Test Assertions Part 1 - Test Assertions Model Version 1.0

Committee Specification 02

30 November 2011

Specification URIs

This version:
http://docs.oasis-open.org/tag/model/v1.0/cs02/testassertionsmodel-1.0-cs02.pdf (Authoritative)
http://docs.oasis-open.org/tag/model/v1.0/cs02/testassertionsmodel-1.0-cs02.html
http://docs.oasis-open.org/tag/model/v1.0/cs02/testassertionsmodel-1.0-cs02.odt

Previous version:
http://docs.oasis-open.org/tag/model/v1.0/cd02/testassertionsmodel-1.0-cd-02.pdf (Authoritative)
http://docs.oasis-open.org/tag/model/v1.0/cd02/testassertionsmodel-1.0-cd-02.html
http://docs.oasis-open.org/tag/model/v1.0/cd02/testassertionsmodel-1.0-cd-02.odt

Latest version:
http://docs.oasis-open.org/tag/model/v1.0/testassertionsmodel-1.0.pdf (Authoritative)
http://docs.oasis-open.org/tag/model/v1.0/testassertionsmodel-1.0.html
http://docs.oasis-open.org/tag/model/v1.0/testassertionsmodel-1.0.odt

Technical Committee:
OASIS Test Assertions Guidelines (TAG) TC

Chairs:
Jacques Durand (jdurand@us.fujitsu.com), Fujitsu Limited
Patrick Curran (Patrick.Curran@oracle.com), Oracle

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

Related work:
This specification is related to:
• Test Assertions Guidelines Version 1.0. Latest version.
  http://docs.oasis-open.org/tag/guidelines/v1.0/testassertionsguidelines.html
• Test Assertions Part 2 - Test Assertion Markup Language Version 1.0. Latest version.
  http://docs.oasis-open.org/tag/taml/v1.0/testassertionmarkuplanguage-1.0.html

Abstract:
The specification defines a model for Test Assertions that are associated with a specification, and
defines their use and semantics.

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:

[Test-Assertions-Model]

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" and "OASIS Test Assertions Model" 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
      1.1.1  Data Model Formal Definition Terminology............................................................ 5
      1.1.2  Domain terminology................................................................................................. 5
   1.2  Normative References....................................................................................................... 6
   1.3  Non-Normative References............................................................................................... 6

2  Role and Rationale .................................................................................................................. 7
   2.1  The Role of a Test Assertion............................................................................................. 7
   2.2  Benefits of Test Assertions............................................................................................... 7

3  Test Assertion........................................................................................................................ 9
   3.1  Test Assertion Overview and Terminology....................................................................... 9
      3.1.1  Core Test Assertion Parts........................................................................................... 9
      3.1.2  Optional Test Assertion Parts.................................................................................... 9
      3.1.3  Implicit Test Assertion Parts...................................................................................... 10
      3.1.4  Informal Notation....................................................................................................... 10
   3.2  Test Assertion Model......................................................................................................... 11
      3.2.1  Convention Used for Formally Defining the Model.................................................... 11
      3.2.2  Mapping Test Assertion Terminology to the Formal Notation.................................. 11
      3.2.3  General Structure of a Test Assertion .................................................................... 12
      3.2.4  testAssertion............................................................................................................. 12
      3.2.5  id............................................................................................................................. 14
      3.2.6  normativeSource....................................................................................................... 15
      3.2.7  target....................................................................................................................... 17
      3.2.8  prerequisite.............................................................................................................. 18
      3.2.9  predicate................................................................................................................ 19
      3.2.10 prescription.............................................................................................................. 19
      3.2.11 description.............................................................................................................. 20
      3.2.12 tag........................................................................................................................ 21
      3.2.13 variable................................................................................................................... 22

4  Conformance.......................................................................................................................... 23
   4.1  Conformance Clause for Test Assertion Formal Representation...................................... 23
   4.2  Conformance Clause for Test Assertion Instances.......................................................... 23

Appendix A.  Deriving a Test Case from a Test Assertion.......................................................... 24

Appendix B.  Acknowledgments.................................................................................................. 25

Appendix C.  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.

1.1.1 Data Model Formal Definition Terminology
The means of formally defining the model in this specification involves the use of terms “class”, “attribute”, “data-type” and “association”. These are terms familiar in an object oriented paradigm but should not be strictly interpreted as object oriented terms. The terms are used as a means of formally defining the data structures in the model and do not specify or imply how that data is to be accessed or used. The use of the object oriented terminology should not be taken to mean that the implementation is to be object oriented.

Class
The term “class” is used when the structure so modeled is a complex grouping of more than one entity (either “attributes” or “associations” or both).

Datatype
The term “datatype” is primarily used of a simple, primitive type such as a string or integer.

Attribute
The term “attribute” is used to specify an entity that is an instance of a primitive or simple datatype such as a string or an integer.

Association
The term “association” is used of an entity which is an instance of a class (i.e. its structure is defined by a class) and which appears as an element inside another class.

1.1.2 Domain terminology
This section provides definitions of terms that are related but not central to the notion of test assertion. These definitions represent a common understanding among practitioners but do not pretend to be here authoritative.

Conformance
The fulfillment of specified requirements by a product, document, process, or service.

Conformance Clause
A statement in a specification that lists all the criteria that must be satisfied by an implementation (data artifact or processor) in order to conform to the specification. The clause refers to a set of normative statements and other parts of the specification for details.

Implementation
A product, document, process, or service that is the realization of a specification or part of a specification.

Normative Statement, Normative Requirement
A statement made in the body of a specification that defines prescriptive requirements on a conformance target.

Test Case
A set of a test tools, software or files (data, programs, scripts, or instructions for manual operations) that verifies the adherence of a test assertion target to one or more normative statements in the specification. Typically a test case is derived from one or more test assertions. Appendix A proposes a more precise definition of the meaning of deriving a test case from a test assertion. Each test case typically includes: (1) a description of the test purpose (what is being tested - the conditions / requirements / capabilities which are to be addressed by a particular test), (2) the pass/fail criteria, (3) traceability information to the verified normative statements, either as a reference to a test assertion, or as a direct reference to the normative statement. They are normally grouped in a test suite.

Test Metadata
Metadata that is included in test cases to facilitate automation and other processing.
1.2 **Normative References**

http://www.iso.org/iso/standards_development/processes_and_procedures/iso_iec_directives_and_iso_supplement.htm

1.3 **Non-Normative References**

[CONFCLAUSE] OASIS, "Guidelines to Writing Conformance Clauses ", September 2007

[CONF1] OASIS, "Conformance requirements for Specifications", March 2002,

[CONF2] OASIS, "Conformance testing and Certification Framework", Conformance TC, June 2001,


[VAR] W3C, "Variability in Specifications", WG note,2005,
http://www.w3.org/TR/2005/NOTE-spec-variability-20050831/
2 Role and Rationale

2.1 The Role of a Test Assertion
A test assertion is a testable or measurable expression for evaluating the adherence of an implementation (or part of it) to one or more normative statements in a specification. It describes the expected output or behavior for the test assertion target within specific operation conditions, in a way that can be measured or tested.

A Test Assertion should not be confused with a Conformance Clause, nor with a Test Case. The specification will often have one or more conformance clauses CONFCLAUSE [[CONF1]][[CONF2]] which define various ways to conform to a specification [[VAR]]. A set of test assertions may be associated with a conformance clause in order to define more precisely what conformance entails for a candidate implementation. Test assertions lie between the specification and any suite of tests to be conducted to determine conformance. Such a test suite is typically comprised of a set of test cases. These test cases are derived from test assertions that address the normative statements of the specification.

2.2 Benefits of Test Assertions

Improving the Specification
When defined at an early stage, test assertions may help provide a tighter specification. Any ambiguities, gaps, contradictions and statements which require excessive or impractical resources for testing can be noted as they become apparent during test assertion creation. If there is still an opportunity to correct or improve the specification, these notes can be the basis of comments to the specification authors. If not developed by the specification authors, test assertions should be reviewed and approved by them which will improve both the quality and time-to-deployment of the specification. Therefore, best results are achieved when assertions are developed in parallel with the specification.

Facilitating Testing
Test assertions provide a starting point for writing a conformance test suite or an interoperability test suite for a specification that can be used during implementation. They simplify the distribution of the test development effort between different organizations while maintaining consistent test quality. By tying test output to specification statements, test assertions improve confidence in the resulting test and provide a basis for coverage analysis (estimating the extent to which the specification is tested).

**Aligning Implementations**

Test assertions provide explicit guidance for implementers of a specification, by stating more concretely and practically the conditions to fulfill in order to conform. Unlike test suites which can only be exercised once the implementation work is done, test assertions are usable early on during the implementation work.
3 Test Assertion

3.1 Test Assertion Overview and Terminology

3.1.1 Core Test Assertion Parts
The following are defined as the "core" parts of a test assertion:

**Identifier**
A unique identifier for the test assertion. It is recommended that the identifier be made universally unique.¹

**Normative Sources**
These refer to the precise specification requirements or normative statements that the test assertion addresses.

**Target**
A test assertion target is the implementation or part of an implementation that is the main object of the test assertion, and of its Normative Sources. It categorizes an implementation or a part of an implementation of the referred specification.

**Predicate**
A predicate asserts, in the form of an expression, the feature (a behavior or a property) described in the specification statement(s) referred by the Normative Sources. If the predicate is an expression which evaluates to “true” over a Target instance, this means that the test assertion target exhibits this feature. “False” means the target does not exhibit this feature.

3.1.2 Optional Test Assertion Parts
The following are defined as the "optional" parts of a test assertion:

**Prescription Level**
The prescription level is a keyword that indicates how imperative it is that the Normative Statement referred to in the Normative Source, be met. The test assertion defines a normative statement which may be mandatory (MUST / REQUIRED / SHALL), permitted (MAY / CAN) or preferred (SHOULD / RECOMMENDED). This property can be termed the test assertion’s prescription level.

NOTE: in the case of the normative source including keywords 'MUST NOT' or 'shall not' the prescription level 'mandatory' is used and the 'NOT' included in the predicate. Similarly, a statement with NOT RECOMMENDED or SHOULD NOT belongs to the 'preferred' level. There are differences between various conventions of normative language [ISO/IEC Directives] [RFC 2119] and the above terms may be extended with more specialized terms for a particular convention and its distinct shades of meaning.

**Prerequisite**
A test assertion Prerequisite is a logical expression (similar to a Predicate) which further qualifies the Target for undergoing the core test (expressed by the Predicate) that addresses the Normative Statement. It may include references to the outcome of other test assertions. If the Prerequisite evaluates to "false" then the Target instance is not qualified for evaluation by the Predicate.

**Tag**
Tags represent properties or 'keywords' that qualify a test assertion. Tags may be given values. Tags provide a means to categorize the test assertions. They enable the grouping of the test assertions, for example based on the type of test they assume or based on some property of their Target.

**Variable**
Variables are convenient for storing values, reuse and shared use, within or across test assertions. Another use of a variable is as parameter or attribute employed by the writer of a test assertion to refer to a value that is not known at the time the test assertion is written, but which will be determined at some later stage, possibly as late as the middle of running a set of tests.

**Description**
An informal definition of the role of the test assertion, with some optional details on some of its parts. This description shall not alter the general meaning of the test assertion and its parts as described in this

¹ One way to do this is to designate a universally unique name for a set of test assertions and to include this name along with the identifier when referencing the test assertion from outside of this set.
model. This description may be used to annotate the test assertion with any information useful to its understanding. It does not need to be an exhaustive description of it.

3.1.3 Implicit Test Assertion Parts

In an actual test assertion definition, the previously mentioned parts are often explicitly represented as elements of the test assertion.

A concrete representation of a test assertion may omit elements (core or optional) provided they are implicit, meaning that the context in which the test assertion is defined, allows for unambiguous determination of the non-explicit element, e.g. via some rule or inference. A common case of implicit test assertion components is the implicit target: When several test assertions relate to the same target, the latter may be described just once as part of the context where the test assertions are defined, so that it does not need to be repeated. This calls for further structural components than those described so far. The more complex structure may include a test assertion set whose model caters for sharing of test assertion parts among a group of test assertions.

3.1.4 Informal Notation

The following notation will be used for a plain English representation of a test assertion. In bold, are the test assertion part names as defined in the above terminology section:

<table>
<thead>
<tr>
<th>Test Assertion Part</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA id:</td>
<td>(here state the Identifier of the test assertion)</td>
</tr>
<tr>
<td>Normative Source:</td>
<td>(here state the Normative Source reference or copy)</td>
</tr>
<tr>
<td>Target:</td>
<td>(here state the Target identifier)</td>
</tr>
<tr>
<td>Prerequisite:</td>
<td>(here an assertion stating the Prerequisite with possible reference to the Target. Notational convention: the reference to the Target is within square brackets.)</td>
</tr>
<tr>
<td>Predicate:</td>
<td>(here an assertion stating the Predicate and referring to the Target. Notational convention: the reference to the Target is within square brackets.)</td>
</tr>
<tr>
<td>Prescription Level:</td>
<td>(here state the Prescription Level of the test assertion, which is a keyword among {mandatory, preferred, permitted})</td>
</tr>
<tr>
<td>Tag:</td>
<td>(here a name / value pair expressing a particular Tag. This test assertion part can be repeated. Notational convention: use the operator '=' between name and value.)</td>
</tr>
<tr>
<td>Variable:</td>
<td>(here an identifier, along with its definition and/or value, representing a Variable that is reused in some other part of the test assertion. This test assertion part can be repeated. Notational convention: use the operator '=' between name and value, and put the definition in parenthesis just after the name, if any.)</td>
</tr>
<tr>
<td>Description:</td>
<td>(here state the Description of the test assertion)</td>
</tr>
</tbody>
</table>

This informal notation will be used for describing examples of test assertions.

Example of informal Test Assertion:

<table>
<thead>
<tr>
<th>Test Assertion Part</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA id:</td>
<td>gizmo-TA300</td>
</tr>
<tr>
<td>Normative Source:</td>
<td>specification requirement 317</td>
</tr>
<tr>
<td>Target:</td>
<td>electrical-gizmo</td>
</tr>
<tr>
<td>Prerequisite:</td>
<td>[The gizmo] has a low-battery indicator.</td>
</tr>
<tr>
<td>Predicate:</td>
<td>The low-battery indicator of [the gizmo] is a red LED that is flashing below CRITICAL-CHARGE battery voltage.</td>
</tr>
<tr>
<td>Prescription Level:</td>
<td>mandatory</td>
</tr>
<tr>
<td>Tag:</td>
<td>conformanceclass = &quot;international&quot;</td>
</tr>
<tr>
<td>Variable:</td>
<td>CRITICAL-CHARGE (the critical voltage limit in a battery).</td>
</tr>
</tbody>
</table>
3.2 Test Assertion Model

3.2.1 Convention Used for Formally Defining the Model
The means of formally defining the model in this specification involves the use of terms “class”, “attribute”, “data-type” and “association” as defined in section 1.1.1.

Example Formal Definition:
```
myclass {
    content : string (0..1)
    id : string (1..1)
    Child : child (1..*)
    Sibling : sibling (0..*)
}
```

With the exception of the example above, all of the textual representations of model constructs in this specification are normative and authoritative. However, some classes in this specification may be extended either by adding further attributes or by adding further associations or both. This is indicated in the prose immediately following the representation of the class.

The class name, here called 'myclass', is shown before the opening curly bracket. The attributes combine the name of the attribute in lower camel case separated by a colon from the name of the data-type on which the type of the attribute is based. The associations combine the name of the association in upper camel case separated by a colon from the name of the class which is associated and which represents the type of the association.

The cardinality is specified using the notation “(x..y)” where “x” represents the lower bound and “y” the upper bound of the cardinality. The symbol “*” represents a limitless upper bound. There are the following cardinalities used in the model:

- (0..1) specifies an optional, singular entity (lower bound 0, upper bound 1)
- (0..*) specifies an optional, multiple entity (lower bound 0, upper bound unlimited)
- (1..1) specifies a mandatory, singular entity (lower bound 1, upper bound 1)
- (1..*) specifies a mandatory, multiple entity (lower bound 1, upper unlimited)
- (x..y) specifies an entity lower bound x, upper y where x and y are positive integers, for example (1..2)

In the example representation above the class “myclass” has a mandatory attribute, shown with “(1..1)” to signify that it is mandatory, called “id” whose content is type “string”. Another attribute named “content” is shown to be optional by the notation (0..1). The “myclass” class has associations to other classes called “child” and “sibling”. These are similar to attributes whose types are complex and represented in this model as classes. The (0..*) notation signifies that the entity named “sibling” has multiple cardinality and is optional. The (1..*) after the association called “child” signifies that this association is mandatory and multiple.

Any graphic images such as class diagrams included in this specification are non-normative. It is the text which shall be taken as normative. Any diagrams are to be interpreted loosely as illustrative material and in the case of any discrepancy with the text it is the text which is to be taken as authoritative.

3.2.2 Mapping Test Assertion Terminology to the Formal Notation

Table 1. Mapping Section 3.1 test assertion parts (as defined in the previous terminology section) to the formal Test Assertions Model

<table>
<thead>
<tr>
<th>Test Assertion Parts</th>
<th>Corresponding Entities in Test Assertions Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Assertion</td>
<td>Class: testAssertion</td>
</tr>
<tr>
<td>Core Parts</td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>attribute: testAssertion.id</td>
</tr>
</tbody>
</table>
### 3.2.3 General Structure of a Test Assertion

A test assertion shall include, implicitly or explicitly the following parts:

- Identifier
- Normative Source
- Target
- Predicate

In addition, a test assertion may optionally include the following parts:

- Prescription Level
- Prerequisite
- Tag (possibly many)
- Variable (possibly many)
- Description

### 3.2.4 testAssertion

An instance of testAssertion is a testable or measurable expression for evaluating the adherence of an implementation (or part of it) to one or more normative statements in a specification.

Formal Definition of 'testAssertion':

```plaintext
testAssertion {
    id : string (1..1)
    NormativeSource : normativeSource (1..1)
    Target : target (1..1)
    Prerequisite : prerequisite (0..1)
    Predicate : predicate (1..1)
    Prescription : prescription (0..1)
    Description : description (0..1)
    Tag : tag (0..*)
    Variable : variable (0..*)
}
```
Semantics:

- The `<id>` attribute (corresponding to the Identifier terminology definition) is uniquely identifying the test assertion.

- The `<NormativeSource>` association (corresponding to the Normative Source terminology definition) is identifying the normative statement in the specification that describes the feature or behavior that needs to be verified over a `<Target>` instance.

- The `<Target>` association (corresponding to the Target terminology definition) is identifying (or categorizing) the specification implementation(s) or parts of, subject to testing.

- The `<Prerequisite>` association (corresponding to the Prerequisite terminology definition) expresses a pre-condition to be satisfied by the `<Target>` in order to qualify for the test expressed by the `<Predicate>`. It is a boolean expression: if evaluates to "true", the `<Predicate>` can be evaluated over the `<Target>`. If evaluates to "false", the `<Target>` is not qualified for this test assertion.

- The `<Predicate>` association (corresponding to the Predicate terminology definition) expresses the feature or behavior expected from the `<Target>` as stated in `<NormativeSource>`. It is a boolean expression: if evaluates to "true", the `<Target>` instance exhibits the expected feature. If "false", the `<Target>` does not.

- The `<Prescription>` association (corresponding to the Prescription Level terminology definition) expresses how imperative is the statement referred by `<NormativeSource>`: usually one level among {"permitted", "preferred", "mandatory"} (corresponds to optional/recommended/required)

- The `<Description>` association (corresponding to the Description terminology definition) gives an informal definition of this particular test assertion.

- The `<Tag>` association (s) (corresponding to the Tag terminology definition) add some annotation mechanism in the form of (name, value) pair(s), or just a (name) property.

- The `<Variable>` association (s) (corresponding to the Variable terminology definition) provide some way to parameterize the expressions used in other elements of the test assertion, or to abstract some of its values. It is in the form of a (name, value) pair or just a (name). An additional definition statement may be added to the name.

Other attributes and associations may be added to the `testAssertion` class.
The `<NormativeSource>`, `<target>` and `<Predicate>` elements, although mandatory, may be implicit and also may be declared in a test assertion set (specified later). An instance of the `testAssertion` class may have any
of its parts defined implicitly, i.e. their actual representation can be inferred, either from a container structure like a "test assertion set" or from other rules.
The Prerequisite and Predicate elements are of same nature. They are logical statement evaluating to true or false, which may in turn be composed of sub-expressions or sub-statements. These sub-expressions may be captured by Variables.

The overall semantics of a Test Assertion with regard to its Target, may be summarized as follows:
- The "Target" is said to be not qualified for the Test Assertion if the Prerequisite (if any) evaluates to "false" over the Target.
- The "Target" is said to fulfill the Normative Statement addressed by the Test Assertion if the Prerequisite (if any) evaluates to "true" over the Target, and the Predicate evaluates to "true".
- The "Target" is said to not fulfill the Normative Statement addressed by the Test Assertion if the Prerequisite (if any) evaluates to "true" over a Target, and the Predicate evaluates to "false".

Test Assertion (Non-Normative UML-Style Class Diagram)

### 3.2.5 id
This attribute is the identifier of the test assertion. Its string value should be universally unique.
### 3.2.6 normativeSource

An instance of 'normativeSource' identifies the normative statement in the specification that describes the feature or behavior that needs to be verified over a 'target' instance.

**Formal Definition of 'normativeSource':**

```plaintext
normativeSource {
    content : string (0..1)
    Comment : comment (0..1)
    Interpretation : interpretation (0..1)
    RefSourceItem : refSourceItem (0..*)
    TextSourceItem : textSourceItem (0..*)
    DerivedSourceItem : derivedSourceItem (0..*)
}
```

**Semantics:**

- The `<content>` attribute allows for quoting the entire normative source inside the test assertion (e.g. a copy of the original normative statement as it appears in a specification), when it is a single statement.
- The `<Comment>` association allows to add comments about the normative source.
- The `<Interpretation>` association may be used to add an alternative description in prose of any kind to a normative source e.g. to clarify its meaning or facilitate human understanding. It may provide further information clarifying how the predicate (or prerequisite) relates to the normative source.
- The `<RefSourceItem>` association references the original normative source statement, when externally defined.
- The `<TextSourceItem>` association quotes verbatim the source item.
- The `<DerivedSourceItem>` association derives a form of words equivalent in meaning to the source item. This is useful when the source consists of tables, diagrams, graphs or text spread over several parts of the specification.

Other attributes may be added to the normativeSource class.

The normative source of a test assertion may be provided as a reference using the `refSourceItem` class.

**Formal Definition of 'refSourceItem':**

```plaintext
refSourceItem {
    name : string (0..1)
    uri : string (0..1)
    documentId : string (0..1)
    versionId : string (0..1)
    revisionId : string (0..1)
    resourceProvenanceId : string (0..1)
}
```
Semantics:

- The `<name>` attribute is the name of the referred document containing the normative statement.
- The `<uri>` attribute contains a URI that locates the resource.
- The `<documentId>` attribute identifies the referred document containing the normative statement.
- The `<versionId>` attribute identifies the version of the referred document.
- The `<revisionId>` attribute identifies the revision of the referred document.
- The `<resourceProvenanceId>` attribute contains additional source information associated with the referred document (such as authorship identifiers to certify its authenticity).

Other attributes may be added to the `refSourceItem` class.

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 association with a class named `textSourceItem` which allows a direct, verbatim quote of the specification text.

Formal Definition of `textSourceItem`:

```
textSourceItem {
  content : string (0..1)
  name : string (0..1)
}
```

Semantics:

- The `<content>` attribute is quoting the normative source item.
- The `<name>` attribute is an informal qualifier of the statement.

Other attributes may be added to the `textSourceItem` class.

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 association to a class named `derivedSourceItem`. This is particularly useful when the source consists of tables, diagrams, graphs or text spread over several parts of the specification.

Formal Definition of `derivedSourceItem`:

```
derivedSourceItem {
  content : string (0..1)
```

name : string (0..1)
uri : string (0..1)
documentId : string (0..1)
versionId : string (0..1)
revisionId : string (0..1)
dateString : string (0..1)
resourceProvenanceId : string (0..1)

}

Semantics:

- The <content> attribute is expressing the normative statement as it has been interpreted from the (possibly non-textual) referred source material.
- other elements have same semantics as in refSourceItem class.

Other attributes may be added to the derivedSourceItem class.

Formal Definition of 'comment':

calendar {
    content : string (0..1)
}

} Other attributes may be added to the calendar class.
The calendar class 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.

Formal Definition of 'interpretation':

interpretation {
    content : string (0..1)
}

} Other attributes may be added to the interpretation class.
The interpretation class 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. It provides further information about how the predicate (or prerequisite) relates to the normative source.

3.2.7 target

An instance of 'target' identifies (or categorizes) the specification implementation(s) or parts of, subject to testing.

Formal Definition of 'target':

target {
    content : string (0..1)
type : string (0..1)
idScheme : string (0..1)
language : string (0..1)
}
Semantics:

- The `<content>` attribute is identifying the target or the set of targets subject to the test assertion. A target can either be a specific item or a category of items.

- The `<type>` attribute specifies the target category. Its values may belong to a controlled vocabulary, ontology or other classification or taxonomy system.

- The `<idScheme>` attribute specifies the identity scheme used for identifying target instances, e.g. in test reports. It allows for generating the identifier of a target instance.

- The `<language>` attribute identifies the expression language used in the content attributes.

Other attributes may be added to the target class.

Where the scheme for listing or categorizing target types is defined in a document, the identifier, URL or URI for this document may be associated with the target using an additional attribute named 'schemeRef'. A target 'schemeRef' attribute or, for a set of test assertions, a shared target 'schemeRef' attribute may be used in cases where the target type scheme is defined using an expression or prose definition within the test assertion or set of test assertions.

The target content is a string. This may be an expression in a specialized formal expression language which may be specified using the 'language' attribute or using a complete conformance profile for that particular use of the markup.

### 3.2.8 prerequisite

An instance of 'prerequisite' expresses a pre-condition to be satisfied by the related 'target' instance in order to qualify for the test expressed by the 'predicate'. It is a boolean expression: if evaluates to "true", the 'predicate' can be evaluated over the 'target'. If evaluates to "false", the 'target' is not qualified for this test assertion.

Formal Definition of 'prerequisite':

```
prerequisite {
  content : string (1..1)
  language : string (0..1)
}
```

Semantics:

- The `<content>` attribute is stating the condition that must be met by the target and/or some collateral artifact, in order for the target to qualify for this test assertion.

- The `<language>` attribute identifies the expression language used for stating the prerequisite condition (content).
Other attributes may be added to the prerequisite class. The prerequisite may be expressed using a specialized formal expression language which should be specified using the 'language' attribute.

The content of the prerequisite class shall contain an expression which evaluates to true or false.

### 3.2.9 predicate
An instance of 'predicate' expresses the feature or behavior expected from the 'target' as stated by the 'normative-Source'. It is a boolean expression: if it evaluates to "true", the related 'target' instance exhibits the expected feature. If "false", the 'target' instance does not.

Formal Definition of 'predicate':

```plaintext
predicate {
    content : string (1..1)
    language : string (0..1)
}
```

**Semantics:**

- The `<content>` attribute is stating the condition that must be met by the target and possibly some additional collateral artifact, in order for the target to fulfill the normative statement or its interpretation addressed by this test assertions.

- The `<language>` attribute identifies the expression language used for stating the predicate condition (content), if applicable.

Other attributes may be added to the predicate class. The predicate may be expressed using a specialized formal expression language which should be specified using the 'language' attribute.

A test assertion predicate shall be worded as an assertion, not as a requirement. Normative keywords from [ISO/IEC Directives] [RFC 2119] (e.g. 'MUST' or 'shall' keyword) shall be absent from the predicate but reflected in the prescription level. The predicate has a clear Boolean value: Either the statement is true, or it is false for a particular target.

### 3.2.10 prescription
An instance of 'prescription' expresses how imperative is the statement referred by 'normativeSource' when applying to a 'target'. It is usually one level among {"permitted", "preferred", "mandatory"}.

Formal Definition of 'prescription':

```plaintext
prescription {
    content : string (0..1)
}
```
level : string (0..1) (allowed values include: mandatory|preferred|permitted)
}

**Semantics:**

- The `<content>` attribute is stating prescription information or annotation associated with the normative statement or its interpretation addressed by this test assertions.

- The `<level>` attribute identifies formally the prescription level, typically using a predefined keyword. The content attribute may add further information about this prescription level.

Other attributes may be added to the prerequisite class.
The possible values for the attribute 'level' of the class prescription shall contain the set of predefined values of mandatory, preferred and permitted. Its values may be extended beyond this minimal set.
The prescription values correspond to the terms used in a specification to denote conformance requirements, or to a more nuanced expression:

- [RFC 2119] terms conveying mandatory nature of a statement such as 'MUST' and 'MUST NOT' and in Annex H of [ISO/IEC Directives] terms 'shall', etc shall correspond to the prescription level value 'mandatory'.
- RFC2119 terms conveying optionality with preference such as 'SHOULD' and 'SHOULD NOT', 'RECOMMENDED', etc and ISO/IEC Directive terms 'should', etc shall correspond to the prescription level value 'preferred'.
- RFC2119 terms conveying optionality without preference 'MAY' and ISO/IEC Directive terms 'may', etc shall correspond to the prescription level value 'permitted'.

The RFC2119 terms for preference do not permit non-conformance without a reason and usually the same 'preferred' prescription level is acceptable but in some cases implementers may wish to make a distinction by making use of the extension facility and specify further enumeration values.
The prescription shall not affect the outcome semantics of the test assertion but may determine how this outcome is to be used, e.g. how the outcome affects conformance or otherwise of the implementation to a conformance profile or to the conformance clause of the specification.
Besides the use of the 'level' attribute, the content (string) may be used to express further information regarding the prescription level using prose or as a logical expression.

### 3.2.11 description

An instance of 'description' gives an informal description of a test assertion.

**Formal Definition of 'description':**

description {
    content : string (0..1)
}

**Semantics:**

- The `<content>` attribute is stating a general plain text description of this test assertion or of its intent.
Other attributes may be added to the description class. 

Notes: 
The description class may be used to add a description in prose of any kind to a test assertion or set of test assertions. This is useful when a test assertion is otherwise expressed purely in a specialized, formal, logical language which might not be intended for legibility to human readers;

### 3.2.12 tag

An instance of 'tag' provides some annotation mechanism in the form of (name, value) pair, or just a (name) property. They may help the grouping or categorizing of test assertions - e.g. all test assertions related to a particular conformance profile "CP-1" may use a tag "conformance_profile" and be tagged with: conformance_profile = "CP-1"

Formal Definition of 'tag':

tag {
  content : string (0..1)
  name : string (1..1)
}

**Semantics:**

- The <content> attribute is the value given to this tag.
- The <name> attribute is the name of the tag.

Other attributes may be added to the tag class.

Notes: 
A special example of tag is to indicate to which versions of a specification the test assertion or set of test assertions applies. 
Another example of tag is to specify that a test assertion or set of test assertions exists to define a particular normative property or a conformance level. The tag class may be used to attach such data to a test assertion or test assertion set.

**Reserved Tag Names:**

DefinesNormativeProperty and NormativeProperty

Often, a specification uses normative statements in order to define properties of a target. For example, a “gizmo” specification may define what is an “electrical gizmo” and what is a “mechanical gizmo”, where “electrical” and “mechanical” are possible properties of a gizmo target. Test assertions associated with the verification of a property are called property test assertions.

A test assertion may be tagged with DefinesNormativeProperty to show that it is a property test assertion, and may be tagged with NormativeProperty to define which target property(ies) it is associated with.

**VersionAdd** and **VersionDrop**

VersionAdd: the lowest numerical version to which the test assertion applies. 
VersionDrop: the lowest numerical version number (after VersionAdd if present) 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.

### 3.2.13 variable

An instance of 'variable' allows for abstracting and naming some value used by one or more parts of a test assertion, or across several test assertions of a set. The value of a variable may be determined at a later time, allowing the test assertion to be parameterized.

**Formal Definition of 'variable':**

```plaintext
variable {
    content : string (0..1)
    name : string (1..1)
    language : string (0..1)
}
```

**Semantics:**

- The `<content>` attribute is the value given to this variable.
- The `<name>` attribute is the name of the variable.
- The `<language>` attribute identifies the expression language used by the value, in case it is an expression.

Other attributes may be added to the `variable` class.

**Notes:**

The variable value may be an expression, the evaluation of which may vary depending on the target instance. It may also be a sub-expression of the predicate or of the prerequisite. The notation used for these variables in the content of predicates, prerequisite, etc. is left to implementations of this model.

As for a 'tag', an instance of 'variable' may state a (name, value) pair or just a (name).
4 Conformance

Test assertion artifacts or implementations subject to conformance to this model are of two kinds:
(1) Formal Representations: Languages or notations that represent the test assertion model described in
Section 3 (for example, an XML mark-up language.)
(2) Test Assertion Instances: Actual instances of test assertions, that follow the modeling principles and
semantics described in Section 3. These may or may not use a formal representation.
To each one of the above implementation classes is associated a conformance clause below.

4.1 Conformance Clause for Test Assertion Formal Representation

In order to conform to this model, formal representations or implementations of class 1:
(a) **shall** represent all test assertion parts (core and optional) defined in Section 3 (3), in accordance with
the normative statements and semantics of this section.
(b) **shall** use names for these parts that are identical or can be unambiguously mapped to the definitions
used in Section 3.

In (a) an implementation **may** implement a datatype in the model with a more restricted datatype. (For ex-
ample an attribute specified with an `datatype` “string” may be implemented as a URI.)
Classes in this specification **may** be extended either by adding further attributes or by adding further as-
sociations or both.

4.2 Conformance Clause for Test Assertion Instances

In order to conform to this model, test assertion instances or implementations of class 2:
(a) **shall** be represented using a notation or language, formal or informal, that maps unambiguously to
the test assertion model defined in Section 3(9)
(b) **shall** include the test assertion parts mandated by the test assertion model (Core TA parts) as defined
in Section 3.1 (9), in accordance with the normative statements and semantics of this section.
(c) **shall** define values for its parts in a way that is consistent with the test assertion semantics stated in
the test assertion model, in Section 3.2 (11).
(d) The semantics of the test assertion over its targets **shall** conform to the semantics of the test assertion
specified in Section 3.2.4 (12)
Test assertion instances **may** be extended with additional parts provided these do not semantically over-
ride or invalidate parts defined in this specification.

Mandatory statements are designated by the keyword ‘**shall**' and ‘**shall not**' in bold type, as described in Annex H of
[ISO/IEC Directives].
Appendix A. Deriving a Test Case from a Test Assertion

Although a test assertion is a testable or measurable expression for evaluating the adherence of an implementation – or a part of it – to some normative statements, the actual verification of the implementation is carried out by the test case(s) derived from this test assertion.

Because of practical constraints, it is often the case that a test case derived from a test assertion only verifies a subset of the implementations (target instances) that fall under this test assertion, or verifies these implementation only under specific conditions, or yet is only able to verify some type of outcome – either success or failure. For this reason, it may take several test cases to “cover” a single test assertion – and sometimes imperfectly. A test case is therefore only an indicator of fulfillment or non-fulfillment for target instances. This means that the fulfillment indication provided by a test case cannot generally be considered as a formal proof or fulfillment (or non-fulfillment) of the normative statement by a target instance.

In relation with the test assertion semantics stated in 3.2.4 the following is a set of criteria defining more precisely under which conditions a Test Case can be said to derive from a Test Assertion:

1. When a Target instance is not qualified for a Test Assertion, a Test Case derived from this Test Assertion does not indicate whether the Target instance fulfills or not the Normative Statement addressed by the Test Assertion.
2. When a Target instance is qualified for a Test Assertion and satisfies the Test Assertion Predicate, a Test Case derived from this Test Assertion either indicates that the Target instance fulfills the Normative Statement addressed by the test assertion, or does not indicate anything.
3. When a Target instance is qualified for a Test Assertion and does not satisfy the Test Assertion Predicate, a Test Case derived from this Test Assertion either indicates that the Target instance does not fulfill the Normative Statement addressed by the test assertion, or does not indicate anything.

A test case is said to be derived from a test assertion if it can process [a subset of] instances of the test assertion Target, and if it can indicate either fulfillment or non-fulfillment for at least a non-empty subset of these target instances, in consistency with the semantics of this test assertion.
Appendix B. Acknowledgments

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

Participants:

- Dennis Hamilton, Individual
- Dmitry Kostvarov, 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, KIEC
## Appendix C. 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>12/15/09</td>
<td>Stephen Green</td>
<td>CD 1 candidate</td>
</tr>
<tr>
<td>CD 2</td>
<td>08/10/10</td>
<td>Jacques Durand</td>
<td>CD 2 draft for PR</td>
</tr>
<tr>
<td>CS1</td>
<td>11/30/10</td>
<td>Jacques Durand</td>
<td>CS1 approved after PR.</td>
</tr>
<tr>
<td>CSD3 candidate</td>
<td>03/28/11</td>
<td>Jacques Durand</td>
<td>Removal of TA set section, addition of new Appendix A (test case derivation).</td>
</tr>
<tr>
<td>CSD3 candidate</td>
<td>04/25/11</td>
<td>Jacques Durand</td>
<td>Minor edits (RFC reference moved, re-title Section 2)</td>
</tr>
<tr>
<td>CSD4 candidate</td>
<td>05/10/11</td>
<td>Jacques Durand</td>
<td>Remove “language” from testAssertion and other normative source-related elements. Added an “id” subsection in 3.2. Other minor edits</td>
</tr>
</tbody>
</table>