Conformance Program Specification for the OASIS Security Assertion Markup Language (SAML) V1.1

Committee Specification, 27 May 2003

Document identifier:

sstc-saml-conform-1.1-cs-01

Location:

http://www.oasis-open.org/committees/documents.php?wg_abbrev=security

Editors:

Eve Maler, Sun Microsystems (eve.maler@sun.com)
Prateek Mishra, Netegrity (pmishra@netegrity.com)
Robert Philpott, RSA Security (rphilpott@rsasecurity.com)

Contributors:

Irving Reid, Baltimore Technologies
Krishna Sankar, Cisco Systems
Hal Lockhart, BEA Systems (formerly of Entegrity Solutions)
Robert Griffin, Entrust (former editor)
Marc Chanliau, Netegrity
Lynne Rosenthal, NIST
Mark Skall, NIST
Darren Platt, formerly with RSA Security
Charles Norwood, SAIC
Emily Xu, Sun Microsystems
Sai Allarvarpu, Sun Microsystems
Mike Myers, Traceroute Security
Mark O'Neill, Vordel
Tony Palmer, Vordel

Abstract:

This specification describes the program and technical requirements for SAML conformance.

Status:

This document is a **Committee Specification** of the OASIS Security Services Technical Committee. This document is updated periodically on no particular schedule. Send comments to the editors.

Committee members should send comments on this specification to the security-services@lists.oasis-open.org list. Others should subscribe to and send comments to the security-services-comment@lists.oasis-open.org list. To subscribe, send an email message to security-services-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

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 Security Services TC web page (http://www.oasis-open.org/committees/security/).

For information on errata discovered in this specification, please refer to the most recent errata document which can be found in the document repository at the Security Services TC web page (http://www.oasis-open.org/committees/security/).

Table of Contents

1 Introduction

This document describes the program and technical requirements for the SAML conformance system.

1.1 Scope of the Conformance Program

SAML deals with a rich set of functionalities ranging from assertions about acts of authentication to assertions for policy enforcement. Not all implementers will choose to implement all aspects of the SAML specifications. In order to achieve compatibility and interoperability, applications and software need to be measured for conformance in a uniform manner. The SAML conformance effort aims at fulfilling this need.

The deliverables of the SAML conformance effort include:

- Conformance clause, defining at a high level what conformance means for the SAML standard.
- Conformance program specification, defining how an implementation or application establishes conformance.
- Input to the creation of a conformance test suite. This is a high-level specification for a set of test programs, result files, and report generation tools that can be used by vendors of SAML-compliant software, buyers interested in confirming SAML compliance of software, and testing labs running conformance tests on behalf of vendors or buyers.

Section 2 of this document provides the SAML Conformance Clause. Section 3 deals with defining and specifying the process by which conformance to the SAML specification set can be demonstrated and certified. Section 4 elucidates the technical requirements that constitute conformance; this includes both the levels of conformance that can be demonstrated and the requirements for each of those levels of conformance. Section 5 describes what a test suite for SAML should include. Section 6 defines the services that may become available to assist in establishing conformance. Section 7 gives information for documents referenced in this specification.

1.2 Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "DOES", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 **[RFC2119]**:

> …they MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)…

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

2 Conformance Clause

The objectives of the SAML Conformance Clause are to:

- Ensure a common understanding of conformance and what is required to claim conformance
- Promote interoperability in the exchange of authentication and authorization information
- Promote uniformity in the development of conformance tests

The SAML Conformance Clause explicitly specifies all of the requirements that have to be satisfied to claim conformance to the SAML standard.

2.1 SAML Specification Set

The following four specifications, in addition to this SAML conformance program specification, comprise the Version 1.1 specification set for SAML:

- Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) **[SAMLCore]**
- Security Considerations for the OASIS Security Assertion Markup Language (SAML) **[SAMLSec]**
- Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) **[SAMLBind]**
- Glossary for the OASIS Security Assertion Markup Language (SAML) **[SAMLGloss]**

The SAML Core document also references the schema definitions for SAML assertions and protocols:

- Assertion schema **[SAMLAssertion]**
- Protocol schema **[SAMLProtocol]**

Although additional documents might use or reference the SAML standard (such as white papers, descriptions of custom profiles, and position papers referencing particular issues), they do not constitute part of the standard.

2.2 Declaration of SAML Conformance

Conformance to the SAML standard can be declared either for the entire standard or for a subset of the standard, based on the requirements that a given implementation or application claims to meet. That is, requirements can be applied at varying levels, so that a given implementation or application of the SAML standard can achieve clearly defined conformance with all or part of the entire set of specifications.

SAML conformance MUST be expressed in terms of which SAML bindings and profiles are supported by a given application or implementation. The application or implementation claiming conformance to the SAML standard MUST support the SOAP protocol binding for assertions containing at least one statement type. An application or implementation MAY also support the web browser profiles.

For any binding for which an application or implementation claims conformance, the level of conformance MUST then be specified in each of these dimensions:

- Whether the application or implementation acts as producer, consumer, or both producer and consumer of the SAML messages in the supported bindings and profiles.
- Which assertions and statements the application or implementation supports for each supported binding.

Table 1 shows the protocols, protocol bindings, and profiles applicable to each SAML assertion/statement type. For each SAML binding or profile to which an application or implementation claims conformance, the claim MUST stipulate whether the producer and/or consumer roles are supported and for which assertions and statements for those roles.

Note that the OASIS Web Services Security Technical Committee has produced a draft "SAML token profile" of the WSS specification **[WSS-SAML]**, which describes how to use SAML assertions to secure a web service message. This specification does not discuss conformance to that profile of SAML.

For example, an implementation consisting solely of an authentication authority responsible for generating assertions containing authentication statements and returning those assertions in response to a SOAP-over-HTTP request for assertion would correspond to the "producer role" for the SOAP over HTTP binding. If the implementation also supported the return of the assertion in the browser/artifact profile, then the "producer role" for that profile would also be supported.

A SAML protocol `<Request>` element may contain any one of `<AuthenticationQuery>`, `<AttributeQuery>`, or `<AuthorizationDecisionQuery>` elements, or, it may contain any number of `<AssertionIDReference>` or `<AssertionArtifact>` elements. For convenience, this document refers to a SAML request with an `<AuthenticationQuery>` element as an "authentication query", a request with an `<AttributeQuery>` element as an "attribute query", and a request with an `<AuthorizationDecisionQuery>` element as an "authorization decision query". SAML requests containing `<AssertionIDReference>` or `<AssertionArtifact>` elements are referred to simply as requests of those types.

Table 1: Protocol Bindings and Profiles for SAML Assertions

Binding or Profile	Consumer Role	Producer Role
SOAP over HTTP protocol binding	Send an authentication query to solicit an assertion containing an authentication statement from a producer; consume the returned response and assertion.	Produce an assertion containing an authentication statement and return a response containing the assertion to the consumer.
	Send an attribute query to solicit an assertion containing an attribute statement from a producer; consume the returned assertion.	Produce an assertion containing an attribute statement and return a response containing the assertion to the consumer.
	Send an authorization decision query to solicit an assertion containing an authorization decision statement from a producer; consume the returned assertion.	Produce an assertion containing an authorization decision statement and return a response containing the assertion to the consumer.
	Send an `<AssertionIDReference>` request to solicit one or more assertions with the associated assertion identifiers from a producer; consume the returned assertions.	Produce a response containing existing assertions with the requested assertion identifiers; send response to the consumer.
Browser/Artifact Profile	Receive one or more artifacts; send an `<AssertionArtifact>` request; ensure that returned assertions	Produce assertions including an SSO assertion and send corresponding artifacts to a consumer; on receiving

	include a singe sign-on assertion; consume the returned assertions.	an `<AssertionArtifact>` request, produce a response containing the associated assertions; send response to the consumer.
Browser/POST Profile	Receive a response message containing one or more assertions including an SSO assertion in a POST message and consume the assertions.	Produce assertions including an SSO assertion; produce a response message containing the assertions; transfer the response to a consumer via a POST message

An application or implementation should express its level of conformance in terminology such as the following:

> [Application or implementation] as both producer and consumer supports all SAML protocol bindings and profiles, for all assertions, statements, and required elements. No optional elements for the assertions, statements, bindings, and profiles are produced.

> [Application or implementation] as both producer and consumer supports the SOAP protocol binding for all queries, assertions, and statements. It produces the `<Conditions>` optional elements for all assertions in the SOAP protocol binding. It does not support the browser profiles for any assertion.

> [Application or implementation] as both producer and consumer supports the SOAP protocol binding for all assertions and statements. It also supports the browser/artifact profile and all required elements. No optional elements for the assertions, statements, bindings, and profiles are produced.

An application or implementation that claims conformance for a particular binding or profile MUST support all required elements of that binding or profile and of the assertions supported with that binding or profile. It MUST also state which assertions and statements are supported and which, if any, optional elements for that binding or profile and corresponding assertions and statements are supported.

2.3 Mandatory/Optional Elements in SAML Conformance

The SOAP protocol binding MUST be implemented by all implementations or applications claiming SAML conformance, for each assertion and statement type claimed as supported through a binding or profile.

The SAML schema and binding specifications include both mandatory and optional elements. A conforming application or implementation MUST be able to handle all valid SAML elements, including those that are optional. However, it does not have to produce those optional elements.

For example:

- An application or implementation that consumes assertions must be able to handle assertions that include the optional `<Condition>` element, such as by rejecting any conditions that it does not recognize.
- An application or implementation that produces assertions may, but is not required to, include the optional `<Condition>` element in those assertions.
- An application or implementation claiming support for an assertion must support the SOAP over HTTP protocol binding. It can also, optionally, implement the protocol by means of another binding.

The test cases for SAML conformance are intended to check for support of all valid SAML elements. They also check whether an implementation or application accepts and properly handles optional assertion elements (such as `<Condition>`) whose value the implementation or application does not recognize.

2.4 Impact of Extensions on SAML Conformance

SAML supports extensions to assertions, statements, protocols, protocol bindings, and profiles. An application or implementation MAY claim conformance to SAML only if its extensions (if any) meet the following requirements:

- Extensions MUST NOT re-define semantics for existing functions.
- Extensions MUST NOT alter the specified behavior of interfaces defined in the SAML specification set.
- Extensions MAY add additional behaviors.
- Extensions MUST NOT cause standard-conforming functions (i.e., functions that do not use the extensions) to execute incorrectly.

SAML bindings and profiles MAY be extended so long as the above conditions are met. If a system is extending SAML assertions or statements:

- The mechanism for determining application conformance and the extensions MUST be clearly described in the documentation, and the extensions MUST be marked as such;
- Extensions MUST follow the spirit, principles, and guidelines of the SAML specification set, that is, the specifications MUST be extended in a standard manner as defined in the extension fields.
- In the case where an implementation has added additional behaviors, the implementation MUST provide a mechanism whereby a conforming application shall be recognized as such, and be executed in an environment that supports the functional behavior defined in this specification set.

Extensions are outside the scope of conformance. There are no mechanisms specified to validate and verify the extensions.

2.5 Maximum Values of Unbounded Elements

The SAML schema supports a number of elements that can be specified multiple times in an assertion, request or response. An application or implementation claiming conformance MUST support at least the values listed in Table 2 below for each of the elements defined as "unbounded" in the SAML schema. In those cases where the maximum value is greater than the listed values, the application or implementation SHOULD state what that maximum supported value is.

However, some of the elements in the table can be nested, such that repeated elements have a multiplicative effect on the number of elements. For example, trees of nested unbounded elements include the following:

Response > Assertion > Statement (of various types)

Response > Assertion > Advice > Assertion

Response > Assertion > Conditions > AudienceRestrictionCondition > Audience

Response > Assertion > Statement > SubjectConfirmation > ConfirmationMethod

Response > Assertion > AttributeStatement > Attribute > AttributeValue

In a response containing 10 assertions, each with 10 AttributeStatements, each with 10 Attributes, each with 10 AttributeValues, this tree alone comprises 10,000 elements.

Therefore, in order to minimize the potential impact of nested unbounded elements, an application or implementation MAY limit the total number of elements supported in a given request, response or (when this is used in the POST profile) assertion to no more than 1000 total elements and still claim conformance to the SAML V1.1 specification set.

Table 2: Unbounded Elements

Element	Parent Element	Maximum Value
Statement (various types)	Assertion	1000
DoNotCacheCondition	Conditions	1000
AudienceRestrictionCondition	Conditions	1000
Audience	AudienceRestrictionCondition	1000
AssertionIDReference	Advice	1000
Assertion	Advice	1000
ConfirmationMethod	SubjectConfirmation	1000
AuthorityBinding	AuthenticationStatement	1000
Attribute	AttributeStatement	1000
AttributeValue	Attribute	1000
Action	AuthorizationDecisionStatement	1000
AssertionIDReference	Evidence	1000
Assertion	Evidence	1000
RespondWith	Request	1000
AssertionIDReference	Request	1000
AssertionArtifact	Request	1000
AttributeDesignator	AttributeQuery	1000
Action	AuthorizationDecisionQuery	1000
Assertion	Response	1000

3 Conformance Process

As discussed in the article “What is this thing called conformance” **[NIST/ITL]**, conformance can comprise any of several levels of formal process:

- **Conformance testing** (also called conformity assessment) is the execution of automated or non-automated scripts, processes, or other mechanisms to determine whether an application or implementation of a specification deviates from that specification. Conformance testing performed by implementors early on in the development process can find and correct their errors before the software reaches the marketplace, without necessarily being part of either a validation or a certification process.
- **Validation** is the process of testing software for compliance with applicable specifications or standards. The validation process consists of the steps necessary to perform the conformance testing by using an official test suite in a prescribed manner.
- **Certification** is the acknowledgment that a validation has been completed and the criteria established by the certifying organization for issuing a certificate have been met. Successful completion of certification results in the issuance of a certificate (or brand) indicating that the implementation conforms to the appropriate specification. It is important to note that certification cannot exist without validation, but validation can exist without certification.

The conformance process for SAML is based on validation rather than certification. That is, no certifying organization has been established with the responsible for issuing a statement of conformance with regard to an application or implementation. Therefore, an implementor who has validated SAML conformance by means of conformance testing MUST NOT use the term “certified for SAML conformance”. Until and if a certification process is in place, vendor declaration of validation will be the only means of asserting that conformance testing has been performed.

The conformance process does not stipulate whether validation is performed by the implementor, by a third party, or by the customer of an application or implementation. Rather, the conformance process describes the way in which conformance testing should be done in order to demonstrate that an application or implementation correctly performs the functionality specified in the standard. Validation achieved through the SAML conformance process provides software developers and users assurance and confidence that the product behaves as expected, performs functions in a known manner, and possesses the prescribed interface or format.

The Security Services Technical Committee is responsible for generating the materials that allow vendors, customers, and third parties to evaluate software for SAML conformance. These materials include documentation describing test cases, linked to use cases and requirements, included in this specification.

The test cases can be used to create a test suite that can be run against an implementation to demonstrate any of the several levels of conformance defined in the conformance clause of the SAML specification. The Security Services Technical Committee is not responsible for developing the test suite nor for testing of particular implementations.

3.1 Implementation and Application Conformance

SAML Conformance is applicable to:

- Implementations of SAML assertions, statements, protocols and bindings. These could be in the form of toolkits, products incorporating SAML components, or reference implementations that demonstrate the use of SAML components.

- Applications that produce or consume SAML protocol bindings or that execute on SAML implementations (for example, using a SAML toolkit to support multi-domain single sign-on)

A conforming **implementation** MUST meet all the following criteria:

1. The implementation MUST support all the required interfaces defined within the specification set for a given binding or profile. It MUST also specify which assertions and statements relevant to that binding or profile are supported. The implementation MUST support the functional behavior described in the specification.
2. The implementation MAY provide additional or enhanced facilities not required by this specification set. These nonstandard extensions MUST NOT alter the specified behavior of interfaces defined in this specification. They MAY add additional behaviors. In these circumstances, the implementation MUST provide a mechanism whereby a SAML conforming application shall be recognized as such, and be executed in an environment that supports the functional behavior defined in this specification set.

A conforming **application** MUST meet all the following criteria:

1. The application MUST be able to execute on any conforming implementation.
2. If an application requires a particular feature set that is not available on a specific implementation, then the application MUST act within the bounds of the SAML specification set, even though that means that the application does not perform any useful function. Specifically, the application MUST do no harm, and MUST correctly return resources and vacate memory upon discovery that a required element is not present.

3.2 Process for Declaring Conformance

The following process is to be followed in declaring that an application or implementation conforms to the SAML standard:

1. Determine which bindings and protocols will be asserted as conforming.
2. Implement the test suite for the conformance tests relevant to the conformance being claimed.
3. Validate the application or implementation by executing those conformance tests.
4. Send the statement claiming conformance to the Security Services Technical Committee so that it can be posted on the SAML web site. A statement of any bindings and profiles being used that are not part of the SAML standard should also be sent to the Security Services Technical Committee at the same time for posting on the SAML web site.

4 Technical Requirements for SAML Conformance

This section defines the technical criteria that apply to declaring conformance to the SAML standard. The requirements are specified as test cases, corresponding to the 12 possible subsets of conformance defined in Table 1.

Each test case includes:

- A description of the test purpose (that is, what is being tested – the conditions, requirements, or capabilities which are to be addressed by a particular test)
- The pass/fail criteria
- A reference to the requirement in the requirements document relevant to the test case
- A reference to the section in the specification set from which the test case is derived (that is, traceability back to the specification)

For each assertion and statement type, both required tests for producing and consuming the assertion, as well as tests related to protocols, bindings, and profiles, are specified.

4.1 Test Group 1 – SOAP over HTTP Protocol Binding

The test cases in this test group check for conformance to the SAML SOAP protocol binding. Any implementation or application claiming conformance to SAML MUST be able to execute these test cases successfully for the claimed assertion or assertions and role (producer or consumer), even if support for this protocol binding is incidental to the primary purposes of the application or implementation.

For convenience, assertions containing an authentication statement will be referred to in this section as *authentication assertions*, assertions containing an attribute statement as *attribute assertions*, and assertions containing an authorization decision statement as *authorization decision assertions*.

4.1.1 Test Case 1-1: SOAP Binding: Implementation-Under-Test Produces Valid Authentication Assertion in Valid Response to Authentication Query

Description: This test case requests and receives an authentication assertion created by an implementation-under-test using an authentication query in the SOAP binding. It then confirms that the authentication assertion returned by the implementation-under-test is valid for all required functionality.

Pass/Fail Criteria: The authentication assertion contains all required elements in the correct format and sequence, the authentication query is accepted by implementation-under-test, and the response contains all required elements in correct sequence.

Requirements Reference: R-AUTHN and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 3.1

Implementation Notes: The implementation-under-test executes the authentication assertion producer role.

4.1.2 Test Case 1-2: SOAP Binding: Implementation-Under-Test Consumes Valid Authentication Assertion, Requested in Valid Authentication Query

Description: This test case receives an authentication query created by an implementation-under-test in the SOAP binding. It confirms that the authentication query is valid for all required functionality. The test case returns an authentication assertion and confirms that the assertion is consumed.

Pass/Fail Criteria: The authentication query contains all required elements in the correct format and sequence; the authentication response and assertion are consumed.

Requirements Reference: R-AUTHN and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 3.1

Implementation Notes: The implementation-under-test executes the authentication assertion consumer role. It is up to the test program and implementation-under-test to determine how to validate that the assertion was consumed.

4.1.3 Test Case 1-3: SOAP Binding: Implementation-Under-Test Produces Valid Attribute Assertion in Valid Response to Attribute Query

Description: This test case requests and receives an attribute assertion created by an implementation-under-test using an attribute query in the SOAP binding. It then confirms that the attribute assertion returned by the implementation-under-test is valid for all required functionality.

Pass/Fail Criteria: The attribute assertion contains all required elements in the correct format and sequence, the attribute query is accepted by implementation-under-test, and the response contains all required elements in correct sequence.

Requirements Reference: R-AUTHZ and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 3.1

Implementation Notes: The implementation-under-test executes the attribute assertion producer role.

4.1.4 Test Case 1-4: SOAP Binding: Implementation-Under-Test Consumes Valid Attribute Assertion, Requested in Valid Attribute Query

Description: This test case receives an attribute query sent by an implementation-under-test in the SOAP binding. It confirms that the attribute query is valid for all required functionality. The test case then returns an attribute assertion and confirms that the assertion is consumed.

Pass/Fail Criteria: The attribute query contains all required elements in the correct format and sequence; attribute response and assertion are consumed.

Requirements Reference: R-AUTHZ and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 3.1

Implementation Notes: The implementation-under-test executes the attribute assertion consumer role. It is up to the test program and implementation-under-test to determine how to validate that assertion was consumed.

4.1.5 Test Case 1-5: SOAP Binding: Implementation-Under-Test Produces Valid Authorization Decision Assertion in Valid Response to Authorization Decision Query

Description: This test case requests and receives an authorization decision assertion created by an implementation-under-test using an authorization decision query in the SOAP binding. It then confirms that the authorization decision assertion returned by the implementation-under-test is valid for all required functionality.

Pass/Fail Criteria: The authorization decision assertion contains all required elements in the correct format and sequence, the authorization decision query is accepted by implementation-under-test, and the response contains all required elements in correct sequence.

Requirements Reference: R-AUTHZDECISION and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 3.1

Implementation Notes: The implementation-under-test executes the authorization decision assertion producer role.

4.1.6 Test Case 1-6: SOAP Binding: Implementation-Under-Test Consumes Valid Authorization Decision Assertion, Requested in Valid Authorization Decision Query

Description: This test case receives an authorization decision query created by an implementation-under-test in the SOAP binding. It confirms that the received authorization decision query is valid for all required functionality. It returns an authorization decision assertion to the implementation-under-test and confirms that the assertion is consumed.

Pass/Fail Criteria: The authorization decision query contains all required elements in the correct format and sequence; authorization decision response and assertion are consumed.

Requirements Reference: R-AUTHZDECISION and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 3.1

Implementation Notes: The implementation-under-test executes the authorization decision assertion consumer role. It is up to the test program and implementation-under-test to determine how to validate that assertion was consumed.

4.1.7 Test Case 1-7: SOAP Binding: Implementation-Under-Test Produces Valid Assertions in Valid Response to AssertionIDReference Request

Description: This test case requests and receives assertions created by an implementation-under-test using an AssertionIDReference request in the SOAP binding. It then confirms that the assertions returned by the implementation-under-test are valid for all required functionality.

Pass/Fail Criteria: The returned assertions contain all required elements in the correct format and sequence, the AssertionIDReference request is accepted by implementation-under-test, and the response contains all required elements in correct sequence.

Requirements Reference: R-AUTHN and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 3.1

Implementation Notes: The implementation-under-test executes the assertion producer role.

4.1.8 Test Case 1-8: SOAP Binding: Implementation-Under-Test Consumes Valid Assertions, Requested in Valid AssertionIDReference Request

Description: This test case receives an AssertionIDReference request in the SOAP binding created by an implementation-under-test. It confirms that the received AssertionIDReference request is valid for all required functionality. The test case returns the requested assertions and confirms that the assertions are consumed.

Pass/Fail Criteria: The AssertionIDReference request contains all required elements in the correct format and sequence; the response and assertions are consumed.

Requirements Reference: R-AUTHN and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 3.1

Implementation Notes: The implementation-under-test executes the assertion consumer role. It is up to the test program and implementation-under-test to determine how to validate that assertions were consumed.

4.2 Test Group 2 – Web Browser SSO Profiles

The test cases in this test group check for conformance to the web browser single sign-on (SSO) profiles of the SAML standard. Both the browser/artifact and browser/POST profiles are optional. Any implementation or application claiming conformance to the browser/artifact profile MUST be able to execute Test Case 2-1 successfully for the assertion producer role and/or Test Case 2-2 successfully for the assertion consumer role. Any implementation or application claiming conformance to the browser/POST profile MUST be able to execute Test Case 2-3 successfully for the assertion producer role and/or Test Case 2-4 successfully for the assertion consumer role.

4.2.1 Test Case 2-1: Browser/Artifact Profile: Valid Assertions Produced in Response to Valid AssertionArtifact Request

Description: This test case receives artifacts in a valid HTTP message from an implementation-under-test. The test case confirms that the artifacts are valid for all required functionality. It then uses the AssertionArtifact request in the SOAP binding to request and receive assertions created by an implementation-under-test corresponding to the artifacts. It then confirms that the returned assertions include an SSO assertion and is valid for all required functionality.

Pass/Fail Criteria: .Received artifacts have expected formats. AssertionArtifact request contains all required elements in correct format and sequence and is accepted by the implementation-under-test; An assertion is returned for every artifact in the AssertionArtifact request. Returned assertions include an SSO assertion.

Requirements Reference: R-AUTHN and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 4.1.1

Implementation Notes: Test program performs the destination site (consumer) operations for the profile; implementation-under-test performs source site (producer) operations.

4.2.2 Test Case 2-2: Browser/Artifact Profile: Valid Assertions Request Corresponding to Valid Artifacts Sent in Valid HTTP Message

Description: This test case sends valid artifacts in a valid HTTP message to an implementation-under-test. The test case then receives an AssertionArtifact request containing the artifacts from the implementation-under-test. It confirms that the AssertionArtifact request is valid for all required functionality, then returns the requested assertions to the implementation-under-test, and confirms that the assertion was consumed.

Pass/Fail Criteria: AssertionArtifact request contains all required elements in the correct format and sequence.

Requirements Reference: R-AUTHN and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 4.1.1

Implementation Notes: Test program performs the source site (producer) operations for the profile; implementation-under-test performs destination site (consumer) operations.

4.2.3 Test Case 2-3: Browser/POST Profile: Valid Assertions Received in Valid HTTP POST

Description: This test case receives an HTTP POST message from an implementation-under-test containing a SAML protocol response message with one or more assertions and including an SSO assertion and checks that the assertions are valid.

Pass/Fail Criteria: SSO assertion sent by implementation-under-test MUST contain all required information in the right sequence and format. Any optional information included (including conditions) MUST NOT compromise the validity of the required information.

Requirements Reference: R-AUTHN and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 4.1.2

Implementation Notes: Test program (consumer role) implementing this test case establishes successful execution of the test case by inspection of the format of the returned assertion.

4.2.4 Test Case 2-4: Browser/Post Profile: ValidAssertions Sent in Valid HTTP POST

Description: This test case sends a SAML protocol response message in an HTTP POST message to an implementation-under-test containing an SSO and other assertions and checks that the assertions are consumed.

Pass/Fail Criteria: Implementation-under-test allows access based on assertions it receives and consumes.

Requirements Reference: R-AUTHN and R-MULTIDOMAIN

Specification Reference: [SAMLCore] Sections 2.3, 2.4, and 3; **[SAMLBind]** Section 4.1.2

Implementation Notes: It is up to the test program and implementation-under-test to determine how to validate that assertion was consumed.

5 Test Suite

A test suite, which is the combination of test cases and test documentation, is used to check whether an implementation or application satisfies the requirements in the standard. The test cases, implemented by a test tool or a set of files (such as data, programs, scripts, or instructions for manual action), check each requirement in the specification to determine whether the results produced by the implementation or application match the expected results, as defined by the specification.

The test documentation describes how the testing is to be done and the directions for the tester to follow. Additionally, the documentation should be detailed enough so that testing of a given implementation can be repeated with no change in test results.

Conformance testing is black-box testing to test the functionality of an implementation. This means that the internal structure or the source code of a candidate implementation is not available to the tester. However, content and format of received or returned messages can be inspected as part of the determination of conformance.

Any test suite for SAML should consist of platform independent, non-biased, objective tests. Generally, a conformance test suite is a collection of combinations of legal and illegal inputs to the implementation being tested, together with a corresponding collection of expected results. Only the requirements specified in the standard are testable. A test suite should not check any implementation properties that are not described by the standard or set of standards. A test suite cannot require features that are optional in a standard, but if such features are present, a test suite could include tests for those features. A test suite does not assess the performance of an implementation unless performance requirements are specified in the specification, although implementation dependencies or machine dependencies can be demonstrated through the execution of the test cases.

The results of conformance testing apply only to the implementation and environment for which the tests are run. Test suites can be provided as a web-based system executed on a remote server, downloadable files for local execution, or a combination of remote and local access and execution. The method for providing and delivering the test suite depends on what is being tested as well as the objective for test suite use – that is, providing self-test capability or formal certification testing.

6 Conformance Services

The OASIS Security Services Technical Committee does not itself provide conformance services. As SAML test suites become available and experience with SAML identified appropriate conformance testing approaches, the Conformance Specification will describe the services which a conformance services organization should provide, including software services, releases, self-test kit, actual computer systems, facilities, web based interfaces, and availability.

7 References

[NIST/ITL]	"What is this thing called conformance" [Rosenthal, Brady; NIST/ITL Bulletin, January 2001] http://www.itl.nist.gov/div897/ctg/conformance/bulletin-conformance.htm.
[RFC2119]	S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.
[SAMLAssertion]	E. Maler et al., *Assertions Schema for the OASIS Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/, OASIS, May 2003.
[SAMLBind]	E. Maler et al., *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/, OASIS, May 2003.
[SAMLCore]	E. Maler et al., *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/, OASIS, May 2003.
[SAMLGloss]	E. Maler et al., *Glossary for the OASIS Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/, OASIS, May 2003.
[SAMLProtocol]	E. Maler et al., *Protocol Schema for the OASIS Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/, OASIS, May 2003.
[SAMLSec]	E. Maler et al., Security Considerations for the OASIS *Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/, OASIS, May 2003.
[WSS-SAML]	P. Hallam-Baker et al., *Web Services Security: SAML Token Profile*, OASIS, March 2003, http://www.oasis-open.org/committees/wss.

Appendix A. Acknowledgments

The editors would like to acknowledge the contributions of the OASIS SAML Technical Committee, whose voting members at the time of publication were:

- Frank Siebenlist, Argonne National Laboratory
- Irving Reid, Baltimore Technologies
- Hal Lockhart, BEA Systems
- Steven Lewis, Booz Allen Hamilton
- John Hughes, Entegrity Solutions
- Carlisle Adams, Entrust
- Jason Rouault, HP
- Maryann Hondo, IBM
- Anthony Nadalin, IBM
- Scott Cantor, Individual
- Bob Morgan, Individual
- Trevor Perrin, Individual
- Padraig Moloney, NASA
- Prateek Mishra, Netegrity (co-chair)
- Frederick Hirsch, Nokia
- Senthil Sengodan, Nokia
- Timo Skytta, Nokia
- Charles Knouse, Oblix
- Steve Anderson, OpenNetwork
- Simon Godik, OverXeer
- Rob Philpott, RSA Security (co-chair)
- Dipak Chopra, SAP
- Jahan Moreh, Sigaba
- Bhavna Bhatnagar, Sun Microsystems
- Jeff Hodges, Sun Microsystems
- Eve Maler, Sun Microsystems (coordinating editor)
- Emily Xu, Sun Microsystems
- Phillip Hallam-Baker, VeriSign

Appendix B. Notices