Test Assertions Part 2 - Test Assertion Markup Language Version 1.0

Committee Specification Draft 03

2 May 2011

Specification URIs:

This Version:
http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0-csd03.html
http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0-csd03.odt
http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0-csd03.pdf
(Authoritative)

Previous Version:
http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0-csd02.html
http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0-csd02.odt
http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0-csd02.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

Copyright © OASIS® 2008-2010. All Rights Reserved.

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.

Testassertionmarkuplanguage-1.0-csd-03
Copyright © OASIS® 2011. All Rights Reserved.
# Table of Contents

1 Introduction ........................................................................................................................................... 5  
   1.1 Terminology ........................................................................................................................................ 5  
   1.2 Normative References .......................................................................................................................... 5  
   1.3 Non-normative References .................................................................................................................... 5  
2 Markup Representation of Test Assertions .................................................................................................. 6  
   2.1 Binding to Test Assertions, Part 1 Test Assertions Model ..................................................................... 6  
   2.2 Conventions Used in the XML Markup and its Usage ........................................................................... 6  
   2.3 Test Assertion .......................................................................................................................................... 7  
      2.3.1 taml:testAssertion .......................................................................................................................... 7  
      2.3.2 taml:normativeSource ..................................................................................................................... 8  
      2.3.3 taml:target ....................................................................................................................................... 10  
      2.3.4 taml:prerequisite ............................................................................................................................ 11  
      2.3.5 taml:predicate .................................................................................................................................. 11  
      2.3.6 taml:prescription ............................................................................................................................. 12  
      2.3.7 taml:description ................................................................................................................................ 12  
      2.3.8 taml:tag ............................................................................................................................................ 13  
      2.3.9 taml:var .......................................................................................................................................... 13  
      2.3.10 taml:report ..................................................................................................................................... 13  
   2.4 Test Assertion Set ..................................................................................................................................... 14  
      2.4.1 taml:testAssertionSet ...................................................................................................................... 15  
      2.4.2 taml:testAssertionRefList .............................................................................................................. 15  
      2.4.3 taml:testAssertionRef .................................................................................................................... 16  
   2.5 Reserved Tag Names ............................................................................................................................... 16  
      2.5.1 NormativeProperty ......................................................................................................................... 16  
      2.5.2 VersionAdd and VersionDrop ......................................................................................................... 17  
3 XML Schema ............................................................................................................................................... 18  
4 Conformance ............................................................................................................................................... 23  
   4.1 Conformance Clause for XML Test Assertion ....................................................................................... 23  
   4.2 Conformance Clause for XML Test Assertion Set .................................................................................. 23  
Appendix A.Acknowledgments ..................................................................................................................... 25  
Appendix B.Revision History ........................................................................................................................ 26
1 Introduction

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


1.3 Non-normative References

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).

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:

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Markup Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class: testAssertion</td>
<td>element: testAssertion</td>
</tr>
<tr>
<td>attribute: id</td>
<td>attribute: id</td>
</tr>
<tr>
<td>attribute: language</td>
<td>attribute: lg</td>
</tr>
<tr>
<td>Class: normativeSource</td>
<td>element: normativeSource</td>
</tr>
<tr>
<td>Class: target</td>
<td>element: target</td>
</tr>
<tr>
<td>Class: predicate</td>
<td>element: predicate</td>
</tr>
<tr>
<td>Class: prerequisite</td>
<td>element: prerequisite</td>
</tr>
<tr>
<td>Class: tag</td>
<td>element: tag</td>
</tr>
<tr>
<td>Class: variable</td>
<td>element: var</td>
</tr>
<tr>
<td>Class: description</td>
<td>element: description</td>
</tr>
<tr>
<td>Class: prescription</td>
<td>element: prescription</td>
</tr>
</tbody>
</table>

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','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, representing the namespace: http://docs.oasis-open.org/ns/tag/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 markup 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

#### 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.

Compact Relax NG definition:

```
<element taml:testAssertion { testAssertion_def }>
 testAssertion_def =
   attribute id { xsd:normalizedString }?,
   attribute lg { xsd:normalizedString }?,
   attribute schemaVersionId { xsd:normalizedString }?,
   attribute * - taml:* { text }*,
   element taml:description { description_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 }*
```

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 expressions, if any, unless shadowed by a similar 'lg' attribute in the part. A profile of this specification **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 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 or inferred from the context of the test assertion.

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 extension to the model.

Like many of the elements in the Test Assertion Markup Language, the `testAssertion` element allows extensions both in its attributes and its children elements. Additional attributes and elements shall be in a namespace other than the `taml` namespace (as indicated here in compact RelaxNG by using the subtraction: - `taml:*`). This rule applies to all extensions allowed in this markup.

**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:

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

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.
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:derivedSourceItem { refSourceItem_def }*,
  element * - taml:* { anyElement }* & text
```

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 taml:normativeSource element allows for mixed content.

Compact Relax NG definition:

```

  element taml:refSourceItem { refSourceItem_def }
  refSourceItem_def =
    attribute name { xsd:normalizedString }?,
    attribute lg { xsd:normalizedString }?,
    attribute uri { xsd:anyURI }?,
    attribute documentId { xsd:normalizedString }?,
    attribute versionId { xsd:normalizedString }?,
    attribute revisionId { xsd:normalizedString }?,
    attribute resourceProvenanceId { xsd:normalizedString }?,
    attribute * - taml:* { 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 other metadata attributes include 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 { xsd:normalizedString }?,
    attribute * - taml:* { 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 { xsd:normalizedString }?,
    attribute * - taml:* { text }*,
    element * { anyElement }* & text
</element>
```

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.

Compact Relax NG definition:

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

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.

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:* { text }*,
    element * - taml:* { anyElement }* & text
</element>
```
A target can either be a specific item or a category of items. The 'target' element has a 'type' attribute which 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 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. The taml:target element allows for mixed content.

### 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.

Compact Relax NG definition:

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

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.

Compact Relax NG definition:

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

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.

### 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.

Compact Relax NG definition:

```xml
element taml:prescription { prescription_def }

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

### 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.

Compact Relax NG definition:

```xml
element taml:description { description_def }

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

Compact Relax NG definition:

```
<element taml:tag { tag_def }>
  <tag_def>
    <attribute name { xsd:normalizedString }>,
    <attribute lg { xsd:normalizedString }>,?
    <attribute * - taml:* { 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.

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.

Compact Relax NG definition:

```
<element taml:var { var_def }>
  <var_def>
    <attribute name { xsd:normalizedString }>,
    <attribute lg { xsd:normalizedString }>,?
    <attribute * - taml:* { text }*,
    <element * - taml:* { anyElement }*,
```

The content of the taml:var element is representing the "content" attribute of the corresponding class ‘variable’ in the model.

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.

2.3.10 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>
```

Testassertionmarkuplanguage-1.0-csd-03
Copyright © OASIS® 2011. All Rights Reserved.
28 April 2011
Page 13 of 26
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).

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 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:

- notQualified corresponds to the target evaluation outcome "Target not qualified".
- pass corresponds to the target evaluation 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 which may be defined in a conformance profile.

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. This attribute is also useful 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 general meaning of the assertion evaluation outcome, e.g. provide a standard error message. A more detailed report or 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. The taml:report element allows for mixed content.

### 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.

Compact Relax NG definition:

```
<element taml:testAssertionSet { testAssertionSet_def }

testAssertionSet_def =
  attribute id { xsd:normalizedString }?,
  attribute name { xsd:normalizedString }?,
  attribute * - taml:* { text }*,
  element taml:common { common_def }?,
  element taml:testAssertionRefList { testAssertionRefList_def } *,
  element taml:testAssertion { testAssertion_def } *

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

The attribute `tml:testAssertionSet/@name` gives a name to the test assertions set. The attribute `tml:testAssertionSet/@id` 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 `tml: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 `tml:common` element allows for mixed content.

2.4.2 taml:testAssertionRefList

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

Compact Relax NG definition:

```
<element taml:testAssertionRefList { testAssertionRefList_def }

testAssertionRefList_def =
  attribute name { xsd:normalizedString }?,
  attribute sourcedoc { xsd:anyURI }?,
  attribute * - taml:* { text }*,
  element taml:testAssertionRef { testAssertionRef_def } *,
  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/@name` 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.

### 2.4.3 `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.

Compact Relax NG definition:

```xml
element taml:testAssertionRef { testAssertionRef_def }

testAssertionRef_def =
  attribute taid { xsd:normalizedString },
  attribute name { xsd:normalizedString }?,
  attribute sourcedoc { xsd:anyURI }?,
  attribute * - taml:* { text }*,
  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` gives a name to the test assertion reference, 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.

### 2.5 Reserved Tag Names

#### 2.5.1 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

```xml
<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>
```

### 2.5.2 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.
3 XML Schema

The following schema is called here the TAML schema:

```xml
<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/
    elementFormDefault="qualified"
    attributeFormDefault="unqualified" version="1.0">

    <xs:element name="testAssertion" type="testAssertion_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:sequence>
            <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any"
                processContents="skip"/>
            <xs:attribute name="lg" type="xs:normalizedString"/>
            <xs:anyAttribute namespace="##other" processContents="skip"/>
        </xs:sequence>
    </xs:complexType>

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

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

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

    <xs:complexType name="namespace_type">
        <xs:sequence>
            <xs:element name="testAssertion" type="testAssertion_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:sequence>
                    <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any"
                        processContents="skip"/>
                    <xs:attribute name="lg" type="xs:normalizedString"/>
                    <xs:anyAttribute namespace="##other" processContents="skip"/>
                </xs:sequence>
            </xs:complexType>

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

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

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

            <xs:complexType name="namespace_type">
                <xs:sequence>
                    <xs:element name="testAssertion" type="testAssertion_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:sequence>
                            <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any"
                                processContents="skip"/>
                            <xs:attribute name="lg" type="xs:normalizedString"/>
                            <xs:anyAttribute namespace="##other" processContents="skip"/>
                        </xs:sequence>
                    </xs:complexType>

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

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

                    <xs:complexType name="interpretation_type">
                        <xs:simpleContent>
                            <xs:extension base="xs:normalizedString">
                                <xs:attribute name="lg" type="xs:normalizedString"/>
                                <xs:anyAttribute namespace="##other" processContents="skip"/>
                            </xs:extension>
                        </xs:simpleContent>
                    </xs:complexType>
                </xs:complexType>
            </xs:complexType>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```
<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="##other" 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="lg" type="xs:normalizedString"/>
  <xs:attribute name="label" type="xs:NCName"/>
  <xs:attribute name="message" type="xs:normalizedString"/>
  <xs:attribute name="when" type="xs:normalizedString"/>
  <xs:anyAttribute namespace="##other" processContents="skip"/>
</xs:complexType>

<xs:attributeGroup name="resource_attributeGroup">
  <xs:attribute name="name" type="xs:normalizedString"/>
  <xs:attribute name="uri" type="xs:anyURI"/>
  <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="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="##other" processContents="skip"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="target_type" mixed="true">
  <xs:sequence>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="type" type="xs:normalizedString"/>
  <xs:attribute name="lg" type="xs:normalizedString"/>
</xs:complexType>
4 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

4.1 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) and 2.5 (Reserved Tag Names).
- 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) and 2.5 (Reserved Tag Names).
- 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.

4.2 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), and 2.5 (Reserved Tag Names).
- 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), and 2.5 (Reserved Tag Names).
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

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>By Whom</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD 1</td>
<td>02/10/10</td>
<td>Stephen Green</td>
<td>CD 1 draft for PR #1</td>
</tr>
<tr>
<td>CD 2</td>
<td>08/10/10</td>
<td>Jacques Durand</td>
<td>CD 2 draft for PR #2</td>
</tr>
<tr>
<td>CD 3</td>
<td>04/24/11</td>
<td>Jacques Durand</td>
<td>CD 3 draft for PR #3. Simplified Test Assertions Set section, modified the schema accordingly.</td>
</tr>
</tbody>
</table>