Test Assertions Part 2 - Test Assertion Markup Language Version 1.0

Committee Specification 01

29 November 2010

Specification URIs:

This Version:

http://docs.oasis-open.org/tag/taml/v1.0/cs01/testassertionmarkuplanguage-1.0-cs-01.html

http://docs.oasis-open.org/tag/taml/v1.0/cs01/testassertionmarkuplanguage-1.0-cs-01.odt

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

Previous Version:

http://docs.oasis-open.org/tag/taml/v1.0/cd02/testassertionmarkuplanguage-1.0-cd-02.html

http://docs.oasis-open.org/tag/taml/v1.0/cd02/testassertionmarkuplanguage-1.0-cd-02.odt

http://docs.oasis-open.org/tag/taml/v1.0/cd02/testassertionmarkuplanguage-1.0-cd-02.pdf (Authoritative)

Latest Version:

http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0.html

http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0.odt

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

Technical Committee:

OASIS Test Assertions Guidelines (TAG)

Chair(s):

Patrick Curran

Jacques Durand

Editor(s):

Stephen D Green

Related Work:

This specification is related to:

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 defines a markup for writing test assertions.

Status:

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.

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

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS", “Test Assertion Markup Language”, “OASIS TAML” are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

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 01, "Test Assertions Model Version 1.0", November 2010 http://docs.oasis-open.org/tag/model/v1.0/cs01/testassertionsmodel-1.0-cs-01.pdf

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

[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.

1.3 Non-normative References

[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] 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
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

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.

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,** representating the namespace: http://docs.oasis-open.org/tag/ns/v1.0/taml/201002/.

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 of their containing element, e.g.: taml:testAssertion/@id for the attribute 'id' of the taml:testAssertion element.

2.3 Test Assertion

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 [TAM],

Compact Relax NG definition:

```
element taml:testAssertion { testAssertion_def }
testAssertion_def =
   attribute id { xsd:normalizedString }?,
   attribute lg { NCName }?,
   attribute schemaVersionId { xsd:normalizedString }?,
   attribute * { 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 }*
```

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.

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 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="XPath2.0"
      xmlns:taml="http://docs.oasis-
open.org/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>
```

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 [TAM].

Compact Relax NG definition:

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

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 name { xsd:normalizedString }?,
   attribute lg { NCName }?,
   attribute uri { xsd:normalizedString }?,
   attribute documentId { xsd:normalizedString }?,
   attribute versionId { xsd:normalizedString }?,
   attribute revisionId { xsd:normalizedString }?,

   attribute resourceProvenanceId { xsd:normalizedString }?,
   attribute * - 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 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.

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 name { xsd:normalizedString }?,
   attribute lg { NCName }?,
   attribute * - xsd:* { text }*,
   text
```

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 lg { NCName }?,
   attribute * - xsd:* { text }*,
   text
```

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 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.

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 [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 * - xsd:* { text }*,
   text ?
```

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 targets to which the test assertion applies. This content may also be a textual representation of the target instance(s) under consideration. .

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 [TAM].

Compact Relax NG definition:

```
element taml:prerequisite { logicalexpr_def }
logicalexpr_def =
   attribute lg { xsd:normalizedString }?,
   attribute * - xsd:* { text }*,
   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.

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 [TAM].

Compact Relax NG definition:

```
element taml:predicate { logicalexpr_def }
logicalexpr_def =
   attribute lg { xsd:normalizedString }?,
   attribute * - xsd:* { text }*,
   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 (see later).

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 [TAM].

Compact Relax NG definition:

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

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 [TAM].

Compact Relax NG definition:

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

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 [TAM].

Compact Relax NG definition:

```
element taml:tag { tag_def }
tag_def =
   attribute name { xsd:normalizedString },
   attribute 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/@name attribute **shall** be used in a tag element and have a non-empty value.

taml:var

The taml:var element is representing the class 'var' 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 name { xsd:normalizedString }?,
   attribute 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.[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'.

taml: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:NCName }?,
   attribute message { text }?,
   attribute when { text }?,
   attribute lg { xsd:normalizedString }?,
   attribute * - xsd:* { text }*,
   text
```

The `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, say, HTML or other simple formatting markup.

The attribute `label` **shall** allow values '`fail`', '`pass`' and '`notQualified`' as content, corresponding to the standard possible outcomes defined in the Test Assertions Model [TAM] as follows

- `notQualified` corresponds to the outcome " **`Target not qualified`**".
- `pass` corresponds to the outcome "**`Normative statement fulfilled [by the Target]`**".
- 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

taml:testAssertionSet

The taml:testAssertionSet element is 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 id { xsd:normalizedString }?,
   attribute lg { NCName }?,
   attribute schemaVersionId { xsd:normalizedString }?,
   attribute * - taml:* { text }*,
   element taml:testAssertionDocumentHeader { testAssertionDocumentHeader _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 reference 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.

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.

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
```

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.

taml:testAssertionRef

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 name { xsd:normalizedString }?,
   attribute lg { xsd:normalizedString }?,
```

```
   attribute * - xsd:* { text }*,
   element taml:testAssertionResource { testAssertionResource_def }*,
   element taml:testAssertionSetId { xsd:normalizedString }*,
   element taml:testAssertionIdList {
      element taml:testAssertionId  { xsd:normalizedString }* }?
   text
```

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.

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.

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.

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 )
```

2.5 Reserved Tag Names

`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>
```

`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.

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

  <xs:complexType name="common_type">
    <xs:sequence>
      <xs:element 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"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="description_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="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: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"
maxOccurs="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"/>
      <xs:element name="textSourceItem" type="textSourceItem_type"
minOccurs="0"
          maxOccurs="unbounded"/>
      <xs:element name="derivedSourceItem" type="refSourceItem_type"
minOccurs="0"
          maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
  </xs:complexType>

  <xs:simpleType name="normativeSourceConflictBaseCode_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="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="predicate_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="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="##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:attribute name="lg" type="xs:normalizedString"/>
```

```
    <xs:attributeGroup 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"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="label" type="xs:normalizedString"/>
    <xs:attribute name="message" type="xs:normalizedString"/>
    <xs:attribute name="when" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##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="name" type="xs:normalizedString"/>
    <xs:attribute name="uri" type="xs:normalizedString"/>
    <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"
minOccurs="0"/>
      <xs:element name="predicate" type="predicateShared_type" minOccurs="0"/>
      <xs:element name="prescription" type="prescriptionShared_type"
minOccurs="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"
maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="sourceDocument_type" mixed="true">
    <xs:sequence>
      <xs:any namespace="##other" processContents="skip" minOccurs="0"
maxOccurs="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="name" type="xs:normalizedString"/>
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="tagConflictBaseCode_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="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"
minOccurs="0"/>
      <xs:element name="var" type="var_type" minOccurs="0"
maxOccurs="unbounded"/>
      <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="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:normalizedString"/>
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:attribute name="schemaVersionId" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="testAssertionDocumentHeader_type">
    <xs:sequence>
      <xs:element name="common" type="common_type"/>
      <xs:any namespace="##other" processContents="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"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:attribute name="name" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="testAssertionResource_type">
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:attributeGroup ref="resource_attributeGroup"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="testAssertionSelection_type">
    <xs:sequence>
      <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"
maxOccurs="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 ref="testAssertionDocumentHeader" minOccurs="0"/>
      <xs:element name="shared" type="shared_type" minOccurs="0"/>
      <xs:element name="testAssertionRef" type="testAssertionRef_type"
minOccurs="0"
          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"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:normalizedString"/>
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:attribute 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="name" type="xs:normalizedString"/>
    <xs:attribute name="lg" type="xs:normalizedString"/>
    <xs:anyAttribute namespace="##any" processContents="skip"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

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

  <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>
```

4 Conformance

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