



Test Assertions Part 2 - Test Assertion Markup Language Version 1.0

Committee Specification Draft 03

08 September 2011

Specification URIs:

This version:

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

Previous version:

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

Latest version:

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

Technical Committee:

OASIS Test Assertions Guidelines (TAG) TC

Chair:

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

Editors:

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

Additional Work Product artifacts:

This prose specification is one component of a Work Product which also includes:
XML schema: [taml/v1.0/csd03/xsd/testAssertionMarkupLanguage.xsd](http://docs.oasis-open.org/tag/taml/v1.0/csd03/xsd/testAssertionMarkupLanguage.xsd)

Related work:

This specification is related to:

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

Declared XML namespaces:

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

Abstract:

This document defines an XML vocabulary for representing test assertions aligned with the Test Assertions Model.

Status:

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

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

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

Citation format:

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

[TAML]

Test Assertions Part 2 - Test Assertion Markup Language Version 1.0. 08 September 2011.
OASIS Committee Specification Draft 03. <http://docs.oasis-open.org/tag/taml/v1.0/csd03/testassertionmarkuplanguage-1.0-csd03.html>.

Notices

Copyright © OASIS Open 2011. 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" and "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.....	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.....	11
2.3.7	taml:description.....	12
2.3.8	taml:tag.....	12
2.3.9	taml:var.....	12
2.3.10	taml:report.....	13
2.4	Test Assertion Set.....	13
2.4.1	taml:testAssertionSet.....	13
2.4.2	taml:testAssertionRefList.....	14
2.4.3	taml:testAssertionRef.....	14
3	XML Schema.....	16
4	Conformance.....	21
4.1	Conformance Clause for XML Test Assertion.....	21
4.2	Conformance Clause for XML Test Assertion Set.....	21
Appendix A.	Acknowledgments.....	22
Appendix B.	Revision History.....	23

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

- [TAM]** OASIS Committee Specification Draft 02, "Test Assertions, Part 1 Test Assertions Model Version 1.0", May 2011 <http://docs.oasis-open.org/tag/model/v1.0/testassertionsmodel-1.0.pdf>
- [ISO/IEC Directives]** ISO/IEC Directives, Part 2 Rules for the structure and drafting of International Standards, International Organization for Standardization, 2004. http://www.iso.org/iso/standards_development/processes_and_procedures/iso_iec_directives_and_iso_supplement.htm
- [XSD1]** Henry S. Thompson, David Beech, Murray Maloney, et. al., editors. XML Schema Part 1: Structures. <http://www.w3.org/TR/xmlschema-1/> . World Wide Web Consortium, 2000.
- [XSD2]** Paul V. Biron and Ashok Malhotra, editors. XML Schema Part 2: Datatypes. <http://www.w3.org/TR/xmlschema-2/> . World Wide Web Consortium, 2000.

1.3 Non-normative References

- [XPATH2]** W3C Recommendation, "XML Path Language (XPath) Version 2.0" . World Wide Web Consortium, 2007. <http://www.w3.org/TR/2007/REC-xpath20-20070123/>
- [NIEM]** National Information Exchange Model, Naming and Design Rules (NDR), Version 1.3. United States government, 2008, <http://niem.gov/library.php#naming> released on 2008-10-31

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 :

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

All element and attributes names are given in lower camel case (lower-case, with embedded concatenated words starting upper-case if any). Type names consist of the element name with the suffix '`_type`' appended.

Where the model specifies an attribute named 'content', usually with a base datatype 'string', the markup provides for this either with a base type of `xsd:string` assigned to an element's type (or a datatype derived from `xsd:string` such as `xsd:normalizedString` or `xsd:token`) or by allowing mixed content for the element's type.

Elements 'testAssertion' and 'testAssertionSet' are global elements and can be top level elements in a TAML markup instance (e.g., be root elements of an XML document). All other TAML elements are local in a markup instance under the TAML schema (i.e., are descendant children of a global element as provided by the schema).

2.2 Conventions Used in the XML Markup and its Usage

The namespace prefix in use for the test assertion markup throughout this document is **taml**, understood to be bound to the namespace: <http://docs.oasis-open.org/ns/tag/taml-201002/>.

Many of the elements in the Test Assertion Markup Language allow extensions both in their attributes and in their children elements. With the exception of any extension inside these three elements: `taml:testAssertionSet/taml:common`, `taml:testAssertion/taml:report` and `taml:testAssertion/taml:normativeSource/taml:comment`, which do not put any restriction on the use of

namespaces in their user-defined child elements, all other additional attributes and elements **shall** be in a namespace other than the taml namespace (as indicated in compact RelaxNG notation by using the subtraction: "- taml:*").

Instances of this markup are intended to be used either "standalone" i.e. in documents that do not contain any other markup foreign to this specification, or "embedded", i.e. as elements inside documents the root element of which belongs to a namespace foreign to this specification. Instances of this markup are XML elements representing either test assertions, or test assertion sets.

The compact Relax NG notation is used for representing the XML definitions.

The XPath notation may be used for representing attributes or elements, relative to 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 name { 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 }*
```

The XML representation of most model elements that are mandatory - i.e. attribute or association of cardinality (1..1) - is optional in the markup. This is because such elements, although mandatory, may be implicitly represented at test assertion set level. When such a mandatory element is not represented in a test assertion instance (taml:testAssertion), this test assertion **shall** be embedded in a test assertion set (see Section 2.4) and this test assertion set **shall** contain a taml:common element that contains the missing test assertion part as a child element.

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

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

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

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

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

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

Example:

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

```
<taml:testAssertion
  id="TA_R6.1"
  lg="http://www.w3.org/TR/xpath20/"
  xmlns:taml="http://docs.oasis-open.org/ns/tag/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:target>
  <taml:predicate>not(@mixed) or @mixed ne 'true'</taml:predicate>
  <taml:prescription level="mandatory"/>
  <taml:report label="fail" message="Rule 6-1 violation"/>
</taml:testAssertion>
```

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

2.3.2 taml:normativeSource

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

Compact Relax NG definition:

```
element taml:normativeSource { normativeSource_def }
normativeSource_def =
  attribute * - taml:* { text }*,
  element taml:comment { comment_def }?,
  element taml:interpretation { interpretation_def }?,
  element taml:refSourceItem { refSourceItem_def }*,
  element taml:textSourceItem { textSourceItem_def }*,
  element taml:derivedSourceItem { refSourceItem_def }*,
  element * - taml:* { anyElement }* & text
```


2.3.2.1 taml:refSourceItem

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

Compact Relax NG definition:

```
element taml:refSourceItem { refSourceItem_def }
refSourceItem_def =
  attribute srcname { 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 attribute `@srcname` represents the name attribute in the model. 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.

2.3.2.2 taml:textSourceItem

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

Compact Relax NG definition:

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

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

2.3.2.3 taml:derivedSourceItem

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

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

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

2.3.2.4 taml:comment

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

Compact Relax NG definition:

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

2.3.2.5 taml:interpretation

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

Compact Relax NG definition:

```
element taml:interpretation { interpretation_def }
interpretation_def =
  attribute * - taml:* { text }*,
  text
```

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
```

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

The target content **may** be an expression in a specialized formal expression language which **should** be identified using the 'lg' attribute. Such an expression or function should identify the set of target instances

to which the test assertion applies. This content may also be a textual representation of the target instance(s) under consideration. The `taml:target` element allows for mixed content.

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
```

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
```

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:

```
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 **shall** be prefixed with a namespace prefix as-

sociated with a namespace declared in the markup. Default namespaces (without a prefix) **shall not** be used.

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:

```
element taml:description { description_def }
description_def =
  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 tname { 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/@tname` attribute corresponds to the name attribute in the model, and **shall** be used in a tag element and have a non-empty value. Some tag names are reserved and have special meaning (listed in the Test Assertions Model [TAM], section 3.2.12). They are:

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

2.3.9 taml:var

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

Compact Relax NG definition:

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

The content of the `taml:var` element is representing the "content" attribute of the corresponding class 'variable' in the model. The `var/@vname` attribute corresponds to the name attribute in the model, and **shall**

be used in a `var` element and have a non-empty value. The `var/@vtype` attribute may be used to indicate the intended type of this `var` element content, e.g. in case it is to be used in some expression. 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 =
  attribute label { xsd:normalizedString }?,
  attribute message { xsd:normalizedString }?,
  attribute when { xsd:normalizedString }?,
  attribute lg { xsd:normalizedString }?,
  attribute * - taml:* { text }*,
  element * { anyElement }* & text

```

The optional `taml:report` element is used to associate a message and a label with each possible evaluation of a target instance, as listed in the Test Assertion Model (Section 3.2.4).

The combination of allowing both mixed content (text can be interspersed with the XML tags) and extra elements from other namespaces ('`xsd:any`') means that the content of this element can be a mixture of text and XML elements.

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. .

The optional `when` attribute **may** be used to state an additional condition that must be satisfied in order for this `report` element to apply, when more than one `report` elements are associated with the same outcome described by the `label` attribute.

The content of optional attribute “message” **shall** describe the general meaning of the assertion evaluation outcome, e.g. provide a standard error message.

The `taml:report` element allows for mixed content.

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 setid { xsd:normalizedString } ?,
    attribute setname { 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 `taml:testAssertionSet/@setname` gives a name to the test assertions set. The attribute `taml:testAssertionSet/@setid` uniquely identifies the test assertions set.

The `testAssertionSet` element **may** be used to group together test assertions either by direct inclusion of the test assertions within the test assertion set, or by references to externally-defined test assertions (using `taml:testAssertionRef`).

A `testAssertionSet` element **may** be used to wrap together all the test assertions in a document. A document containing a set of test assertions **may** have `testAssertionSet` as the top element.

The `common` element **may** be used to group together parts or definitions that are common to all test assertions in the set, e.g. “global” var definitions, tag definitions, parts of a test assertion that are implicitly shared by all. Any individual test assertion in the set can however override the elements in the `common` element. The `common` element may contain additional descriptions and elements not specified here, that help understand the purpose of this set and its relationship with external material.

The `taml:common` element allows for mixed content.

2.4.2 taml:testAssertionRefList

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

Compact Relax NG definition:

```

element taml:testAssertionRefList { testAssertionRefList_def }
testAssertionRefList_def =
    attribute listname { xsd:normalizedString }?,
    attribute sourcedoc { xsd:anyURI }?,
    attribute * - taml:* { text } *,
    element taml:testAssertionRef { testAssertionRef_def } *,
    text

```

A test assertion set in which references are made to existing test assertions defined outside of the test assertion set element (whether in the same document or other documents) shall use the `testAssertionRefList` child element to do so.

The attribute `taml:testAssertionRefList/@listname` gives a name to the reference list. More than one reference lists may be used inside a test assertion set element.

The attribute `taml:testAssertionRefList/@sourcedoc` may be used to specify a URL resolving to a document or a resource that contains the externally-defined test assertions referenced in this list.

The element `taml:testAssertionRefList/taml:testAssertionRef` identifies one externally-defined test assertion, and can be repeated.

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:

```
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` is the name of the referenced test assertion if any, for convenience.

The attribute `taml:testAssertionRef/@sourcedoc` may be used to specify a URL resolving to a document or resource that contains the externally-defined and referenced test assertion. In case this attribute is also used over the parent `taml:testAssertionRefList`, the URL of the reference overrides the URL of the parent.

The value (string) of `taml:testAssertionRef` may be used for describing the referred test assertion, for convenience.

3 XML Schema

The following schema is called here the TAML schema:

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

  <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" pro-
cessContents="skip"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="common_type" mixed="true">
    <xs:sequence>
      <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any" pro-
cessContents="skip"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="description_type" mixed="true">
    <xs:sequence>
      <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any" pro-
cessContents="skip"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="skip"/>
  </xs:complexType>

  <xs:complexType name="interpretation_type">
    <xs:simpleContent>
      <xs:extension base="xs:normalizedString">
        <xs:anyAttribute namespace="##other" processContents="skip"/>
      </xs:extension>
    </xs:simpleContent>
  </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:sequence>
  </xs:complexType>
</xs:schema>
```



```

        <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" minOc-
curs="0"
        maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOc-
curs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="skip"/>
</xs:complexType>

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

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

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

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

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

<xs:complexType name="refSourceItem_type">
    <xs:simpleContent>
        <xs:extension base="xs:normalizedString">
            <xs:attributeGroup ref="resource_attributeGroup"/>
            <xs: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:normalizedString"/>
  <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="srcname" 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="tname" type="xs:normalizedString" use="required"/>
      <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:attribute name="idscheme" type="xs:normalizedString"/>
  <xs:anyAttribute namespace="##other" processContents="skip"/>
</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:sequence>
</xs:complexType>

```

```

    <xs:element name="tag" type="tag_type" minOccurs="0" maxOccurs="unbound-
ded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOc-
curs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:normalizedString" use="required"/>
  <xs:attribute name="lg" type="xs:normalizedString"/>
  <xs:attribute name="name" type="xs:normalizedString"/>
  <xs:attribute name="schemaVersionId" type="xs:normalizedString"/>
  <xs:anyAttribute namespace="##other" processContents="skip"/>
</xs:complexType>

<xs:complexType name="testAssertionRefList_type">
  <xs:sequence>
    <xs:element maxOccurs="unbounded" minOccurs="0" name="testAssertionRef"
type="testAssertionRef_type"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOc-
curs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="sourcedoc" type="xs:anyURI"/>
  <xs:attribute name="name" type="xs:normalizedString"/>
  <xs:anyAttribute namespace="##other" processContents="skip"/>
</xs:complexType>

<xs:complexType name="testAssertionRef_type">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="taid" type="xs:normalizedString"/>
      <xs:attribute name="name" type="xs:normalizedString"/>
      <xs:attribute name="sourcedoc" type="xs:anyURI"/>
      <xs:anyAttribute namespace="##other" processContents="skip"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="testAssertionSet_type">
  <xs:sequence>
    <xs:element minOccurs="0" name="common" type="common_type"/>
    <xs:element name="testAssertionRefList"
type="testAssertionRefList_type" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" name="testAssertion"
type="testAssertion_type"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOc-
curs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="setid" type="xs:normalizedString"/>
  <xs:attribute name="setname" type="xs:normalizedString"/>
  <xs:anyAttribute namespace="##other" 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="txtname" type="xs:normalizedString"/>
      <xs:anyAttribute namespace="##other" processContents="skip"/>
    </xs:extension>
  </xs:simpleContent>

```

```

</xs:complexType>

<xs:complexType name="var_type" mixed="true">
  <xs:sequence>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="lg" type="xs:normalizedString"/>
  <xs:attribute name="vname" type="xs:normalizedString" use="required"/>
  <xs:attribute name="vtype" type="xs:normalizedString"/>
  <xs:anyAttribute namespace="##other" processContents="skip"/>
</xs:complexType>

</xs:schema>

```

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).
- Uses the markup in compliance with the general semantics of a test assertion and its parts as described in the Test Assertions Model [\[TAM\]](#) specification.

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

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

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).
- 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).

Appendix A. Acknowledgments

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

Participants:

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

Appendix B. Revision History

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