.0/csprd03/testassertionmarkuplanguage-1.0-csprd03.html.

~~The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/tag/.~~

Notices

Table of Contents

1 Introduction

[All text is normative unless otherwise indicated.]

1.1 Terminology

Within this specification, the key words "shall", "shall not", "should", "should not" and "may" are to be interpreted as described in Annex H of [ISO/IEC Directives] if they appear in bold letters.

1.2 Normative References

[TAM] OASIS Committee Specification Draft 02, "Test Assertions, Part 1 Test Assertions Model Version 1.0", May 2011 http://docs.oasis-open.org/tag/model/v1.0/testassertionsmodel-1.0**[TAM]** OASIS Committee Draft 02, "Test Assertions Model Version 1.0", August 2010 http://docs.oasis-open.org/tag/model/v1.0/cd02/testassertionsmodel-1.0-cd-02.pdf

[ISO/IEC Directives] ISO/IEC Directives, Part 2 Rules for the structure and drafting of International Standards, International Organization for Standardization, 2004

[ISO/IEC Directives] ISO/IEC Directives, Part 2 Rules for the structure and drafting of International Standards, International Organization for Standardization, 2004. http://www.iso.org/iso/standards_development/processes_and_procedures/iso_iec_directives_and_iso_supplement.htm

[RFC 2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[XSD1] Henry S. Thompson, David Beech, Murray Maloney, et. al., editors. XML Schema Part 1: Structures. http://www.w3.org/TR/xmlschema-1/ . World Wide Web Consortium, 2000.

[XSD2] Paul V. Biron and Ashok Malhotra, editors. XML Schema Part 2: Datatypes. http://www.w3.org/TR/xmlschema-2/ . World Wide Web Consortium, 2000.

[XSD1] Henry S. Thompson, David Beech, Murray Maloney, et. al., editors. XML Schema Part 1: Structures. http://www.w3.org/TR/xmlschema-1/ . World Wide Web Consortium, 2000.

[XSD2] Paul V. Biron and Ashok Malhotra, editors. XML Schema Part 2: Datatypes. http://www.w3.org/TR/xmlschema-2/ . World Wide Web Consortium, 2000.

1.3 Non-normative References

[XPATH2] W3C Recommendation, "XML Path Language (XPath) Version 2.0" . World Wide Web Consortium, 2007. http://www.w3.org/TR/2007/REC-xpath20-20070123/

[NIEM] National Information Exchange Model, Naming and Design Rules (NDR), Version 1.3. United States government, 2008, http://niem.gov/library.php#naming released on 2008-10-31

[XPATH1] W3C Recommendation, "XML Path Language (XPath) Version 1.0" http://www.w3.org/TR/xpath . World Wide Web Consortium, 1999.

2 Markup Representation of Test Assertions

2.1 Binding to Test Assertions, Part 1 Test Assertions Model

This specification defines markup for test assertions conforming to the model defined in the OASIS TAG TC Test Assertions Part 1, Test Assertions Model [TAM] Section 3 (Test Assertion) ~~both Section 3 (Test Assertion) and Section 4 (Test Assertion Set)~~.
Each 'class' in the Test Assertions Model is represented by an element of the same or similar name in the Test Assertion Markup Language ~~, with exceptions as follows~~:

Model Name	**Markup Name**
Class: testAssertion	element: testAssertion
attribute: id	attribute: id
Class: normativeSource	element: normativeSource
Class: target	element: target
Class: predicate	element: predicate
Class: prerequisite	element: prerequisite
Class: tag	element: tag
Class: variable	element: var
Class: description	element: description
Class: prescription	element: prescription
~~**Model Name**~~	~~**Markup Name**~~
~~Class: testAssertion~~	~~element: testAssertion~~
~~attribute: id~~	~~attribute: id~~
~~attribute: language~~	~~attribute: lg~~
~~Class: normativeSource~~	~~element: normativeSource~~
~~Class: target~~	~~element: target~~
~~Class: predicate~~	~~element: predicate~~
~~Class: prerequisite~~	~~element: prerequisite~~
~~Class: tag~~	~~element: tag~~
~~Class: variable~~	~~element: var~~

~~Class: description~~	~~element: description~~
~~Class: prescription~~	~~element: prescription~~

All element and attributes names are given in lower camel case (lower-case, with embedded concatenated words starting upper-case if any). Type names consist of the element name with the suffix '`_type`' appended.
~~There are classes in the Test Assertions Model [TAM] which are associated with the class '`shared`'. These classes are suffixed with '`Shared`' to distinguish them from classes of the same name associated with the '`testAssertion`' class. In the Test Assertion Markup Language the names of the complex types which correspond to these 'shared' classes include the '`Shared`' suffix while the corresponding element names do not.~~
~~All element and attributes names are given in lower camel case. Type names consist of the element name with the suffix '`_type`' appended.~~
Where the model specifies an attribute named '`content`', usually with a base datatype '`string`', the markup provides for this either with a base type of `xsd:string` assigned to an element's type (or a datatype derived from `xsd:string` such as `xsd:normalizedString` or `xsd:token`) or by allowing mixed content for the element's type.
Elements '`testAssertion`' and '`testAssertionSet`' are global elements and can be top level elements in a TAML markup instance (e.g., be root elements of an XML document). All other TAML elements are local in a markup instance under the TAML schema (i.e., are descendant children of a global element as provided by the schema).

~~Markup cardinalities are the same as those specified in the model.~~
~~Elements '`testAssertion`', '`testAssertionSet`' and '`testAssertionDocumentHeader`' are declared as global elements and can be used as top level elements in a markup instance; all other elements are declared locally and are not valid as top level elements in a markup instance.~~

2.2 Conventions Used in the XML Markup and its Usage

The namespace prefix in use for the test assertion markup throughout this document is **taml,** understood to be bound to the namespace: http://docs.oasis-open.org/ns/tag/taml-~~representating the namespace: http://docs.oasis-open.org/tag/ns/v1.0/taml/~~201002/.
Many of the elements in the Test Assertion Markup Language allow extensions both in their attributes and in their children elements. With the exception of any extension inside these three elements: `taml:testAssertionSet/taml:common`, `taml:testAssertion/taml:report` and `taml:testAssertion/taml:normativeSource/taml:comment`, which do not put any restriction on the use of namespaces in their user-defined child elements, all other additional attributes and elements **shall** be in a namespace other than the taml namespace (as indicated in compact RelaxNG notation by using the subtraction: "- taml:*").
~~It is recommended to use this prefix in all instances of this markup.~~
~~In many cases, the XML representation of a mandatory model element - i.e. an attribute or association of cardinality (1..1) - is optional in the markup. This is because such elements, although mandatory, may be implicitly represented and therefore not using the conventional explicit mark-up element intended for them.~~
Instances of this markup are intended to be used either "standalone" i.e. in documents that do not contain any other markup foreign to this specification, or "embedded", i.e. as elements inside documents the root element of which belongs to a namespace foreign to this specification. Instances of this markup are XML elements representing either test assertions, or test assertion sets.
The compact Relax NG notation is used for representing the XML definitions.
The XPath notation may be used for representing attributes or elements, relative to~~of~~ their containing element, e.g.: taml:testAssertion/@id for the attribute 'id' of the taml:testAssertion element.

2.3 Test Assertion

2.3.1 taml:testAssertion

The taml:testAssertion element is representing the class 'testAssertion' in the Test Assertions Model [TAM] Detailed semantics of this class and its elements can be found in the Test Assertions Model. ~~Detailed semantics of this class and its elements can be found in [TAM],~~

Compact Relax NG definition:

```
element taml:testAssertion { testAssertion_def }
testAssertion_def =
       attribute id { xsd:normalizedString }?,
       attribute lg { xsd:normalizedStringNCName }?,
       attribute name { xsd:normalizedString }?,
       attribute schemaVersionId { xsd:normalizedString }?,
       attribute * - taml:* { text }*,
       element taml:description { dprescription { prescription_def }?,
       element taml:var { var_def }*,
       element taml:normativeSource { normativeSource_def }?,
       element taml:target { target_def }?,
       element taml:prerequisite { logicalexpr_def }?,
       element taml:predicate { logicalexpr_def }?,
       element taml:prescription { prescription_def }?,
       element taml:tag { tag_def }*,
       element taml:report { report_def }*,
       element * - taml:* { anyElement }* { anyElement }*
```

The XML representation of most model elements that are mandatory - i.e. attribute or association of cardinality (1..1) - is optional in the markup. This is because such elements, although mandatory, may be implicitly represented at test assertion set level. When such a mandatory element is not represented in a test assertion instance (taml:testAssertion), this test assertion **shall** be embedded in a test assertion set (see Section 2.4) and this test assertion set **shall** contain a taml:common element that contains the missing test assertion part as a child~~If no provision is made for an implicit identifier to be assigned to a test assertion, a test assertion identifier **shall** be provided for every test assertion using the '`id`' attribute of the '`testAssertion`'~~ element.

The `testAssertion` element has an optional language attribute, '`lg`', which is an addition for TAML purposes (it has no correspondence in the Test Assertions Model). This attribute is used to explicitly declare which prose or expression language is used for the expressions in the associated element - in that case throughout the test assertion. It is possible to declare the language for an individual part of a test assertion such as the predicate or the prerequisite (discussed later). If the 'lg' attribute is provided for a 'testAssertion' element, that attribute value **shall** be used as the default value of the 'lg' attribute for any descendant element of the 'testAssertion' element for which an 'lg' attribute is allowed and optional but not provided.

The optional 'name' attribute is an addition for TAML purposes, and provides for attaching an informal, human-readable name to the test assertion for convenient display and intuitive referencing, in addition to the formal identification attribute 'id'.

The `testAssertion` attribute '`schemaVersionId`' **should** be used as version identifier of the markup language published schema.

Conformance to the Test Assertions Model [TAM] requires that a test assertion **shall** have a normative source, a target and a predicate although the representation of these may be implicit. So the `normativeSource`, `target` or `predicate` elements **may** be absent from a `testAssertion` element as their contents could be inherited from a test assertion set taml:common element. .

Conformance to the Test Assertions Model [TAM] requires that a test assertion **may** have prerequisite(s), prescription level and tags, either implicitly or explicitly. It also specifies a part called a variable represented here by taml:var.

One additional, optional element added for convenience to the usability of the markup and for tool support is '`taml:report`'. It does not correspond to a part defined in the Test Assertions Model [TAM] but is specified here as an addition for TAML purposes .

Like many of the elements in the Test Assertion Markup Language, the `testAssertion` element has a language attribute, '`lg`'. This attribute is used to explicitly declare which prose or expression language is used for the logical expressions in the associated element - in that case throughout the test assertion. It is possible to declare the language for an individual part of a test assertion such as the predicate or the prerequisite (discussed later). Declaring the language for the test assertion as a whole using the '`lg`' attribute of the `testAssertion` element **shall** mean that every part in the test assertion uses that language for its expression. A profile **may** specify a set of language identifiers for use with this attribute.
The `testAssertion` attribute '`schemaVersionId`' **should** be used as part of the default Test Assertion Markup Language (version 1) version methodology which assigns a version identifier to every version of the markup language published schema. The version methodology allows that several versions of the schema **may** use the same namespace when they are considered to be compatible with previous versions using that namespace. These versions are denoted 'minor versions' while 'major versions' of the markup schema have differing namespaces. Test Assertion Markup Language schema 'minor versions' **should** be distinguished in the element at the top level of an XML instance or fragment (such as a fragment embedded within another markup) by the provision of a version identifier in the '`schemaVersionId`' attribute of this element when that top level element is either the '`testAssertion`' or '`testAssertionSet`' element.
Conformance to the Test Assertions Model [TAM] requires that a test assertion **shall** have a normative source, a target and a predicate unless either or all of these are implicit. So the `normativeSource`, `target` and `predicate` elements **may** be implicit and also may be inherited from a test assertion set or document ancestor of the test assertion (specified later).
Conformance to the Test Assertions Model [TAM] requires that a test assertion **may** have prerequisite(s), prescription level and tags, either implicitly or explicitly. It also specifies a part called a variable represented here by taml:var..
One additional, optional element added for convenience to the usability of the markup and for tool support is 'taml:`report`'. It does not correspond to a part defined in the Test Assertions Model [TAM] but is a allowed as a conforming extension.

Example:

The test assertion below is addressing a requirement about XML schema Naming and Design Rules (NDR) from a National Information Exchange Model 5 specification. It uses XPath as expression language for several of its elements (predicate, target) and attributes (target/@idscheme)NIEM specification. It uses XPath as expression language for several of its elements (predicate, target) and attributes (target/@idscheme). It concerns targets that are xsd:complexType elements in an XML schema:

```
<taml:testAssertion
            id="TA_R6.1"
            lg="http://www.w3.org/TR/xpath20/XPath2.0"
            xmlns:taml="http://docs.oasis-open.org/ns/tag/taml-
tag/ns/v1.0/taml/201002/">
      <taml:description>xsd:complexType/@mixed value check, as
specified in NIEM</taml:description>
   <taml:normativeSource>[Rule 6-1] Within the schema, an element
xsd:complexType SHALL NOT own the attribute mixed with the value true.
   </taml:normativeSource>
   <taml:target type="complexType"
idscheme="fn:concat('complexType:',@name)">//xsd:complexType</taml:tar
get>
   <taml:predicate>not(@mixed) or @mixed ne 'true'</taml:predicate>
   <taml:prescription level="mandatory"/>
      <taml:report label="fail" message="Rule 6-1 violation"/>
</taml:testAssertion>
```

In the above example, XPath2.0 (as indicated by taml:testAssertion/@lg= "http://www.w3.org/TR/xpath20/") is the default expression language for all parts of the test assertions that have @lg as optional attribute but do not use it, such as taml:target , taml:predicate, and taml:report.

2.3.2 taml:normativeSource

The taml:normativeSource element is representing the class 'normativeSource' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in the Test Assertions Model. The `taml:normativeSource` element allows for mixed content.~~[TAM].~~

Compact Relax NG definition:

```
element taml:normativeSource { normativeSource_def }
normativeSource_def =
  attribute * - taml:* { text }*,

  element taml:comment { comment_def }?,
  element taml:interpretation { interpretation_def }?,
  element taml:refSourceItem { refSourceItem_def }*,
  element taml:textSourceItem { textSourceItem_def }*,
  element taml:derivedSourceItem { refSourceItem_def }*,
  element * - taml:* { anyElement }* & textext ?
```

2.3.2.1 taml:refSourceItem

The normative source may include elements named '`refSourceItem`' so that one or more references **may** be used to point to the original text as it exists in the specification itself.
~~The normative source includes an element named '`refSourceItem`' so that a reference **may** be used to point to the original text as it exists in the specification itself.~~

Compact Relax NG definition:

```
element taml:refSourceItem { refSourceItem_def }
refSourceItem_def =
       attribute srcname { xsd:normalizedString }?,
       attribute uri { xsd:anyURI        attribute lg { NCName }?,
       attribute uri { xsd:normalizedString }?,
       attribute documentId { xsd:normalizedString }?,
       attribute versionId { xsd:normalizedString }?,
       attribute revisionId { xsd:normalizedString }?,

       attribute resourceProvenanceId { xsd:normalizedString }?,
       attribute * - taml:* xsd:* { text }*,
       text
```

The `refSourceItem` element provides for metadata which **may** be used to specify the identification of a normative source item resource. The `uri` attribute **may** contain a URL, URI or IRI pointing to the location of the source item. The attribute `@srcname` represents the name attribute in the model. The other metadata attributes include ~~or URI or IRI pointing to the location of the source item. The other metadata attributes includes~~ information about the kind of resource involved and most appropriately its provenance (such as authorship identifiers to certify its authenticity) and version, etc. The actual content of the `refSourceItem` element may be a string describing informally this source.

2.3.2.2 taml:textSourceItem

An alternative to using a reference to point to the normative source in a specification is to actually quote verbatim the source item so the normative source includes an element named '`textSourceItem`' which allows a direct, verbatim quote of the specification text.

Compact Relax NG definition:


```
element taml:textSourceItem { textSourceItem_def }
textSourceItem_def =
       attribute txtname { xsd:normalizedString }?,
       attribute * - taml:*  { text }*  attribute lg { NCName }?,
       attribute * - xsd:* { text }*,
       text
```

The attribute `@txtname` represents the name attribute in the model.

2.3.2.3 taml:derivedSourceItem

An alternative again to quoting verbatim the source item is to derive a form of words equivalent in meaning to the source item and for this the normative source includes an element named '`derivedSourceItem`'. This is particularly useful when the source consists of tables, diagrams, graphs or text spread over several parts of the specification.

The compact Relax NG definition of `taml:derivedSourceItem` is same as for `taml:refSourceItem` (see `refSourceItem_def`).

The `derivedSourceItem` element provides for metadata which may be used to specify the identification of the normative source item resource from which the source information has been derived. The element has a structure similar to the `refSourceItem` element. The main difference with `refSourceItem` is that the content of the `derivedSourceItem` element shall represent the derived re-wording of the source.

2.3.2.4 taml:comment

The `comment` element **may** be used to simply add comments of any kind (or as further specified in a conformance profile for this markup or a customization thereof) to a normative source test assertion part. The `taml:comment` element allows for mixed content.

~~An alternative again to quoting verbatim the source item is to derive a form of words equivalent in meaning to the source item and for this the normative source includes an element named '`derivedSourceItem`'. This is particularly useful when the source consists of tables, diagrams, graphs or text spread over several parts of the specification. The `derivedSourceItem` element provides for metadata which may be used to specify the identification of the normative source item resource from which the source information has been derived. The element has a structure similar to the `refSourceItem` element. The main difference with `refSourceItem` is that the content of the `derivedSourceItem` element shall represent the derived re-wording of the source.~~

Compact Relax NG definition:

```
element taml:comment { comment_def }
comment_def =
       attribute * - taml:*  { text }*  attribute lg { NCName }?,
       element * { anyElement }*  &  text

       attribute * - xsd:* { text }*,
       text
```

2.3.2.5 taml:interpretation

The `interpretation` element **may** be used to simply add an alternative description in prose of any kind to a normative source test assertion part. This allows a prose expression to be added to improve human understanding of its logic.

~~The `comment` element **may** be used to simply add comments of any kind (or as further specified in a conformance profile for this markup or a customization thereof) to a normative source test assertion part.~~

Compact Relax NG definition:


```
element taml:interpretation { interpretation_def }
interpretation_def =
       attribute name { xsd:normalizedString }?,
       attribute * - taml:*  { text }* attribute lg { NCName }?,
       attribute * - xsd:* { text }*,
       text
```

The interpretation element **may** be used to simply add an alternative description in prose of any kind (or as further specified in a conformance profile for this markup or a customization thereof) to a normative source test assertion part. This allows a prose expression to be added to improve human understanding of its logic.

2.3.3 taml:target

The taml:target element is representing the class 'target' in the Test Assertions Model [TAM] . Detailed semantics about this class and its elements can be found in the Test Assertions Model[TAM].

Compact Relax NG definition:

```
element taml:target { target_def }
target_def =
       attribute type { xsd:normalizedString }?,
       attribute idscheme { xsd:normalizedString }?,
       attribute lg { _xsd:normalizedString }?,
       attribute * - taml:* xsd:* { text }*,
       element * - taml:*  { anyElement }*  & text
       text ?
```

The content of `taml:target` may identify a single item (an implementation or part of it) under test, or a class of items to which the test applies. For example, in the NIEM example of test assertion (section 2.3.1) the target contains an XPath expression: //xsd:complexType, which indicates that any element that matches this expression inside an XML schema document under test, will qualify as a target for this test assertion. The '`target`' element has a '`type`' attribute which **should** be used to identify the target category, when defined. A target '`idscheme`' attribute **may** be used to specify the identity scheme associated with this target type or category. For example, its value can be a function such as an XPath expression, that produces a unique ID for each target instance, as illustrated in the NIEM example of section 2.3.1. In case the test assertion applies to a single target instance (as opposed to a category of targets), the '`idscheme`'A target can either be a specific item or a category of items. The 'target' element has a 'type' attribute **should** be used to identify the target category, when defined. A target 'idscheme' attribute or, for a set of test assertions, a shared target 'idscheme' attribute (see later) **may** be used to specify the identity scheme associated with this target type or category. For example, its value can be a function such as an XPath expression, that produces a unique ID for each target instance. In case the test assertion applies to a single target instance (as opposed to a category of targets), the idscheme attribute **may** contain the identifier of this target.

The target content **may** be an expression in a specialized formal expression language which **should** be identified using the '`lg`' attribute. Such an expression or function should identify the set of target instances to which the test assertion applies. This content may also be a textual representation of the target instance(s) under consideration. The `taml:target` element allows for mixed content.s to which the test assertion applies. This content may also be a textual representation of the target instance(s) under consideration. .

2.3.4 taml:prerequisite

The taml:prerequisite element is representing the class 'prerequisite' in the Test Assertions Model [TAM] . Detailed semantics about this class and its elements can be found in the Test Assertions Model[TAM].

Compact Relax NG definition:

```
element taml:prerequisite { logicalexpr_def }
logicalexpr_def =
       attribute lg { _xsd:normalizedString }?,
       attribute * - taml~~xsd~~:* { text }*,
       element * - taml:* { anyElement }* & ~~text~~
```

The prerequisite **may** be expressed using a specialized formal expression language which **may** be identified using the 'lg' attribute. The prerequisite content is stating a logical expression or statement to be evaluated (as "true" or "false") over the target, or over some collateral artifact or a set of these, e.g. identified using variables (see later), or over a combination of these. The taml:prerequisite element allows for mixed content.

2.3.5 taml:predicate

The taml:predicate element is representing the class 'predicate' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in the Test Assertions Model[TAM].

Compact Relax NG definition:

```
element taml:predicate { logicalexpr_def }
logicalexpr_def =
       attribute lg { _xsd:normalizedString }?,
       attribute * - taml~~xsd~~:* { text }*,
       element * - taml:* { anyElement }* & ~~text~~
```

The predicate **may** be expressed using a specialized formal expression language which **may** be identified using the 'lg' attribute. The predicate content is stating a logical expression or statement to be evaluated (as "true" or "false") over the target, and optionally over a set of collateral artifacts, e.g. identified using variables. The taml:predicate element allows for mixed content ~~(see later)~~.

2.3.6 taml:prescription

The taml:prescription element is representing the class 'prescription' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in the Test Assertions Model~~Detailed semantics about this class and its elements can be found in [TAM]~~.

Compact Relax NG definition:

```
element taml:prescription { prescription_def }
prescription_def =
       attribute level { "mandatory" | "preferred" | "permitted" }?,
       attribute * - taml~~xsd~~:* { text }*,
       text
       ~~text ?~~
```

The allowable values for the attribute 'level' of the element prescription **may** be extended beyond the predefined values of mandatory, preferred and permitted.
The base datatype of any custom extended enumerations for prescription levels **shall** be W3C XML Schema [XSD2] datatype 'QName'. Custom enumerations **shall** be prefixed with a namespace prefix as-

sociated with a namespace declared in the markup. Default namespaces (without a prefix) **shall not** be used.
~~The base datatype of any custom extended enumerations for prescription levels **shall** be W3C XML Schema [XSD2] datatype 'Qname'. Custom enumerations **should** be prefixed with a namespace prefix associated with a namespace declared in the markup. Default namespaces (without a prefix) **shall not** be used.~~

Besides the use of the '`level`' attribute, the element content (`xsd:normalizedString`) **may** be used to express further or more detailed information regarding the prescription level using prose or as a logical expression.

2.3.7 taml:description

The taml:description element is representing the class 'description' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in the Test Assertions Model~~[TAM]~~.

Compact Relax NG definition:

```
element taml:description { description_def }
description_def =
       attribute * - taml:* { text }*~~, attribute lg { xsd:normalizedString }?~~,
       element * { anyElement }*  &  text~~attribute * - xsd:* { text }*,~~
```

The `description` element may be used to add a description in prose of any kind (or as further specified in a conformance profile for this markup or a customization thereof) to a test assertion. The `taml:description` element allows for mixed content.

```
~~       text~~
```

~~The `description` element may be used to add a description in prose of any kind (or as further specified in a conformance profile for this markup or a customization thereof) to a test assertion.~~

2.3.8 taml:tag

The taml:tag element is representing the class 'tag' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in the Test Assertions Model~~[TAM]~~.

Compact Relax NG definition:

```
element taml:tag { tag_def }
tag_def =
       attribute tname { xsd:normalizedString },
       attribute * - taml:* { text }*~~lg { xsd:normalizedString }?~~,
~~       attribute * - xsd:* { text }*,~~
       text
```

The content of the taml:tag element is representing the "content" attribute of the corresponding class 'tag' in the model.
The tag/@tname attribute corresponds to the name attribute in the model, and **shall** be used in a tag element and have a non-empty value. Some tag names are reserved and have special meaning (listed in the Test Assertions Model [TAM], section 3.2.12). They are:~~name attribute **shall** be used in a tag element and have a non-empty value.~~

- **`DefinesNormativeProperty`**
- **`NormativeProperty`**
- **`VersionAdd`**
- **`VersionDrop`**

2.3.9 taml:var

The taml:var element is representing the class 'variable' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in the Test Assertions Model.~~' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in [TAM]~~.

Compact Relax NG definition:

```
element taml:var { var_def }
var_def =
       attribute vname { xsd:normalizedString }~~name { xsd:normalizedString }?~~,
       attribute vtype~~lg~~ { xsd:normalizedString }?,
       attribute lg {  xsd:normalizedString }?~~* - xsd:* { text }*~~,
       attribute * - taml:* { text }*,
       element * - taml:*  { anyElement }*  &  text
       ~~text~~
```

The content of the taml:var element is representing the "content" attribute of the corresponding class 'variable' in the model. The var/@vname attribute corresponds to the name attribute in the model, and **shall** be used in a var element and have a non-empty value. The var/@vtype attribute may be used to indicate the intended type of this var element content, e.g. in case it is to be used in some expression~~tag element is representing the "content" attribute of the corresponding class 'tag' in the model.[TAM]~~.
When declared inside a test assertion, the scope of a variable includes all the parts of this test assertion. The variable may be referred to in any part of a test assertion e.g. using a notation such as '$variable1' where the corresponding variable is named 'variable1'. The `taml:var` element allows for mixed content.
~~When declared inside a test assertion, the scope of a variable includes all the parts of this test assertion. The variable may be referred to ,in any part of a test assertion e.g. using a notation such as '$variable1' where the corresponding variable is named 'variable1'.~~

2.3.10taml:report

The taml:report element is not representing any class in the Test Assertions Model [TAM]. It is added for convenience when test assertions are expected to contain reporting information to be used by test cases derived from these test assertions.

Compact Relax NG definition:

```
element taml:report { report_def }
report_def =
       attribute label { xsd:normalizedString~~NCName~~ }?,
       attribute message { xsd:normalizedString~~text~~ }?,
       attribute when { xsd:normalizedString~~text~~ }?,
       attribute lg { xsd:normalizedString }?,
       attribute * - taml~~xsd~~:* { text }*,
       element *  { anyElement }*  &  ~~—~~ text
```

The optional `taml:report` element is used to associate a message and a label with each possible evaluation of a target instance, as listed in the Test Assertion Model (Section 3.2.4). ~~`report` element **may** be used to specify what messages and labels are included in any reports generated from any test cases based on the test assertion. It **may** also be used to specify the outcomes of a test case and under which condition(s) each outcome is to be produced, so the `report` element is optional and of multiple cardinality in the `testAssertion` element.~~
The combination of allowing both mixed content (text can be interspersed with the XML tags) and extra elements from other namespaces ('`xsd:any`') means that the content of this element can be a mixture of text and XML elements. ~~, say, HTML or other simple formatting markup.~~
The attribute `label` **shall** allow values '`fail`', '`pass`' and '`notQualified`' as content, corresponding to the test assertion semantics defined in the Test Assertions Model [TAM] (section 3.2.4) as follows: ~~standard possible outcomes defined in the Test Assertions Model [TAM] as follows~~

- "notQualified" corresponds to the target evaluation outcome "Target not qualified" notQualified corresponds to the outcome "Target not qualified".
- "pass" corresponds to the target evaluation outcome "Normative statement fulfilled [by the Target]" pass corresponds to the outcome "Normative statement fulfilled [by the Target]".
- "fail" corresponds to the target evaluation outcome "Normative statement NOT fulfilled [by the Target]".

Further values - e.g. 'warning', 'undetermined' - **may** be added. .
The optional `when` attribute **may** be used to state an additional condition that must be satisfied in order for this `report` element to apply, when more than one `report` elements are associated with the same outcome described by the `label` attribute.
The content of optional attribute "`message`" **shall** describe the general meaning of the assertion evaluation outcome, e.g. provide a standard error message.
The `taml:report` element allows for mixed content.

- fail corresponds to the outcome "Normative statement not fulfilled [by the Target]"

Further values - e.g. 'warning', 'undetermined' - **may** be added which may be defined in a conformance profile.
The optional `when` attribute **may** be used to state the condition that must be satisfied in order for the outcome described by the `label` attribute, to apply. This attribute is usefule when defining new outcomes (values for `label`) beyond the standard possible outcomes ('`fail`', '`pass`' and '`notQualified`'), or when a standard outcome uses an interpretation different from the default interpretation, which is:

- outcome `notQualified` is conditioned by the taml:prerequisite evaluating to 'false'.
- outcome `pass` is conditioned by the taml:predicate evaluating to 'true', and the taml:prerequisite (if any) evaluating to 'true').
- outcome `fail` is conditioned by the taml:predicate evaluating to 'false', and the taml:prerequisite (if any) evaluating to 'true').

The content of optional attribute `message` **shall** describe the meaning of the assertion outcome, e.g. s provide an error message. A more detailed diagnostic message **may** be provided in the content of the `report` element. Further attributes **may** be defined for the `report` element in a conformance profile.

2.4 Test Assertion Set

The test assertion set markup described here is an extension to the markup that strictly represents the test assertion model. For convenience, it represents a container element for sets of test assertions. However, test assertions elements may be part of XML documents that do not make use of the container element described here.

2.4.1 taml:testAssertionSet

The taml:testAssertionSet element is the container element for test assertions. representing the class 'testAssertionSet' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in [TAM].

Compact Relax NG definition:

```
element taml:testAssertionSet { testAssertionSet_def }
testAssertionSet_def =
       attribute setid { xsd:normalizedString } id { xsd:normalizedString }?,
       attribute setname { xsd:normalizedString       attribute lg { NCName }?,
       attribute * - taml:* { text } *  attribute schemaVersionId
{ xsd:normalizedString }?,
       element taml:common { common_def }?    attribute *   taml:* { text }*,
       element taml:testAssertionRefList { testAssertionRefList_def }
*DocumentHeader { testAssertionDocumentHeader _def }?,
       element taml:testAssertion { testAssertion_def } *
```



```
__
common_def =
       attribute lg { xsd:normalizedString }?,
       attribute * - taml:*  { text }*,
       element * { anyElement }*  &  text
```

The attribute `taml:testAssertionSet/@setname` gives a name to the test assertions set. The attribute `taml:testAssertionSet/@setid` uniquely identifies the test assertions set.
The `testAssertionSet` element **may** be used to group together test assertions either by direct inclusion of the test assertions within the test assertion set, or by references to externally-defined test assertions (using taml:testAssertionRef).
A `testAssertionSet` element **may** be used to wrap together all the test assertions in a document. A document containing a set of test assertions **may** have `testAssertionSet` as the top element.
The `common` element **may** be used to group together parts or definitions that are common to all test assertions in the set, e.g. "global" var definitions, tag definitions, parts of a test assertion that are implicitly shared by all. Any individual test assertion in the set can however override the elements in the `common` element. The `common` element may contain additional descriptions and elements not specified here, that help understand the purpose of this set and its relationship with external material.
The `taml:common` element allows for mixed content.

2.4.2 taml:testAssertionRefList

The taml:testAssertionRefList element is representing a list of references to test assertion element(s) defined outside the taml:testAssertionSet parent element (e.g. described in another taml:testAssertionSet instance.)
Compact Relax NG definition:

```
element taml:testAssertionRefList { testAssertionRefList_def }
testAssertionRefList_def =
       attribute listname { xsd:normalizedString }?,
       attribute sourcedoc { xsd:anyURI }?,
       attribute * - taml:* { text }*,
       element taml:testAssertionRef { testAssertionRef_def }*,
       element taml:shared { shared_def }?,
       
       element taml:testAssertionRef { testAssertionRef_def }*,
       element taml:testAssertionSet { testAssertionSet_def }*,
       element taml:testAssertionSelection { testAssertionSelection_def }*,
       element taml:testAssertionList {
              element taml:testAssertion { testAssertion_def }* }?,
       element { anyElement }*
```

The `testAssertionSet` element **may** be used to group together test assertions either by direct inclusion of the test assertions within the test assertion set, using the taml:testAssertionList, container or by references to constructs (possibly defined externally) that contain these test assertions, such as taml:testAssertionRef , (recursively) taml:testAssertionSet or taml:testAssertionSelection..
An instance **may** have this as the top element.
A test assertion set **may** be used to wrap together all the test assertions in a document. In this case the `testAssertionDocumentHeader` **may** be used once within a document either on its own (i.e. outside the `testAssertionSet` element) or as a direct child of the outermost `testAssertionSet` element. See section later on `testAssertionDocumentHeader`.
Another purpose of the test assertion set is that it **may** be used to provide a set of shared test assertion parts and their values in the same way to more than one test assertion (either to limit repetition or to ensure that the values correspond or to provide scope for variables across such test assertions). (See the section on the '`shared`' element below.)
The `testAssertionSelection` child element **may** be used when test assertions are to be imported from some other source(s), yet only a subset is of interest based on a selection criterion. The `testAssertionSelection` element allows for providing this selection criterion while preserving the refer-

ence to the original set of test assertions. The selection criterion may either be a list of test assertion Ids, or a logical condition e.g. an XPath expression appropriate to the markup such as '`//taml:testAssertion[...]`' which identifies for association with the test assertion set all current document test assertions written in the Test Assertion Markup Language or a subset of these.

2.4.3 taml:shared

The taml:shared element is representing the class 'shared' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in [TAM].
Compact Relax NG definition:

```
element taml:shared { shared_def }
shared_def =
      attribute * - taml:* { text }*,
      element taml:prescription { prescription_def }?,
      element taml:var { var_def }*,
      element taml:normativeSource { normativeSource_def }?,
      element taml:target { target_def }?,
      element taml:prerequisite { logicalexpr_def }?,
      element taml:predicate { logicalexpr_def }?,
      element taml:prescription { prescription_def }?,
      element taml:tag { tag_def }*,
      element taml:report { report_def }*,
      element { anyElement }*
```

The child element named '`shared`' of the `testAssertionSet` element **may** be used to provide one or more test assertion parts either as default values for missing parts in the test assertions defined for this set, or as overrides (either overridden by or overriding any corresponding parts of test assertions defined within the set) or as composites (composing as either conjunctions or disjunctions with any corresponding parts of the test assertions within the set) to all the test assertions of the test assertion set.
The '`normativeSource`', '`target`', '`predicate`', '`prerequisite`', '`prescription`', '`interpretation`', the identically named '`tag`' elements, the identically named '`var`' elements and the '`report`', elements, when they are children of this '`shared`' element, are extended with a '`conflict`' attribute which can take values as follows:

shared/normativeSource/@conflict = { conjunction | overriding | overridden }
shared/target/@conflict = { conjunction | disjunction | overriding | overridden }
shared/prerequisite/@conflict = { conjunction | disjunction | overriding | overridden }
shared/predicate/@conflict = { conjunction | disjunction | overriding | overridden }
shared/prescription/@conflict = { overriding | overridden }
shared/description/@conflict = { conjunction | disjunction | overriding | overridden }
shared/tag/@conflict = { conjunction | disjunction | overriding | overridden }
shared/var/@conflict = { conjunction | disjunction | overriding | overridden }
shared/report/@conflict = { conjunction | disjunction | overriding | overridden }

The values of the '`conflict`' attribute **may** be extended. Custom values **may** be ignored by an implementation. The base datatype of the custom extended enumeration for the '`conflict`' attribute is W3C XML Schema [XSD2] datatype 'Qname'. Custom enumerations **should** be prefixed with a namespace prefix associated with a namespace declared in the markup. Default namespaces (without a prefix) **shall not** be used.

2.4.4 taml:testAssertionSelection

The taml:testAssertionSelection element is representing the class 'testAssertionSelection' in the Test Assertions Model [TAM] . Detailed semantics about this class and its elements can be found in [TAM].
Compact Relax NG definition:

```
element taml:testAssertionSelection { testAssertionSelection_def }
testAssertionSelection_def =
       attribute name { xsd:normalizedString }?,
       attribute lg { xsd:normalizedString }?,
       attribute expr { xsd:normalizedString }?,
       attribute * - xsd:* { text }*,
       element taml:testAssertionSet { testAssertionSet_def }*,
       element taml:testAssertionIdList {
              element taml:testAssertionId { xsd:normalizedString }* }?
       text
```

A test assertion set in which references are made to existing test assertions defined outside of the test assertion set element (whether in the same document or other documents) shall use the `testAssertionRefList` child element to do so.
The attribute `taml:testAssertionRefList/@listname` gives a name to the reference list. More than one reference lists may be used inside a test assertion set element.
The attribute `taml:testAssertionRefList/@sourcedoc` may be used to specify a URL resolving to a document or a resource that contains the externally-defined test assertions referenced in this list.
The element taml:testAssertionRefList/taml:testAssertionRef identifies one externally-defined test assertion, and can be repeated.
The attribute 'expr' contains the selection criterion that corresponds to the 'content' attribute in the test assertion model.
The attribute 'lg' corresponds to the 'language' attribute in the test assertion model.
The element `taml:testAssertionSelection/taml:testAssertionIdList` identifies a list of test assertions by their `taml:testAssertion/@id` attribute value.
The element `taml:testAssertionSet` identifies the source set of test assertions, a subset of which must be selected, either by specifying a list of `taml:testAssertionId` instances, or by using the logical expression in @expr, or both.

2.4.5 taml:testAssertionRef

A test assertion set `may` refer to one or more test assertions by their test assertion identifiers to locate them in external resources, rather than include the test assertions literally within the set.
The taml:testAssertionRef element is identifying an externally-defined test assertion by its test assertion ID, and optionally by a document (@`sourcedoc`) where the test assertion is defined.
The taml:testAssertionRef element is representing the class 'testAssertionRef' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in [TAM].
Compact Relax NG definition:

```
element taml:testAssertionRef { testAssertionRef_def }
testAssertionRef_def =
       attribute taid { xsd:normalizedString },
       attribute name { xsd:normalizedString }?,
       attribute sourcedoc { xsd:anyURIlg { xsd:normalizedString }?,
       attribute * - tamlxsd:* { text }*,
       element taml:testAssertionResource { testAssertionResource_def }*,
       element taml:testAssertionSetId { xsd:normalizedString }*,
       element taml:testAssertionIdList {
              element taml:testAssertionId      { xsd:normalizedString }* }?
       text
```

The attribute `taml:testAssertionRef/@taid` is the ID of the referenced test assertion (id attribute in the taml:testAssertion element).

The attribute `taml:testAssertionRef/@name` is the name of the referenced test assertion if any, for convenience.
The attribute `taml:testAssertionRef/@sourcedoc` may be used to specify a URL resolving to a document or resource that contains the externally-defined and referenced test assertion. In case this attribute is also used over the parent `taml:testAssertionRefList`, the URL of the reference overrides the URL of the parent.
The value (string) of taml:testAssertionRef may be used for describing the referred test assertion, for convenience.

3 XML Schema

The following schema is called here the TAML schema:
A test assertion set may refer to one or more test assertions by their test assertion identifiers or by other means to locate them in external resources, rather than include the test assertions literally within the set.
A test assertion set in which references are made to other test assertions outside of the set (whether in the same document or other documents) shall use the testAssertionRef child element to do so.
The element taml:testAssertionRef/taml:testAssertionIdList identifies a list of referred test assertion by their taml:testAssertion/@id attribute value.
The element taml:testAssertionRef/taml:testAssertionSetId identifies a referred set of test assertion by its taml:testAssertionSet/@id attribute value.
The testAssertionRef **may** be used to refer to a test assertion set as a whole, rather than a reference to each test assertion individually.

3.1.1 taml:testAssertionResource

The taml:testAssertionResource element is representing the class 'testAssertionResource' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in [TAM].
Compact Relax NG definition:

```
element taml:testAssertionResource { testAssertionResource_def }
testAssertionResource_def =
      attribute name { xsd:normalizedString }?,
      attribute lg { xsd:normalizedString }?,
      attribute uri { xsd:normalizedString }?,
      attribute documentId { xsd:normalizedString }?,
      attribute * - xsd:* { text }*,
      text
```

The taml:testAssertionResource/@name attribute allows for giving a name to the external resource.
The content of the taml:testAssertionResource element allows for an informal description of the resource.

3.1.2 testAssertionDocumentHeader

The taml:testAssertionDocumentHeader element is representing the class 'testAssertionDocumentHeader' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in [TAM].

Compact Relax NG definition:

```
element taml:testAssertionDocumentHeader { testAssertionDocumentHeader_def }
testAssertionDocumentHeader_def =
      attribute * - xsd:* { text }*,
      element taml:common { common_def }
```

The testAssertionDocumentHeader element **may** be used to provide metadata (author, location, etc) about the specification to which test assertions are associated when such test assertions are interspersed within a document written with a markup other than Test Assertion Markup Language.
The testAssertionDocumentHeader element **may**, alternatively, provide a container for metadata about the specification in the outermost testAssertionSet of a test assertion document or where an implementation only allows one test assertion set for each document.

There **shall** be no more than one testAssertionDocumentHeader element used in any given document.

3.1.3 common

The taml:common element is representing the class 'common' in the Test Assertions Model [TAM]. Detailed semantics about this class and its elements can be found in [TAM].

Compact Relax NG definition:

```
element taml:testAssertionDocumentHeader { testAssertionDocumentHeader_def }
testAssertionDocumentHeader_def =
      attribute * - xsd:* { text }*,
      element taml:common { common_def }
common_def =
      element taml:sourceDocument { sourceDocument_def } ?,
      element taml:authors { authors_def } ?,
      element taml:location { location_def } ?
sourceDocument_def =
      ( element { anyElement }* | text )
authors_def =
      ( element { anyElement }* | text )
location_def =
      ( element { anyElement }* | text )
```

```
<xs:schema xmlns="http://docs.oasis-open.org/ns/tag/taml-201002/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://docs.oasis-open.org/ns/tag/taml-201002/" element-
FormDefault="qualified"
  attributeFormDefault="unqualified" version="1.0">
```

3.2 Reserved Tag Names

3.2.1 NormativeProperty

A test assertion **may** be tagged to show that it is used in defining a "property" of an implementation (e.g. a conformance profile) using the reserved word tag name NormativeProperty.

TA id: widget-TA104-2
Normative Source: specification requirement 104
Target: widget
Predicate: [the widget] is from 5 to 15 centimeters long in its longer dimension.
Prescription Level: mandatory
Tag: NormativeProperty = medium-sized

The Test Assertion Markup Language allows this to be represented as follows

```
      <testAssertion id="widget-TA104-2">
      . . .
            <predicate> [the widget] is from LENGTH-A to LENGTH-B long in its
longer dimension</predicate>
      . . .
            <tag name="NormativeProperty">medium-sized</tag>
      </testAssertion>
```

3.2.2 VersionAdd and VersionDrop

tag: VersionAdd: the lowest numerical version to which the test assertion applies.

~~`tag: VersionDrop`: the lowest numerical version number to which the test assertion does NOT apply.~~

~~Both `VersionAdd` and `VersionDrop` are optional tags. The absence of both tags **shall** mean that the test assertion is valid in all specification versions. If only a `VersionAdd` tag exists and its value is X, the test assertion will be valid in version X of the specification and all subsequent versions. If only a `VersionDrop` tag exists and its value is Y, the test assertion **shall** be valid in all versions of the specification prior to version Y. If both `VersionAdd` and `VersionDrop` tags exist, the test assertion **shall** be valid in version X and all subsequent versions up to but not including version Y. Based on these rules, the set of test assertions that apply to a specific version of the specification can be determined.~~

4 XML Schema

```
<xs:schema xmlns="http://docs.oasis-open.org/tag/ns/v1.0/taml/201002/"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://docs.oasis-open.org/tag/ns/v1.0/taml/201002/"
elementFormDefault="qualified"
      attributeFormDefault="unqualified" version="1.0">

  <xs:element name="testAssertion" type="testAssertion_type"/>
  <xs:element name="common" type="common_type"/>
  <xs:element name="testAssertionDocumentHeader" type="testAssertionDocument-
Header_type"/>
  <xs:element name="testAssertionSet" type="testAssertionSet_type"/>

  <xs:simpleType name="codeExtension_type">
    <xs:restriction base="xs:QName">
      <xs:pattern value="[\c-[:]]+:[\c-[:]]+"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="comment_type" mixed="true">
    <xs:sequenceimpleContent>
      <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any" pro-
cessContents="skip"/extension base="xs:string">
    </xs:sequence    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##other    <xs:anyAttribute namespace="##any"
processContents="skip"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="common_type" mixed="true">
    <xs:sequence>
      <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any" pro-
cessContents="skipelement name="sourceDocument" type="sourceDocument_type"
minOccurs="0"/>
      <xs:element name="authors" type="authors_type" minOccurs="0"/>
      <xs:element name="location" type="location_type" minOccurs="0"/>
      <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOc-
curs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="description_type" mixed="true">
    <xs:sequenceimpleContent>
      <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any" pro-
cessContents="skip"/extension base="xs:normalizedString">
    </xs:sequence    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##other    <xs:anyAttribute namespace="##any"
processContents="skip"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="descriptionConflictBaseCode_type">
```



```
    <xs:restriction base="xs:normalizedString">
      <xs:enumeration value="overriding"/>
      <xs:enumeration value="overridden"/>
      <xs:enumeration value="conjunction"/>
      <xs:enumeration value="disjunction"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="descriptionConflictCode_type">
    <xs:union memberTypes="descriptionConflictBaseCode_type
codeExtension_type"/>
  </xs:simpleType>

  <xs:complexType name="descriptionShared_type">
    <xs:simpleContent>
      <xs:extension base="description_type">
    <xs:attribute name="conflict" type="descriptionConflictCode_type"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="interpretation_type">
    <xs:simpleContent>
      <xs:extension base="xs:normalizedString">
        <xs:anyAttribute namespace="##other" processContents="skip
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="namespace_type">
    <xs:sequence>
      <xs:element name="prefix" type="xs:token"/>
      <xs:element name="uri" type="xs:anyURI"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="namespaces_type" mixed="true">
    <xs:sequence>
      <xs:element name="namespace" type="namespace_type" minOccurs="0" maxOc
curs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="normativeSource_type" mixed="true">
    <xs:sequence>
      <xs:element name="comment" type="comment_type" minOccurs="0"/>
      <xs:element name="interpretation" type="interpretation_type"
minOccurs="0"/>
      <xs:element name="refSourceItem" type="refSourceItem_type" minOccurs="0"
maxOccurs="unbounded"/>
          maxOccurs="unbounded"/>
      <xs:element name="textSourceItem" type="textSourceItem_type"
minOccurs="0"
      maxOccurs="unbounded"/>
      <xs:element name="derivedSourceItem" type="refSourceItem_type" minOc-
curs="0"
      maxOccurs="unbounded"/>
```



```
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##otherany" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="predicate_type" mixed="trusimpleType name="normativeSourceConflictBaseCode_type">
    <xs:sequencerestriction base="xs:normalizedString">
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unboundedenumeration value="overriding"/>
    </xs:sequence>  <xs:enumeration value="overridden"/>
    <xs:attribute name="lg" type="xs:normalizedString"><xs:enumeration value="conjunction"/>
    <xs:anyAttribute namespace="##other" processContents="skip"><xs:enumeration value="disjunction"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="normativeSourceConflictCode_type">
  <xs:union memberTypes="normativeSourceConflictBaseCode_type codeExtension_type"/>
 </xs:simpleType>

 <xs:complexType name="normativeSourceShared_type" mixed="true">
  <xs:complexContent>
   <xs:extension base="normativeSource_type">
  <xs:attribute name="conflict" type="normativeSourceConflictCode_type"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>

  <xs:complexType name="prerequisite_type" mixed="trudicate_type">
    <xs:sequenceimpleContent>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/extension base="xs:normalizedString">
    </xs:sequence>  <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:attribute name="lg" type="xs:normalizedString"><xs:anyAttribute namespace="##any" processContents="skip"/>
    <xs:anyAttribute namespace="##other" processContents="skip"/> </xs:extension>
  </xs:simpleContent>
 </xs:complexType>

 <xs:simpleType name="predicateConflictBaseCode_type">
  <xs:restriction base="xs:normalizedString">
   <xs:enumeration value="overriding"/>
   <xs:enumeration value="overridden"/>
   <xs:enumeration value="conjunction"/>
   <xs:enumeration value="disjunction"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="predicateConflictCode_type">
  <xs:union memberTypes="predicateConflictBaseCode_type codeExtension_type"/>
 </xs:simpleType>

 <xs:complexType name="predicateShared_type">
```



```
    <xs:simpleContent>
      <xs:extension base="predicate_type">
    <xs:attribute name="conflict" type="predicateConflictCode_type"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="prerequisite_type">
    <xs:simpleContent>
      <xs:extension base="xs:normalizedString">
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="prerequisiteConflictBaseCode_type">
    <xs:restriction base="xs:normalizedString">
      <xs:enumeration value="conjunction"/>
      <xs:enumeration value="disjunction"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="prerequisiteConflictCode_type">
    <xs:union memberTypes="prerequisiteConflictBaseCode_type
codeExtension_type"/>
  </xs:simpleType>

  <xs:complexType name="prerequisiteShared_type">
    <xs:simpleContent>
      <xs:extension base="prerequisite_type">
    <xs:attribute name="conflict" type="prerequisiteConflictCode_type"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="prescription_type">
    <xs:simpleContent>
      <xs:extension base="xs:normalizedString">
        <xs:attribute name="level" type="prescriptionLevelCode_type"/>
        <xs:anyAttribute namespace="##other    <xs:anyAttribute
namespace="##any" processContents="skip"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="prescriptionConflictBaseCode_type">
    <xs:restriction base="xs:normalizedString">
      <xs:enumeration value="overriding"/>
      <xs:enumeration value="overridden"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="prescriptionConflictCode_type">
    <xs:union memberTypes="prescriptionConflictBaseCode_type
codeExtension_type"/>
  </xs:simpleType>

  <xs:complexType name="prescriptionShared_type">
```



```
    <xs:simpleContent>
      <xs:extension base="prescription_type">
    <xs:attribute name="conflict" type="prescriptionConflictCode_type"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="prescriptionLevelCode_type">
    <xs:union memberTypes="prescriptionLevelBaseCode_type
codeExtension_type"/>
  </xs:simpleType>

  <xs:simpleType name="prescriptionLevelBaseCode_type">
    <xs:restriction base="xs:normalizedString">
      <xs:enumeration value="mandatory"/>
      <xs:enumeration value="permitted"/>
      <xs:enumeration value="preferred"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="refSourceItem_type">
    <xs:simpleContent>
      <xs:extension base="xs:normalizedString">
        <xs:attributeGroup ref="resource_attributeGroup    <xs:attribute
name="lg" type="xs:normalizedString"/>
        <xs:anyAttribute namespace="##other" processContents="ski <xs:attrib-
uteGroup ref="resource_attributeGroup"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="report_type" mixed="true">
    <xs:sequence>
      <xs:any namespace="##any" processContents="skip" minOccurs="0" maxOc-
curs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:attribute name="label" type="xs:normalizedString"/>
    <xs:attribute name="message" type="xs:normalizedString"/>
    <xs:attribute name="when" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##other any" processContents="skip"/>
  </xs:complexType>

  <xs:simpleType name="reportConflictBaseCode_type">
    <xs:restriction base="xs:normalizedString">
      <xs:enumeration value="conjunction"/>
      <xs:enumeration value="disjunction"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="reportConflictCode_type">
    <xs:union memberTypes="reportConflictBaseCode_type codeExtension_type"/>
  </xs:simpleType>

  <xs:complexType name="reportShared_type" mixed="true">
    <xs:complexContent>
      <xs:extension base="report_type">
    <xs:attribute name="conflict" type="reportConflictCode_type"/>
```



```
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:attributeGroup name="resource_attributeGroup">
    <xs:attribute name="srcname" type="xs:normalizedString"/>
    <xs:attribute name="uri" type="xs:anyURInormalizedString"/>
    <xs:attribute name="documentId" type="xs:normalizedString"/>
    <xs:attribute name="versionId" type="xs:normalizedString"/>
    <xs:attribute name="revisionId" type="xs:normalizedString"/>
    <xs:attribute name="resourceProvenanceId" type="xs:normalizedString"/>
  </xs:attributeGroup>

  <xs:complexType name="shared_type">
    <xs:sequence>
      <xs:element name="normativeSource" type="normativeSourceShared_type"
minOccurs="0"/>
      <xs:element name="target" type="targetShared_type" minOccurs="0"/>
      <xs:element name="prerequisite" type="prerequisiteShared_type" minOc-
curs="0"/>
      <xs:element name="predicate" type="predicateShared_type" minOccurs="0"/>
      <xs:element name="prescription" type="prescriptionShared_type" minOc-
curs="0"/>
      <xs:element name="description" type="descriptionShared_type"
minOccurs="0"/>
      <xs:element name="tag" type="tagShared_type" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="var" type="varShared_type" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="report" type="reportShared_type" minOccurs="0" maxOc-
curs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOc-
curs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="sourceDocument_type" mixed="true">
    <xs:sequence>
      <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOc-
curs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="revision" type="xs:normalizedString"/>
    <xs:attribute name="version" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="tag_type">
    <xs:simpleContent>
      <xs:extension base="xs:normalizedString">
        <xs:attribute name="tname" type="xs:normalizedString" use="required
    <xs:attribute name="name" type="xs:normalizedString"/>
        <xs:anyAttribute namespace="##other" processContents="skip
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
```



```
  <xs:complexType name="target_type" mixed="trusimpleType name="tagConflict-
BaseCode_type">
    <xs:sequencerestriction base="xs:normalizedString">
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOc-
curs="unboundedenumeration value="overriding"/>
    </xs:sequence>  <xs:enumeration value="overridden"/>
    <xs:attribute name="type" type="xs:normalizedString"> <xs:enumeration
value="conjunction"/>
    <xs:attribute name="lg" type="xs:normalizedString"> <xs:enumeration
value="disjunction"/>
    <xs:attribute name="idscheme" type="xs:normalizedString"/></xs:restriction>
   <xs:anyAttribute namespace="##other" processContents="skip"/></xs:simple-
Type>

 <xs:simpleType name="tagConflictCode_type">
   <xs:union memberTypes="tagConflictBaseCode_type codeExtension_type"/>
 </xs:simpleType>

 <xs:complexType name="tagShared_type">
   <xs:simpleContent>
     <xs:extension base="tag_type">
   <xs:attribute name="conflict" type="tagConflictCode_type"/>
     </xs:extension>
   </xs:simpleContent>
  </xs:complexType>

 <xs:complexType name="target_type">
   <xs:simpleContent>
     <xs:extension base="xs:normalizedString">
   <xs:attribute name="type" type="xs:normalizedString"/>
   <xs:attribute name="lg" type="xs:normalizedString"/>
   <xs:attribute name="idscheme" type="xs:normalizedString"/>
   <xs:anyAttribute namespace="##any" processContents="skip"/>
     </xs:extension>
   </xs:simpleContent>
 </xs:complexType>

 <xs:simpleType name="targetConflictBaseCode_type">
   <xs:restriction base="xs:normalizedString">
     <xs:enumeration value="overriding"/>
     <xs:enumeration value="overridden"/>
     <xs:enumeration value="conjunction"/>
     <xs:enumeration value="disjunction"/>
   </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="targetConflictCode_type">
   <xs:union memberTypes="targetConflictBaseCode_type codeExtension_type"/>
 </xs:simpleType>

 <xs:complexType name="targetShared_type" mixed="true">
   <xs:simpleContent>
     <xs:extension base="target_type">
   <xs:attribute name="conflict" type="targetConflictCode_type"/>
     </xs:extension>
   </xs:simpleContent>
 </xs:complexType>

  <xs:complexType name="testAssertion_type">
```



```
    <xs:sequence>
      <xs:element name="description" type="description_type" minOccurs="0"/>
      <xs:element name="normativeSource" type="normativeSource_type" minOc-
curs="0"/>
      <xs:element name="var" type="var_type" minOccurs="0" maxOccurs="unboun-
ded"/>
      <xs:element name="target" type="target_type" minOccurs="0"/>
      <xs:element name="prerequisite" type="prerequisite_type" minOccurs="0"/>
      <xs:element name="predicate" type="predicate_type" minOccurs="0"/>
      <xs:element name="prescription" type="prescription_type" minOccurs="0"/>
      <xs:element name="report" type="report_type" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="tag" type="tag_type" minOccurs="0" maxOccurs="unboun-
ded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOc-
curs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:normalizedString" use="required"/>
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:attribute name="name" type="xs:normalizedString"/>
    <xs:attribute name="schemaVersionId" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##other~~any~~" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="testAssertionRefList~~DocumentHeader~~_type">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="testAssertion-
Ref"~~name="common" type="common_type"/>~~
        type="testAssertionRef_type~~<xs:any namespace="##other" processCon-
tents="skip" minOccurs="0" maxOccurs="unbounded~~"/>
    ~~</xs:sequence>~~
  ~~</xs:complexType>~~

  ~~<xs:complexType name="testAssertionRef_type">~~
    ~~<xs:sequence>~~
      ~~<xs:choice>~~
    ~~<xs:element name="testAssertionResource" type="testAssertionResource_type"~~
          ~~minOccurs="0" maxOccurs="unbounded"/>~~
    ~~<xs:element maxOccurs="unbounded" name="testAssertionSetId"~~
          ~~type="xs:normalizedString" minOccurs="0"/>~~
    ~~<xs:element minOccurs="0" name="testAssertionIdList">~~
      ~~<xs:complexType>~~
        ~~<xs:sequence>~~
          ~~<xs:element maxOccurs="unbounded" name="testAssertionId"~~
              ~~type="xs:normalizedString" minOccurs="0"/>~~
        ~~</xs:sequence>~~
      ~~</xs:complexType>~~
    ~~</xs:element>~~
      ~~</xs:choice>~~
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOc-
curs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="sourcedoc" type="xs:anyURI~~lg" type="xs:normalized-
String~~"/>
    <xs:attribute name="name" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##other~~any~~" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="testAssertionRef~~source~~_type">
```



```
    <xs:simpleContentattribute name="lg" type="xs:normalizedString"/>
      <xs:extension base="xs:string"><xs:attributeGroup ref="resource_attrib-
uteGroup"/>
      <xs:attribute name="taid" type="xs:normalizedString<xs:anyAttribute-
namespace="##any" processContents="skip"/>
      <xs:attribute name="name" type="xs:normalizedString"/</xs:complexType>
      <xs:attribute name="sourcedoc" type="xs:anyURI"/>
      <xs:anyAttribute namespace="##other" processContents="skip"/<xs:com-
plexType name="testAssertionSelection_type">
    </xs:extension<xs:sequence>
    </xs:simpleContent><xs:choice>
    <xs:element maxOccurs="unbounded" name="testAssertionSetId"
        type="xs:normalizedString" minOccurs="0"/>
    <xs:element minOccurs="0" name="testAssertionIdList">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="testAssertionId"
            type="xs:normalizedString" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
      </xs:choice>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOc-
curs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:normalizedString"/>
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:attribute name="expr" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="testAssertionSet_type">
    <xs:sequence>
      <xs:element minOccurs="0" name="common" type="common_typeref="test-
AssertionDocumentHeader" minOccurs="0"/>
      <xs:element name="testAssertionRefList"
type="testAssertionRefList_type" minOccurs="0"shared" type="shared_type"-
minOccurs="0"/>
        maxOccurs="unbounded"/><xs:element name="testAssertionRef" type="tes-
tAssertionRef_type" minOccurs="0"
      <xs:element maxOccurs="unbounded" minOccurs="0" name="testAssertion"
type="testAssertion_type          maxOccurs="unbounded"/>
      <xs:element ref="testAssertionSet" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="testAssertionSelection" minOccurs="0"
        type="testAssertionSelection_type" maxOccurs="unbounded"/>
      <xs:element minOccurs="0" name="testAssertionList">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="testAssertion" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
      </xs:element>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOc-
curs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="setid" type="xs:normalizedString"/>
    <xs:attribute name="setnamelg" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##other" processContents="skipttribute-
name="schemaVersionId" type="xs:normalizedString"/>
```



```
    <xs:attribute name="time" type="xs:time"/>
    <xs:attribute name="date" type="xs:date"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="textSourceItem_type">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="extension" type="xs:boolean"/>
        <xs:attribute name="txt <xs:attribute name="name" type="xs:normal-
izedString"/>
        <xs:anyAttribute namespace="##other" processContents="skip
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="var_type" mixed="true">
    <xs:sequenceimpleContent>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOc-
curs="unbounded"/extension base="xs:normalizedString">
    </xs:sequence  <xs:attribute name="name" type="xs:normalizedString"/>
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:attribute name="vname" type="xs:normalizedString" use="required
    <xs:anyAttribute namespace="##any" processContents="skip"/>
    <xs:attribute name="vtype" type="xs:normalizedString"/  </xs:exten-
sion>
    <xs:anyAttribute namespace="##other" processContents="skip"/</xs:simple-
Content>
  </xs:complexType>

</xs:schema  <xs:simpleType name="varConflictBaseCode_type">
    <xs:restriction base="xs:normalizedString">
      <xs:enumeration value="overriding"/>
      <xs:enumeration value="overridden"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="varConflictCode_type">
    <xs:union memberTypes="varConflictBaseCode_type codeExtension_type"/>
  </xs:simpleType>

  <xs:complexType name="varShared_type">
    <xs:simpleContent>
      <xs:extension base="var_type">
    <xs:attribute name="conflict" type="varConflictCode_type"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="document_type">
    <xs:simpleContent>
      <xs:extension base="xs:normalizedString">
    <xs:attribute name="label"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="authors_type" mixed="true">
    <xs:sequence>
```



```
      <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
  </xs:complexType>
  <xs:complexType name="location_type" mixed="true">
    <xs:sequence>
      <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
  </xs:complexType>
</xs:schema>
```


5 Conformance

Test assertion artifacts or implementations subject to conformance to this XML markup are of two kinds:
(a) XML Test assertion
(b) XML Test assertion Set

5.1 Conformance Clause for XML Test Assertion

An XML Test Assertion is said to be strictly conforming if it is a Test Assertion Markup Language testAssertion element that:

- Is valid according to the TAML Schema (Section 3)
- Does not make use of any extension element or attribute allowed by the TAML Schema.
- Satisfies all normative mandatory provisions ("shall", "shall not" keywords) in Sections 2.3 (Test Assertion).
- Uses the markup in compliance with the general semantics of a test assertion and its parts as described in the Test Assertions Model [TAM] specification.

An XML Test Assertion is said to be conforming if it is a Test Assertion Markup Language testAssertion element that:

- Is valid according to the TAML Schema (Section 3)
- May use any extension element or attribute allowed by the TAML Schema.
- Satisfies all normative mandatory provisions ("shall", "shall not" keywords) in Sections 2.3 (Test Assertion).
- Uses the markup in compliance with the general semantics of a test assertion and its parts as described in the Test Assertions Model [TAM] specification.
- If the test assertion makes use of extension elements, a derived test assertion obtained by removing all extensions is still a strictly conforming test assertion that uses the markup in compliance with the general semantics of a test assertion.

5.2 Conformance Clause for XML Test Assertion Set

An XML Test Assertion Set is said to be strictly conforming if it is a Test Assertion Markup Language testAssertionSet element that:

- Is valid according to the XML Schema (Section 3)
- Does not make use of any extension element or attribute allowed by the TAML Schema (except for the <common> element and its content)..
- Only contains or refers to strictly conforming test assertions.
- Satisfies all normative mandatory provisions ("shall", "shall not" keywords) in Sections 2.3 (Test Assertion), (2.4 Test Assertion Set).
- If it has a <taml:common> element, only uses children elements in it that would qualify as children elements of strictly conforming testAssertion elements.

An XML Test Assertion Set is said to be conforming if it is a Test Assertion Markup Language testAssertionSet element that:

- Is valid according to the XML Schema (Section 3)
- May use any extension element or attribute allowed by the TAML Schema.
- Satisfies all normative mandatory provisions ("shall", "shall not" keywords) in Sections 2.3 (Test Assertion), 2.4 (Test Assertion Set).

~~A Conforming Test Assertion is a Test Assertion Markup Language testAssertion element that:~~

- ~~is valid according to the XML Schema (Section 3)~~
- ~~satisfies all normative mandatory provisions ("must", "must not", "shall", "shall not" keywords) in Sections 2.3 Test Assertion and 2.5 Reserved Tag Names.~~
- ~~uses the mark-up in compliance with the general semantics of a test assertion and its parts as described in the Test Assertions Model [TAM] specification.~~

~~A Conforming Test Assertion Set is a Test Assertion Markup Language testAssertionSet element that:~~

- ~~is valid according to the XML Schema (Section 3)~~

- ~~satisfies all normative mandatory provisions ("must", "must not", "shall", "shall not" keywords) in Sections 2.3 Test Assertion, 2.4 Test Assertion Set, and 2.5 Reserved Tag Names.~~
- ~~uses the mark-up in compliance with the general semantics of a test assertion set as described in the Test Assertions Model [TAM] specification.~~

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged

Participants:

- David Pawson, Royal National Institute for the Blind
- Dennis Hamilton, Individual
- Dmitry Kostovarov, Oracle Corporation
- Dong-Hoon Lim, KIEC
- Hyunbo Cho, Pohang University
- Jacques Durand, Fujitsu
- Kevin Looney, Oracle Corporation
- Kyoung-Rog Yi, KIEC
- Lynne Rosenthal, NIST
- Patrick Curran, Oracle Corporation
- Paul Rank, Oracle Corporation
- Serm Kulvatunyou, NIST
- Stephen D. Green, Document Engineering Services
- Tim Boland, NIST
- Victor Rudometov, Oracle Corporation
- Youngkon Lee, Korea TAG forum

Appendix B. Revision History

Rev	Date	By Whom	What
CD 1	02/10/10	Stephen Green	CD 1 draft for PR #1
CD 2	08/10/10	Jacques Durand	CD 2 draft for PR #2
CD 3	04/24/11	Jacques Durand	CD 3 draft for PR #3. Simplified Test Assertions Set section, modified the schema accordingly.
CSD 4	05/20/11	Jacques Durand	candidate draft for PR #3. Various edits.
CSD 5	06/04/11	Jacques Durand	candidate draft for PR #3. Additional edits: - renamed several occurrences of attr @name in schema, in order to distinguish them fro TA name. - removed section 2.5 that is redundant with same content in TA model, and just restated the reserved tag names. - added subsections for each part of normativeSource. - added NIEM reference. - other editorial improvements sugegsted by Dennis.
CSD 6	08/29/11	Jacques Durand	candidate draft for PR #3. Additional edits: - several editorial improvements, including better abstracts in front page.. - added @vtype to the var element.
~~Rev~~	~~Date~~	~~By Whom~~	~~What~~
~~CD 1~~	~~02/10/10~~	~~Stephen Green~~	~~CD 1 draft for PR #1~~
~~CD 2~~	~~08/10/10~~	~~Jacques Durand~~	~~CD 2 draft for PR #2~~