International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.1141

(06/2006)

SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY

Telecommunication security

Security Assertion Markup Language (SAML 2.0)

ITU-T Recommendation X.1141

ITU-T

International
Telecommunication
Union

ITU-T X-SERIES RECOMMENDATIONS

DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY

PUBLIC DATA NETWORKS	
Services and facilities	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalling and switching	X.50–X.89
Network aspects	X.90–X.149
Maintenance	X.150–X.179
Administrative arrangements	X.180–X.199
OPEN SYSTEMS INTERCONNECTION	
Model and notation	X.200–X.209
Service definitions	X.210–X.219
Connection-mode protocol specifications	X.220–X.229
Connectionless-mode protocol specifications	X.230–X.239
PICS proformas	X.240–X.259
Protocol Identification	X.260–X.269
Security Protocols	X.270–X.279
Layer Managed Objects	X.280–X.289
Conformance testing	X.290–X.299
INTERWORKING BETWEEN NETWORKS	
General	X.300–X.349
Satellite data transmission systems	X.350–X.369
IP-based networks	X.370–X.379
MESSAGE HANDLING SYSTEMS	X.400–X.499
DIRECTORY	X.500–X.599
OSI NETWORKING AND SYSTEM ASPECTS	
Networking	X.600–X.629
Efficiency	X.630–X.639
Quality of service	X.640–X.649
Naming, Addressing and Registration	X.650–X.679
Abstract Syntax Notation One (ASN.1)	X.680–X.699
OSI MANAGEMENT	
Systems Management framework and architecture	X.700–X.709
Management Communication Service and Protocol	X.710–X.719
Structure of Management Information	X.720–X.729
Management functions and ODMA functions	X.730–X.799
SECURITY	X.800–X.849
OSI APPLICATIONS	
Commitment, Concurrency and Recovery	X.850–X.859
Transaction processing	X.860–X.879
Remote operations	X.880–X.889
Generic applications of ASN.1	X.890–X.899
OPEN DISTRIBUTED PROCESSING	X.900–X.999
TELECOMMUNICATION SECURITY	**X.1000–**

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation X.1141

Security Assertion Markup Language (SAML 2.0)

Summary

SAML is an XML-based framework for exchanging security information. This security information is expressed in the form of assertions about subjects, where a subject is an entity (either human or computer) that has an identity in some security domain. A single assertion might contain several different internal statements about authentication, authorization and attributes. This Recommendation defines a protocol by which clients can request assertions from SAML authorities and get a response from them. This protocol, consisting of XML-based request and response message formats, can be bound to many different underlying communications and transport protocols; SAML currently defines one binding to SOAP over HTTP. In creating their responses, SAML authorities can use various sources of information, such as external policy stores and assertions that were received as input in requests. This Recommendation defines SAML assertions elements, subjects, conditions, processing rules and statements. Additionally, it develops a comprehensive SAML metadata profile that includes associated namespace, common data types, processing rules and signature processing. Several protocol bindings such as SOAP, PAOS (reverse SOAP), HTTP redirect, HTTP POST, among others, are also developed. This Recommendation provides a comprehensive list of SAML profiles such as web browser SSO profile and single logout profile to enable the wide adoption of SAML 2.0 in the industry. Guidelines for authentication context and conformance are also provided.

This Recommendation is technically equivalent and compatible with the OASIS SAML 2.0 standard.

Source

ITU-T Recommendation X.1141 was approved on 13 June 2006 by ITU-T Study Group 17 (2005-2008) under the ITU-T Recommendation A.8 procedure.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

CONTENTS

Page

ITU-T Recommendation X.1141

Security Assertion Markup Language (SAML 2.0)

1 Scope

This Recommendation defines the Security Assertion Markup Language (SAML 2.0). SAML defines the syntax and processing semantics of assertions made about a subject by a system entity. In the course of making, or relying upon such assertions, SAML system entities may use other protocols to communicate either regarding an assertion itself, or the subject of an assertion. This Recommendation defines the structure of SAML assertions, an associated set of protocols, in addition to the processing rules involved in managing a SAML system.

SAML assertions and protocol messages are encoded in XML and use XML namespaces. They are typically embedded in other structures for transport, such as HTTP POST requests or XML-encoded SOAP messages. This Recommendation also specifies SAML bindings that provide frameworks for the embedding and transport of SAML protocol messages. Furthermore, this Recommendation also provides a baseline set of profiles for the use of SAML assertions and protocols to accomplish specific use cases or achieve interoperability when using SAML features.

This Recommendation defines the following:

1) Conformance requirements for SAML;
2) Assertions and protocols for SAML:
    - SAML assertions schema;
    - SAML protocols schema.
3) Bindings for SAML;
4) Profiles for SAML:
    - SAML ECP profile schema;
    - SAML X.500/LDAP attribute profile schema;
    - SAML DCE PAC attribute profile schema;
    - SAML XACML attribute profile schema.
5) Metadata for SAML;
6) SAML metadata schema;
7) Authentication context for SAML.

2 References

The following Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision, and parties to agreements based on this Recommendation are encouraged to investigate the possibility of applying the most recent editions of the Recommendations and other references listed below. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations. The IETF maintains a list of RFCs, together with those that have been obsoleted by later RFCs. W3C, the Unicode Consortium and Liberty Alliance maintain a list of latest Recommendations and other publications.

– ITU-T Recommendation X.660 (2004) | ISO/IEC 9834-1:2005, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: General procedures and top arcs of the ASN.1 Object Identifier tree.*

– ITU-T Recommendation X.667 (2004) | ISO/IEC 9834-8:2005, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: Generation and Registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components.*

- ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.800 (1991), *Security architecture for Open Systems Interconnection for CCITT applications.*
- ITU-T Recommendation X.811 (1995) | ISO/IEC 10181-2:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: authentication framework.*
- ITU-T Recommendation X.812 (1995) | ISO/IEC 10181-3:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Access control framework.*
- ITU-T Recommendation X.1142 (2006), *eXtensible Access Control Markup Language (XACML 2.0).*
- IETF RFC 1034 (1987), *Domain Names – Concepts and Facilities*.
- IETF RFC 1510 (1993), *The Kerberos Network Authentication Service (V5).*
- IETF RFC 1750 (1994), *Randomness Recommendations for Security*.
- IETF RFC 1951 (1996), *DEFLATE Compressed Data Format Specification Version 1.3*.
- IETF RFC 1991 (1996), *PGP Message Exchange Formats.*
- IETF RFC 2045 (1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*.
- IETF RFC 2119 (1997), *Keywords for use in RFCs to Indicate Requirement Levels*.
- IETF RFC 2246 (1999), *The TLS Protocol Version 1.0*.
- IETF RFC 2253 (1997), *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*.
- IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax*.
- IETF RFC 2535 (1999), *Domain Name System Security Extensions*.
- IETF RFC 2616 (1999), *Hypertext Transfer Protocol – HTTP/1.1*.
- IETF RFC 2617 (1999), *HTTP Authentication: Basic and Digest Access Authentication*.
- IETF RFC 2798 (2000), *Definition of the inetOrgPerson LDAP Object Class*.
- IETF RFC 2828 (2000), *Internet Security Glossary*.
- IETF RFC 2914 (2000), *Congestion Control Principles*.
- IETF RFC 2915 (2000), *The Naming Authority Pointer (NAPTR) DNS Resource Record*.
- IETF RFC 2945 (2000), *The SRP Authentication and Key Exchange System*.
- IETF RFC 2965 (2000), *HTTP State Management Mechanism*.
- IETF RFC 3023 (2001), *XML Media Types*.
- IETF RFC 3061 (2001), *A URN Namespace of Object Identifiers*.
- IETF RFC 3075 (2001), *XML-Signature Syntax and Processing*.
- IETF RFC 3377 (2002), *Lightweight Directory Access Protocol (v3): Technical Specification*.
- IETF RFC 3403 (2002), *Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database*.
- IETF RFC 3513 (2003), *Internet Protocol Version 6 (IPv6) Addressing Architecture*.
- IETF RFC 3546 (2003), *Transport Layer Security (TLS) Extensions*.
- IETF RFC 3923 (2004), *End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP).*
- IETF RFC 4122 (2005), *A Universally Unique IDentifier (UUID) URN Namespace*.
- Liberty Alliance POAS:2003, R. Aarts, *Liberty Reverse HTTP Binding for SOAP Specification Version 1.0, Liberty Alliance Project.*
- OASIS WSS:2006, WS-Security Core Specification 1.1.
- UNICODE-C, M. Davis; M. J. Dürst: *Unicode Normalization Forms.* UNICODE Consortium, March 2001.
- W3C Canonicalization:2002, *Exclusive XML Canonicalization Version 1.0,* W3C Recommendation, Copyright © [18 July 2002] World Wide Web Consortium, (Massachusetts Institute of Technology,

Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/xml-exc-c14n/.

– W3C Character Model:2005, *Character Model for the World Wide Web 1.0: Fundamentals*, W3C Recommendation, Copyright © [15 February 2005] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/2005/REC-charmod-20050215/.

– W3C Datatypes:2001, *XML Schema Part 2: Datatypes*, W3C Recommendation, Copyright © [2 May 2001] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

– W3C Encryption:2002, *XML Encryption Syntax and Processing*, W3C Recommendation, Copyright © [10 December 2002] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/.

– W3C Web Services Glossary:2004, *Web Services Glossary*, W3C Note, Copyright © [11 February 2004] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/ws-gloss/.

– W3C HTML:1999, *HTML 4.01 Specification*, W3C Recommendation, Copyright © [24 December 1999] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/REC-html40/.

– W3C Namespaces:1999, *Namespaces in XML*, W3C Recommendation, Copyright © [14 January 1999] World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/REC-xml-names/.

– W3C Primer:2005, *SOAP Version 1.2 Part 0: Primer*, W3C Recommendation, Copyright © [24 June 2005] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/2003/REC-soap12-part0-20030624/.

– W3C Signature:2002, *XML Signature Syntax and Processing*, W3C Recommendation, Copyright © [12 February 2002] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/xmldsigcore/.

– W3C Signature Schema:2001, *XML Signature Schema*, W3C Recommendation, Copyright © [1 March 2001] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd.

– W3C String:1998, *Requirements for String Identity Matching and String Indexing*, W3C Note, Copyright © [10 July 1998] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University, http://www.w3.org/TR/WD-charreq.

– W3C SOAP:2000, *Simple Object Access Protocol (SOAP) 1.1*, W3C Note, Copyright © [08 May 2000] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/2000/NOTE-SOAP-20000508.

– W3C XHTML:2002, *The Extensible HyperText Markup Language (Second Edition)*, W3C Recommendation, Copyright © [1 August 2002] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University*)*, http://www.w3.org/TR/xhtml1/.

– W3C XML 1.0:2004, *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation, Copyright © [4 February 2004] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/REC-xml/.

– W3C XML Schema Part 1:2001, *XML Schema Part 1: Structures*, W3C Recommendation, Copyright © [2 May 2001] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.

NOTE – The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

3 Definitions

For the purposes of this Recommendation, the following definitions apply.

3.1 Imported definitions

3.1.1 This Recommendation uses the following term defined in ITU-T Rec. X.667:

a) UUID.

3.1.2 This Recommendation uses the following terms defined in ITU-T Rec. X.680:

a) Object identifier;

b) Open type notation.

3.1.3 This Recommendation uses the following term defined in ITU-T Rec. X.811:

a) Principle.

3.1.4 This Recommendation uses the following terms defined in ITU-T Rec. X.812:

a) Access control information;

b) User.

3.1.5 This Recommendation uses the following terms defined in W3C Web Services Glossary:

a) Initial SOAP sender;

b) Namespace;

c) Ultimate SOAP receiver;

d) XML schema.

3.1.6 This Recommendation uses the following terms defined in IETF RFC 2828:

a) Access;

b) Access control;

c) Proxy;

d) Proxy server;

f) Pull;

e) Push;

g) Security architecture;

h) Security policy;

i) Security service.

3.1.7 This Recommendation uses the following terms defined in IETF RFC 2396:

a) Uniform resource identifier (URI);

b) URI reference.

3.2 Additional definitions

3.2.1 **access rights**: A description of the type of authorized interactions a subject can have with a resource. Examples include read, write, execute, add, modify, and delete.

3.2.2 **account**: A formal business agreement for providing regular dealings and services between a principal and a business service provider.

3.2.3 **account linkage**: A method of relating accounts at two different providers that represent the same principal so that the providers can communicate about the principal. Account linkage can be established through the sharing of attributes or through identity federation.

3.2.4 **active role**: A role that a system entity has donned when performing some operation, for example accessing a resource.

3.2.5 **administrative domain**: An environment or context that is defined by some combination of one or more administrative policies, Internet domain name registrations, civil legal entities (for example, individuals, corporations, or other formally organized entities), plus a collection of hosts, network devices and the interconnecting networks (and possibly other traits), plus (often various) network services and applications running upon them. An administrative domain may contain or define one or more security domains. An administrative domain may encompass a single site or multiple sites. The traits defining an administrative domain may, and in many cases will, evolve over time. Administrative domains may interact and enter into agreements for providing and/or consuming services across administrative domain boundaries.

3.2.6 **administrator**: A person who installs or maintains a system or who uses it to manage system entities, users, and/or content. An administrator is typically affiliated with a particular administrative domain and may be affiliated with more than one administrative domain.

3.2.7 **affiliation; affiliation group**: A set of system entities that share a single namespace (in the federated sense) of identifiers for principals.

3.2.8 **anonymity**: The quality or state of being anonymous, which is the condition of having a name or identity that is unknown or concealed.

3.2.9 **asserting party**: Formally, the administrative domain that hosts one or more SAML authorities. Informally, an instance of a SAML authority.

3.2.10 **assertion**: A piece of data produced by a SAML authority regarding either an act of authentication performed on a subject, attribute information about the subject, or authorization data applying to the subject with respect to a specified resource.

3.2.11 **attribute**: A distinct characteristic of an object. For real world objects, attributes are often specified in terms of physical traits, such as size, shape, weight, and colour. Objects in cyberspace might have attributes describing size, type of encoding, network address, and so on. Attributes are often represented as pairs of "attribute name" and "attribute value(s)", e.g., "foo" has the value 'bar', "count" has the value 1, "gizmo" has the values "frob" and "2".

3.2.12 **attribute assertion**: An assertion that conveys information about attributes of a subject.

3.2.13 **attribute authority**: A system entity that produces attribute assertions.

3.2.14 **authentication**: Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be within a degree of confidence.

3.2.15 **authentication assertion**: An assertion that conveys information about a successful act of authentication that took place for a subject.

3.2.16 **authentication authority**: A system entity that produces authentication assertions.

3.2.17 **authorization**: The process of determining, by evaluating applicable access control information, whether a subject is allowed to have the specified types of access to a particular resource. Usually, authorization is in the context of authentication. Once a subject is authenticated, it may be authorized to perform different types of access.

3.2.18 **authorization decision**: The result of an act of authorization. The result may be negative, that is, it may indicate that the subject is not allowed any access to the resource.

3.2.19 **authorization decision assertion**: An assertion that conveys information about an authorization decision.

3.2.20 **back channel**: Back channel refers to direct communications between two system entities without "redirecting" messages through another system entity such as an HTTP client (e.g., a user agent).

3.2.21 **binding; protocol binding**: Generically, a specification of the mapping of some given protocol's messages, and perhaps message exchange patterns, onto another protocol, in a concrete fashion. For example, the mapping of the SAML `<AuthnRequest>` message onto HTTP is one example of a binding. The mapping of that same SAML message onto SOAP is another binding. In the SAML context, each binding is given a name in the pattern "SAML xxx binding".

3.2.22 **credentials**: Data that is transferred to establish a claimed principal identity.

3.2.23 **end user**: A natural person who makes use of resources for application purposes.

3.2.24 **entity**: See system entity.

3.2.25 **federate**: To link or bind two or more entities together.

3.2.26 **federation**: This term is used in two senses:

1) The act of establishing a relationship between two entities.
2) An association comprising any number of service providers and identity providers.

3.2.27 **federated identity**: A principal's identity is said to be federated between a set of providers when there is an agreement between the providers on a set of identifiers and/or attributes to use to refer to the principal.

3.2.28 front channel: Front channel refers to the "communications channel" that can be effected between two HTTP-speaking servers by employing "HTTP redirect" messages and thus passing messages to each other via a user agent, e.g., a web browser, or any other HTTP client.

3.2.29 identifier: A data object (for example, a string) mapped to a system entity that uniquely refers to the system entity. A system entity may have multiple distinct identifiers referring to it. An identifier is essentially a "distinguished attribute" of an entity.

3.2.30 identity: The essence of an entity. One's identity is often described by one's characteristics, among which may be any number of identifiers.

3.2.31 identity defederation: The action occurring when Providers agree to stop referring to a principal via a certain set of identifiers and/or attributes.

3.2.32 identity federation: The act of creating a federated identity on behalf of a principal.

3.2.33 identity provider: A kind of service provider that creates, maintains, and manages identity information for principals and provides principal authentication to other service providers within a federation, such as with web browser profiles.

3.2.34 identity provider lite: A kind of service provider that creates, maintains, and manages identity information for principals and provides principal authentication to other service providers within a federation, using only required portions of SAML.

3.2.35 login, logon, sign-on: The process whereby a user presents credentials to an authentication authority, establishes a simple session, and optionally establishes a rich session.

3.2.36 logout, logoff, sign-off: The process whereby a user signifies desire to terminate a simple session or rich session.

3.2.37 markup language: A set of XML elements and XML attributes to be applied to the structure of an XML document for a specific purpose. A markup language is typically defined by means of a set of XML schemas and accompanying documentation.

3.2.38 name qualifier: A string that disambiguates an identifier that may be used in more than one namespace (in the federated sense) to represent different principals.

3.2.39 party: Informally, one or more principals participating in some process or communication, such as receiving an assertion or accessing a resource.

3.2.40 persistent pseudonym: A privacy-preserving name identifier assigned by a provider to identify a principal to a given relying party for an extended period of time that spans multiple sessions; can be used to represent an identity federation.

3.2.41 policy decision point (PDP): A system entity that makes authorization decisions for itself or for other system entities that request such decisions. For example, a SAML PDP consumes authorization decision requests, and produces authorization decision assertions in response. A PDP is an "authorization decision authority".

3.2.42 policy enforcement point (PEP): A system entity that requests and subsequently enforces authorization decisions. For example, a SAML PEP sends authorization decision requests to a PDP, and consumes the authorization decision assertions sent in response.

3.2.43 principal identity: A representation of a principal's identity, typically an identifier.

3.2.44 profile: A set of rules for one of several purposes; each set is given a name in the pattern "xxx profile of SAML" or "xxx SAML profile":

1) Rules for how to embed assertions into and extract them from a protocol or other context of use.
2) Rules for using SAML protocol messages in a particular context of use.
3) Rules for mapping attributes expressed in SAML to another attribute representation system. Such a set of rules is known as an "attribute profile".

3.2.45 protocol binding: See "Binding".

3.2.46 provider: A generic way to refer to both identity providers and service providers.

3.2.47 relying party: A system entity that decides to take an action based on information from another system entity. For example, a SAML relying party depends on receiving assertions from an asserting party (a SAML authority) about a subject.

3.2.48 requester: A system entity that utilizes the SAML protocol to request services from another system entity (a SAML authority, a responder). The term "client" for this notion is not used because many system entities simultaneously or serially act as both clients and servers. In cases where the SOAP binding for SAML is being used, the SAML requester is architecturally distinct from the initial SOAP sender.

3.2.49 resource: Data contained in an information system (for example, in the form of files, information in memory, etc.), as well as:

1) A service provided by a system.
2) An item of system equipment (in other words, a system component such as hardware, firmware, software, or documentation).

3.2.50 responder: A system entity (a SAML authority) that utilizes the SAML protocol to respond to a request for services from another system entity (a requester). The term "server" for this notion is not used because many system entities simultaneously or serially act as both clients and servers. In cases where the SOAP binding for SAML is being used, the SAML responder is architecturally distinct from the ultimate SOAP receiver.

3.2.51 role: Dictionaries define a role as "a character or part played by a performer" or "a function or position". System entities do various types of roles serially and/or simultaneously, for example, active roles and passive roles. The notion of an Administrator is often an example of a role.

3.2.52 SAML artifact: A small, fixed-size, structured data object pointing to a typically larger, variably-sized SAML protocol message. SAML artifacts are designed to be embedded in URLs and conveyed in HTTP messages, such as HTTP response messages with "3xx Redirection" status codes, and subsequent HTTP GET messages. In this way, a service provider may indirectly, via a user agent, convey a SAML artifact to another provider, who may subsequently dereference the SAML artifact via a direct interaction with the supplying provider, and obtain the SAML Protocol message.

3.2.53 SAML authority: An abstract system entity in the SAML domain model that issues assertions. See also attribute authority, authentication authority, and policy decision point (PDP).

3.2.54 security: A collection of safeguards that ensure the confidentiality of information, protect the systems or networks used to process it, and control access to them. Security typically encompasses the concepts of secrecy, confidentiality, integrity, and availability. It is intended to ensure that a system resists potentially correlated attacks.

3.2.55 security assertion: An assertion that is scrutinized in the context of security architecture.

3.2.56 security context: With respect to an individual SAML protocol message, the message's security context is the semantic union of the message's security header blocks (if any) along with other security mechanisms that may be employed in the message's delivery to a recipient. With respect to the latter, an example is security mechanisms employed at lower network stack layers such as HTTP, TLS and IPSec.

3.2.57 security domain: An environment or context that is defined by security models and security architecture, including a set of resources and set of system entities that are authorized to access the resources. One or more security domains may reside in a single administrative domain. The traits defining a given security domain typically evolve over time.

3.2.58 security policy expression: A mapping of principal identities and/or attributes thereof with allowable actions. Security policy expressions are often essentially access control lists.

3.2.59 service provider: A role donned by a system entity where the system entity provides services to principals or other system entities.

3.2.60 service provider lite: A role donned by a system entity where the system entity provides services to principals or other system entities using only required portion of SAML protocol.

3.2.61 session: A lasting interaction between system entities, often involving a Principal, typified by the maintenance of some state of the interaction for the duration of the interaction.

3.2.62 session authority: A role donned by a system entity when it maintains state related to sessions.

3.2.63 session participant: A role donned by a system entity when it participates in a session with at least a session authority.

3.2.64 sign-off: See "logout".

3.2.65 sign-on: See "login".

3.2.66 site: An informal term for an administrative domain in geographical or DNS name sense. It may refer to a particular geographical or topological portion of an administrative domain, or it may encompass multiple administrative domains, as may be the case at an ASP site.

3.2.67 subject: A principal in the context of a security domain. SAML assertions make declarations about subjects.

3.2.68 system entity; entity: An active element of a computer/network system. For example, an automated process or set of processes, a subsystem, a person or group of persons that incorporates a distinct set of functionality.

3.2.69 time-out: A period of time after which some condition becomes true if some event has not occurred. For example, a session that is terminated because its state has been inactive for a specified period of time is said to "time out".

3.2.70 transient pseudonym: A privacy-preserving identifier assigned by an identity provider to identify a principal to a given relying party for a relatively short period of time that need not span multiple sessions.

3.2.71 XML attribute: An XML data structure that is embedded in the start-tag of an XML element and that has a name and a value.

3.2.72 XML element: An XML data structure that is hierarchically arranged among other such structures in an XML document and is indicated by either a start-tag and end-tag or an empty tag.

4 Abbreviations

For the purposes of this Recommendation, the following abbreviations apply.

AA	Attribute Authority
ASN.1	Abstract Syntax Notation One
ASP	Application Service Provider
CA	Certification Authority
CMP	Certificate Management Protocol
CRL	Certificate Revocation List
DCE	Distributed Computing Environment
DDDS	Dynamic Delegation Discovery System
DNS	Domain Name System
ECP	Enhanced Client/Proxy
HTTP	HyperText Transfer Protocol
HTTPS	Secure HyperText Transfer Protocol
IdP	Identity Provider
IdP Lite	Identity Provider Lite
IP	Internet Protocol
IPSec	Internet Protocol Security
MD5	Message Digest algorithm 5
MIME	Multipurpose Internet Mail Extensions
NAPTR	Naming Authority PoinTeR
OID	Object IDentifier
PAC	Privilege Attribute Certificates
PAOS	Reverse SOAP
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PGP	Pretty Good Privacy
PKI	Public-Key Infrastructure
POP	Proof of Possession
RA	Registration Authority
RSA	Rivest, Shamir, Adleman (public key algorithm)

SHA-1	Secure Hash Algorithm 1
SP	Service Provider
SPKI	Simple Public Key Infrastructure
SP Lite	Service Provider Lite
SSO	Single Sign On
TLS	Transport Layer Security protocol
URI	Uniform Resource Identifier
UTC	Coordinated Universal Time
UUID	Universal Unique IDentifier
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

5 Conventions

This Recommendation uses the keywords "must", "must not", "required", "shall", "shall not", "should", "should not", "recommended", "may", and "optional". In this Recommendation, these terms are to be interpreted as described in IETF RFC 2119.

This Recommendation uses XML schema documents conforming to W3C XML Schema Part 1, W3C XML Schema Part 2 and normative text of those specifications to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages. In cases of disagreement between the SAML schema documents and schema listings in this Recommendation, the schema documents take precedence. Note that, in some cases, this Recommendation imposes constraints beyond those indicated by the schema documents.

6 Overview

This Recommendation is intended to specify the Security Assertion Markup Language version 2 (SAML v2.0). It defines the syntax and processing semantics of assertions made about a subject by a system entity. In the course of making, or relying upon such assertions, SAML system entities may use other protocols to communicate either regarding an assertion itself, or the subject of an assertion. This Recommendation defines both the structure of SAML assertions, and an associated set of protocols, in addition to the processing rules involved in managing a SAML system.

SAML assertions and protocol messages are encoded in XML and use XML namespaces. They are typically embedded in other structures for transport, such as HTTP POST requests or XML-encoded SOAP messages. Clause 7 defines the common data types SAML uses. Clause 8 provides a framework for SAML assertions and protocols. Clause 9 describes the SAML metadata model. Clause 10 develops frameworks for the embedding and transport of SAML protocol messages. Clause 11 provides a baseline set of profiles for the use of SAML assertions and protocols to accomplish specific use cases or achieve interoperability when using SAML features. Clause 12 discusses authentication context for SAML. In particular, the following contexts are specified:

- SAML authentication context schema;
- SAML authentication context schema types;
- SAML context class schema for Internet protocol;
- SAML context class schema for Internet protocol password;
- SAML context class schema for Kerberos;
- SAML context class schema for mobile one-factor unregistered;
- SAML context class schema for mobile two-factor unregistered;
- SAML context class schema for mobile one-factor contract;
- SAML context class schema for mobile two-factor contract;
- SAML context class schema for password;
- SAML context class schema for password protected transport;
- SAML context class schema for previous session;
- SAML context class schema for public key – X.509;
- SAML context class schema for public key – PGP;

- SAML context class schema for public key – SPKI;
- SAML context class schema for public key – XML signature;
- SAML context class schema for smartcard;
- SAML context class schema for smartcard PKI;
- SAML context class schema for software PKI;
- SAML context class schema for telephony;
- SAML context class schema for telephony (nomadic);
- SAML context class schema for telephony (personalized);
- SAML context class schema for telephony (authenticated);
- SAML context class schema for secure remote password;
- SAML context class schema for SL/TLS certificate-based client authentication;
- SAML context class schema for time sync token.

Clause 13 provides a framework for implementers of SAML that need to be followed in order to ensure conformance. In clause 13, conformance requirements are discussed including operational modes and security models. Annex A includes a list of all associated SAML schemas.

7 Common data types

The following clauses define how to use and interpret common data types that appear throughout the SAML schemas.

7.1 String values

All SAML string values have the type **xs:string**, which is built in W3C XML Datatypes. Unless otherwise noted in this Recommendation, all strings in SAML messages must consist of at least one non-whitespace character.

Unless otherwise noted in this Recommendation or particular profiles, all elements in SAML documents that have the XML Schema **xs:string** type, or a type derived from that, must be compared using an exact binary comparison. In particular, SAML implementations and deployments must not depend on case-insensitive string comparisons, normalization or trimming of whitespace, or conversion of locale-specific formats such as numbers or currency. This requirement is intended to conform to W3C String.

If an implementation is comparing values that are represented using different character encodings, the implementation must use a comparison method that returns the same result as converting both values to the Unicode character encoding, Normalization Form C, and then performing an exact binary comparison. This requirement is intended to conform to the W3C Character Model in particular the rules for Unicode-normalized text.

Applications that compare data received in SAML documents to data from external sources must take into account the normalization rules specified for XML. Text contained within elements is normalized so that line endings are represented using linefeed characters (ASCII CODE 10_{Decimal}). XML attribute values defined as strings (or types derived from strings) are normalized as described in W3C XML 1.0, 3.3.3. All whitespace characters are replaced with blanks (ASCII CODE 32_{Decimal}).

This Recommendation does not define collation or sorting order for XML attribute values or element content. SAML implementations must not depend on specific sorting orders for values, because these can differ depending on the local settings of the hosts involved.

7.2 URI values

All SAML URI reference values have the type **xs:anyURI**, which is built into W3C XML Datatypes.

Unless otherwise indicated in this Recommendation, all URI reference values used within SAML-defined elements or attributes must consist of at least one non-whitespace character, and are required to be absolute.

This Recommendation makes extensive use of URI references as identifiers, such as status codes, format types, attribute and system entity names, etc. Therefore, it is essential that the values be both unique and consistent, such that the same URI is never used at different times to represent different underlying information.

7.3 Time values

All SAML time values have the type **xs:dateTime**, which is built into W3C XML Datatypes and must be expressed in UTC form, with no time zone component.

SAML system entities should not rely on time resolution finer than milliseconds. Implementations must not generate time instants that specify leap seconds.

7.4 ID and ID reference values

The **xs:ID** simple type is used to declare SAML identifiers for assertions, requests, and responses. Values declared to be of type **xs:ID** in this Recommendation must satisfy the following properties in addition to those imposed by the definition of the **xs:ID** type itself:

- Any party that assigns an identifier must ensure that there is negligible probability that that party or any other party will accidentally assign the same identifier to a different data object.
- Where a data object declares that it has a particular identifier, there must be exactly one such declaration.

The mechanism by which a SAML system entity ensures that the identifier is unique is left to the implementation. In the case that a random or pseudo-random technique is employed, the probability of two randomly chosen identifiers being identical must be less than or equal to 2^{-128} and should be less than or equal to 2^{-160}. This requirement may be met by encoding a randomly chosen value between 128 and 160 bits in length. The encoding must conform to the rules defining the **xs:ID** datatype. A pseudo-random generator must be seeded with unique material in order to ensure the desired uniqueness properties between different systems.

The **xs:NCName** simple type is used in SAML to reference identifiers of type **xs:ID** since **xs:IDREF** cannot be used for this purpose. In SAML, the element referred to by a SAML identifier reference might actually be defined in a document separate from that in which the identifier reference is used. Using **xs:IDREF** would violate the requirement that its value match the value of an ID attribute on some element in the same XML document.

8 SAML assertions and protocols

SAML defines the syntax and processing semantics of assertions made about a subject by a system entity. In the course of making, or relying upon such assertions, SAML system entities may use other protocols to communicate an assertion itself, or the subject of an assertion. This clause defines the structure of SAML assertions, an associated set of protocols, in addition to the processing rules involved in managing a SAML system.

SAML assertions and protocol messages are encoded in XML (see W3C XML 1.0) and use XML namespaces (see W3C Namespaces). They are typically embedded in other structures for transport, such as HTTP POST requests or XML-encoded SOAP messages. Clause 10 provides frameworks for the embedding and transport of SAML protocol messages. Clause 11 provides a baseline set of profiles for the use of SAML assertions and protocols to accomplish specific use cases or achieve interoperability when using SAML features.

8.1 SAML assertions

An assertion is a package of information that supplies zero or more statements made by a **SAML authority**; SAML authorities are sometimes referred to as **asserting parties** in discussions of assertion generation and exchange, and system entities that use received assertions are known as **relying parties**. (These terms are different from **requester** and **responder**, which are reserved for discussions of SAML protocol message exchange.)

SAML assertions are usually made about a **subject**, represented by the <`Subject`> element. However, the <`Subject`> element is optional, and other specifications and profiles may utilize the SAML assertion structure to make similar statements without specifying a subject, or possibly specifying the subject in an alternate way. Typically there are a number of **service providers** that can make use of assertions about a subject in order to control access and provide customized service, and accordingly they become the relying parties of an asserting party called an **identity provider**.

This Recommendation defines three different kinds of assertion statements that can be created by a SAML authority. All SAML-defined statements are associated with a subject. The three kinds of statement defined in this Recommendation are:

- **Authentication**: The assertion subject was authenticated by a particular means at a particular time.
- **Attribute**: The assertion subject is associated with the supplied attributes.
- **Authorization decision**: A request to allow the assertion subject to access the specified resource has been granted or denied.

NOTE (informative) – PE13 (see OASIS PE:2006) suggests to append "or is indeterminate" to the above paragraph.

The outer structure of an assertion is generic, providing information that is common to all of the statements within it. Within an assertion, a series of inner elements describe the authentication, attribute, authorization decision, or user-defined statements containing the specifics.

Extensions are permitted by the SAML assertion schema as described in 8.6, allowing user-defined extensions to assertions and statements, as well as allowing the definition of new kinds of assertions and statements.

8.1.1 Schema header and namespace declarations

The following schema fragment defines the XML namespaces and other header information for the assertion schema:

```
<schema targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
20020212/xmldsig-core-schema.xsd"/>
    <import namespace="http://www.w3.org/2001/04/xmlenc#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-
20021210/xenc-schema.xsd"/>
    <annotation>
        <documentation>
            Document identifier: saml-schema-assertion-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
            V1.0 (November, 2002):
              Initial Standard Schema.
            V1.1 (September, 2003):
              Updates within the same V1.0 namespace.
            V2.0 (March, 2005):
              New assertion schema for SAML V2.0 namespace.
        </documentation>
    </annotation>
…
</schema>
```

8.1.2 Name identifiers

The following clauses define the SAML constructs that contain descriptive identifiers for subjects and the issuers of assertions and protocol messages.

There are a number of circumstances in SAML in which it is useful for two system entities to communicate regarding a third party; for example, the SAML authentication request protocol enables third-party authentication of a subject. Thus, it is useful to establish a means by which parties may be associated with identifiers that are meaningful to each of the parties. In some cases, it will be necessary to limit the scope within which an identifier is used to a small set of system entities (to preserve the privacy of a subject, for example). Similar identifiers may also be used to refer to the issuer of a SAML protocol message or assertion.

It is possible that two or more system entities may use the same name identifier value when referring to different identities. Thus, each entity may have a different understanding of that same name. SAML provides **name qualifiers** to disambiguate a name identifier by effectively placing it in a federated **namespace** related to the name qualifiers. SAML v2.0 allows an identifier to be qualified in terms of both an asserting party and a particular relying party or affiliation, allowing identifiers to exhibit pair-wise semantics, when required.

Name identifiers may also be encrypted to further improve their privacy-preserving characteristics, particularly in cases where the identifier may be transmitted via an intermediary.

NOTE – To avoid use of relatively advanced XML schema constructs, the various types of identifier elements do not share a common type hierarchy.

8.1.2.1 Element <BaseID>

The <BaseID> element is an extension point that allows applications to add new kinds of identifiers. Its **BaseIDAbstractType** complex type is abstract and is thus usable only as the base of a derived type. It includes the following attributes for use by extended identifier representations:

– NameQualifier [Optional]

The security or administrative domain that qualifies the identifier. This attribute provides a means to federate identifiers from disparate user stores without collision.

– SPNameQualifier [Optional]

Further qualifies an identifier with the name of a service provider or affiliation of providers. This attribute provides an additional means to federate identifiers on the basis of the relying party or parties.

The NameQualifier and SPNameQualifier attributes should be omitted unless the identifier's type definition explicitly defines their use and semantics.

The following schema fragment defines the <BaseID> element and its **BaseIDAbstractType** complex type:

```
<attributeGroup name="IDNameQualifiers">
    <attribute name="NameQualifier" type="string" use="optional"/>
    <attribute name="SPNameQualifier" type="string" use="optional"/>
</attributeGroup>
<element name="BaseID" type="saml:BaseIDAbstractType"/>
<complexType name="BaseIDAbstractType" abstract="true">
    <attributeGroup ref="saml:IDNameQualifiers"/>
</complexType>
```

8.1.2.2 Complex type NameIDType

The **NameIDType** complex type is used when an element serves to represent an entity by a string-valued name. It is a more restricted form of identifier than the <BaseID> element and is the type underlying both the <NameID> and <Issuer> elements. In addition to the string content containing the actual identifier, it provides the following optional attributes:

– NameQualifier [Optional]

The security or administrative domain that qualifies the name. This attribute provides a means to federate names from disparate user stores without collision.

– SPNameQualifier [Optional]

Further qualifies a name with the name of a service provider or affiliation of providers. This attribute provides an additional means to federate names on the basis of the relying party or parties.

– Format [Optional]

A URI reference representing the classification of string-based identifier information. See 8.7.3 for the SAML-defined URI references that may be used as the value of the Format attribute and their associated descriptions and processing rules. Unless otherwise specified by an element based on this type, if no Format value is provided, then the value urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified (see 8.7.3.1) is in effect.

When a Format value other than one specified in 8.7.3 is used, the content of an element of this type is to be interpreted according to the definition of that format as provided outside of this Recommendation. If not otherwise indicated by the definition of the format, issues of anonymity, pseudonymity, and the persistence of the identifier with respect to the asserting and relying parties are implementation-specific.

– SPProvidedID [Optional]

A name identifier established by a service provider or affiliation of providers for the entity, if different from the primary name identifier given in the content of the element. This attribute provides a means of integrating the use of SAML with existing identifiers already in use by a service provider. For example, an existing identifier can be "attached" to the entity using the Name Identifier Management protocol defined in 8.2.8.

Additional rules for the content of (or the omission of) these attributes can be defined by elements that make use of this type, and by specific Format definitions. The NameQualifier and SPNameQualifier attributes should be omitted unless the element or format explicitly defines their use and semantics.

The following schema fragment defines the **NameIDType** complex type:

```
<complexType name="NameIDType">
    <simpleContent>
         <extension base="string">
              <attributeGroup ref="saml:IDNameQualifiers"/>
              <attribute name="Format" type="anyURI" use="optional"/>
              <attribute name="SPProvidedID" type="string" use="optional"/>
         </extension>
    </simpleContent>
</complexType>
```

8.1.2.3 Element <NameID>

The `<NameID>` element is of type **NameIDType** (see 8.1.2.2), and is used in various SAML assertion constructs such as the `<Subject>` and `<SubjectConfirmation>` elements, and in various protocol messages (see 8.2).

The following schema fragment defines the `<NameID>` element:

```
<element name="NameID" type="saml:NameIDType"/>
```

8.1.2.4 Element <EncryptedID>

The `<EncryptedID>` element is of type **EncryptedElementType**, and carries the content of an unencrypted identifier element in encrypted fashion, as defined by W3C Encryption. The `<EncryptedID>` element contains the following elements:

– `<xenc:EncryptedData>` [Required]

The encrypted content and associated encryption details, as defined in W3C Encryption. The Type attribute should be present and, if present, must contain a value of `http://www.w3.org/2001/04/xmlenc#Element`. The encrypted content must contain an element that has a type of **NameIDType** or **AssertionType**, or a type that is derived from **BaseIDAbstractType**, **NameIDType**, or **AssertionType**.

– `<xenc:EncryptedKey>` [Zero or More]

Wrapped decryption keys, as defined in W3C Encryption. Each wrapped key should include a Recipient attribute that specifies the entity for whom the key has been encrypted. The value of the Recipient attribute should be the URI identifier of a SAML system entity, as defined by 8.4.

Encrypted identifiers are intended as a privacy protection mechanism when the plain-text value passes through an intermediary. As such, the ciphertext must be unique to any given encryption operation. For more on such issues, see W3C XML Encryption, 6.3.

An entire assertion can be encrypted into this element and used as an identifier. In such a case, the `<Subject>` element of the encrypted assertion supplies the "identifier" of the subject of the enclosing assertion. Hence, if the identifying assertion is invalid, then so is the enclosing assertion.

The following schema fragment defines the `<EncryptedID>` element and its **EncryptedElementType** complex type:

```
<complexType name="EncryptedElementType">
    <sequence>
         <element ref="xenc:EncryptedData"/>
         <element ref="xenc:EncryptedKey" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
</complexType>
<element name="EncryptedID" type="saml:EncryptedElementType"/>
```

8.1.2.5 Element <Issuer>

The `<Issuer>` element, with complex type **NameIDType**, provides information about the issuer of a SAML assertion or protocol message. The element requires the use of a string to carry the issuer's name, but permits various pieces of descriptive data (see 8.1.2.2).

Overriding the usual rule for this element's type, if no `Format` value is provided with this element, then the value `urn:oasis:names:tc:SAML:2.0:nameid-format:entity` is in effect (see 8.1.2.2).

The following schema fragment defines the `<Issuer>` element:

```
<element name="Issuer" type="saml:NameIDType"/>
```

8.1.3 Assertions

The following clauses define the SAML constructs that either contain assertion information or provide a means to refer to an existing assertion.

8.1.3.1 Element <AssertionIDRef>

The `<AssertionIDRef>` element makes a reference to a SAML assertion by its unique identifier. The specific authority who issued the assertion or from whom the assertion can be obtained is not specified as part of the reference. See 8.2.3 for a protocol element that uses such a reference to ask for the corresponding assertion.

The following schema fragment defines the `<AssertionIDRef>` element:

```
<element name="AssertionIDRef" type="NCName"/>
```

8.1.3.2 Element <AssertionURIRef>

The `<AssertionURIRef>` element makes a reference to a SAML assertion by URI reference. The URI reference may be used to retrieve the corresponding assertion in a manner specific to the URI reference. See 7.3 for information on how this element is used in a protocol binding to accomplish this.

The following schema fragment defines the `<AssertionURIRef>` element:

```
<element name="AssertionURIRef" type="anyURI"/>
```

8.1.3.3 Element <Assertion>

The `<Assertion>` element is of the **AssertionType** complex type. This type specifies the basic information that is common to all assertions, including the following elements and attributes:

– `Version` [Required]

The version of this assertion. The identifier for the version of SAML defined in this Recommendation is "2.0". SAML versioning is discussed in 8.3.

– `ID` [Required]

The identifier for this assertion. It is of type **xs:ID**, and must follow the requirements specified in 7.3 for identifier uniqueness.

`IssueInstant` [Required]

The time instant of issue in UTC, as described in 7.3.

– `<Issuer>` [Required]

The SAML authority that is making the claim(s) in the assertion. The issuer should be unambiguous to the intended relying parties.

This Recommendation defines no particular relationship between the entity represented by this element and the signer of the assertion (if any). Any such requirements imposed by a relying party that consumes the assertion or by specific profiles are application-specific.

– `<ds:Signature>` [Optional]

An XML Signature that protects the integrity of and authenticates the issuer of the assertion, as described below and in 8.4.

– `<Subject>` [Optional]

The subject of the statement(s) in the assertion.

– `<Conditions>` [Optional]

Conditions that must be evaluated when assessing the validity of and/or when using the assertion. See 8.1.5 for additional information on how to evaluate conditions.

– `<Advice>` [Optional]

Additional information related to the assertion that assists processing in certain situations but which may be ignored by applications that do not understand the advice or do not wish to make use of it.

Zero or more of the following statement elements:

– `<Statement>`

A statement of a type defined in an extension schema. An **xsi:type** attribute must be used to indicate the actual statement type.

– `<AuthnStatement>`

An authentication statement.

– `<AuthzDecisionStatement>`

An authorization decision statement.

– `<AttributeStatement>`

An attribute statement.

An assertion with no statements must contain a `<Subject>` element. Such an assertion identifies a principal in a manner which can be referenced or confirmed using SAML methods, but asserts no further information associated with that principal.

Otherwise `<Subject>`, if present, identifies the subject of all of the statements in the assertion. If `<Subject>` is omitted, then the statements in the assertion apply to a subject or subjects identified in an application- or profile-specific manner. SAML itself defines no such statements, and an assertion without a subject has no defined meaning in this Recommendation.

Depending on the requirements of particular protocols or profiles, the issuer of a SAML assertion may often need to be authenticated, and integrity protection may often be required. Authentication and message integrity may be provided by mechanisms provided by a protocol binding in use during the delivery of an assertion (see clause 10). The SAML assertion may be signed, which provides both authentication of the issuer and integrity protection.

If such a signature is used, then the `<ds:Signature>` element must be present, and a relying party must verify that the signature is valid (that is, that the assertion has not been tampered with) in accordance with W3C XML Signature. If it is invalid, then the relying party must not rely on the contents of the assertion. If it is valid, then the relying party should evaluate the signature to determine the identity and appropriateness of the issuer and may continue to process the assertion in accordance with this Recommendation and as it deems appropriate (for example, evaluating conditions, advice, following profile-specific rules, and so on).

Whether signed or unsigned, the inclusion of multiple statements within a single assertion is semantically equivalent to a set of assertions containing those statements individually (provided the subject, conditions, etc. are also the same).

The following schema fragment defines the `<Assertion>` element and its **AssertionType** complex type:

```
<element name="Assertion" type="saml:AssertionType"/>
<complexType name="AssertionType">
       <sequence>
              <element ref="saml:Issuer"/>
              <element ref="ds:Signature" minOccurs="0"/>
              <element ref="saml:Subject" minOccurs="0"/>
              <element ref="saml:Conditions" minOccurs="0"/>
              <element ref="saml:Advice" minOccurs="0"/>
              <choice minOccurs="0" maxOccurs="unbounded">
                     <element ref="saml:Statement"/>
                     <element ref="saml:AuthnStatement"/>
                     <element ref="saml:AuthzDecisionStatement"/>
                     <element ref="saml:AttributeStatement"/>
              </choice>
       </sequence>
       <attribute name="Version" type="string" use="required"/>
       <attribute name="ID" type="ID" use="required"/>
       <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>
```

8.1.3.4 Element <EncryptedAssertion>

The `<EncryptedAssertion>` element represents an assertion in encrypted fashion, as defined in W3C Encryption. The `<EncryptedAssertion>` element contains the following elements:

– `<xenc:EncryptedData>` [Required]

The encrypted content and associated encryption details, as defined in W3C Encryption. The Type attribute should be present and, if present, must contain a value of `http://www.w3.org/2001/04/xmlenc#Element`. The encrypted content must contain an element that has a type of or derived from **AssertionType**.

– `<xenc:EncryptedKey>` [Zero or More]

Wrapped decryption keys, as defined in W3C Encryption. Each wrapped key should include a `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of the `Recipient` attribute should be the URI identifier of a SAML system entity as defined by 8.7.

Encrypted assertions are intended as a confidentiality protection mechanism when the plain-text value passes through an intermediary.

The following schema fragment defines the `<EncryptedAssertion>` element:

```
<element name="EncryptedAssertion" type="saml:EncryptedElementType"/>
```

8.1.4 Subjects

This clause defines the SAML constructs used to describe the subject of an assertion. The optional `<Subject>` element specifies the principal that is the subject of all of the (zero or more) statements in the assertion. It contains an identifier, a series of one or more subject confirmations, or both:

– `<BaseID>`, `<NameID>`, or `<EncryptedID>` [Optional]

Identifies the subject.

– `<SubjectConfirmation>` [Zero or More]

Information that allows the subject to be confirmed. If more than one subject confirmation is provided, then satisfying any one of them is sufficient to confirm the subject for the purpose of applying the assertion.

A `<Subject>` element can contain both an identifier and zero or more subject confirmations which a relying party can verify when processing an assertion. If any one of the included subject confirmations are verified, the relying party may treat the entity presenting the assertion as one that the asserting party has associated with the principal identified in the name identifier and associated with the statements in the assertion. This attesting entity and the actual subject may or may not be the same entity.

If there are no subject confirmations included, then any relationship between the presenter of the assertion and the actual subject is unspecified.

A `<Subject>` element should not identify more than one principal.

The following schema fragment defines the `<Subject>` element and its **SubjectType** complex type:

```
<element name="Subject" type="saml:SubjectType"/>
<complexType name="SubjectType">
      <choice>
       <sequence>
             <choice>
                    <element ref="saml:BaseID"/>
                    <element ref="saml:NameID"/>
                    <element ref="saml:EncryptedID"/>
             </choice>
             <element ref="saml:SubjectConfirmation" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
       <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
      </choice>
</complexType>
```

8.1.4.1 Element <SubjectConfirmation>

The `<SubjectConfirmation>` element provides the means for a relying party to verify the correspondence of the subject of the assertion with the party with whom the relying party is communicating. It contains the following attributes and elements:

– `Method` [Required]

A URI reference that identifies a protocol or mechanism to be used to confirm the subject. URI references identifying SAML-defined confirmation methods are defined in clause 11. Additional methods may be added by defining new URIs and profiles or by private agreement.

– `<BaseID>`, `<NameID>`, or `<EncryptedID>` [Optional]

Identifies the entity expected to satisfy the enclosing subject confirmation requirements.

– `<SubjectConfirmationData>` [Optional]

Additional confirmation information to be used by a specific confirmation method. For example, typical content of this element might be a `<ds:KeyInfo>` element as defined in W3C Encryption, which identifies a cryptographic key (see also 8.1.4.3). Particular confirmation methods may define a schema type to describe the elements, attributes, or content that may appear in the `<SubjectConfirmationData>` element.

The following schema fragment defines the `<SubjectConfirmation>` element and its **SubjectConfirmationType** complex type:

```
<element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
<complexType name="SubjectConfirmationType">
      <sequence>
            <choice minOccurs="0">
                  <element ref="saml:BaseID"/>
                  <element ref="saml:NameID"/>
                  <element ref="saml:EncryptedID"/>
            </choice>
            <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
      </sequence>
      <attribute name="Method" type="anyURI" use="required"/>
</complexType>
```

8.1.4.2 Element <SubjectConfirmationData>

The `<SubjectConfirmationData>` element has the **SubjectConfirmationDataType** complex type. It specifies additional data that allows the subject to be confirmed or constrains the circumstances under which the act of subject confirmation can take place. Subject confirmation takes place when a relying party seeks to verify the relationship between an entity presenting the assertion (that is, the attesting entity) and the subject of the assertion's claims. It contains the following optional attributes that can apply to any method:

– `NotBefore` [Optional]

A time instant before which the subject cannot be confirmed. The time value is encoded in UTC, as described in 7.3.

– `NotOnOrAfter` [Optional]

A time instant at which the subject can no longer be confirmed. The time value is encoded in UTC, as described in 7.3.

– `Recipient` [Optional]

A URI specifying the entity or location to which an attesting entity can present the assertion. For example, this attribute might indicate that the assertion must be delivered to a particular network endpoint in order to prevent an intermediary from redirecting it someplace else.

– `InResponseTo` [Optional]

The `ID` of a SAML protocol message in response to which an attesting entity can present the assertion. For example, this attribute might be used to correlate the assertion to a SAML request that resulted in its presentation.

– `Address` [Optional]

The network address/location from which an attesting entity can present the assertion. For example, this attribute might be used to bind the assertion to particular client addresses to prevent an attacker from easily stealing and presenting the assertion from another location. IPv4 addresses should be represented in the usual dotted-decimal format (e.g., "1.2.3.4"). IPv6 addresses should be represented as defined in IETF RFC 3513, 2.2 (e.g., "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

– Arbitrary attributes

This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary namespace-qualified XML attributes to be added to `<SubjectConfirmationData>` constructs without the need for an explicit schema extension. This allows additional fields to be added as needed to supply additional confirmation-related information. SAML extensions must not add local (non-namespace-qualified) XML attributes or XML attributes qualified by a SAML-defined namespace to the **SubjectConfirmationDataType** complex type or a derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

– Arbitrary elements

This complex type uses an `<xs:any>` extension point to allow arbitrary XML elements to be added to `<SubjectConfirmationData>` constructs without the need for an explicit schema extension. This allows additional elements to be added as needed to supply additional confirmation-related information.

Particular confirmation methods and profiles that make use of those methods may require the use of one or more of the attributes defined within this complex type. For examples of how these attributes (and subject confirmation in general) can be used, see clause 13.

The time period specified by the optional `NotBefore` and `NotOnOrAfter` attributes, if present, should fall within the overall assertion validity period as specified by the `<Conditions>` element's `NotBefore` and `NotOnOrAfter` attributes. If both attributes are present, the value for `NotBefore` must be less than (earlier than) the value for `NotOnOrAfter`.

The following schema fragment defines the `<SubjectConfirmationData>` element and its **SubjectConfirmationDataType** complex type:

```
<element name="SubjectConfirmationData"
type="saml:SubjectConfirmationDataType"/>
<complexType name="SubjectConfirmationDataType" mixed="true">
       <complexContent>
              <restriction base="anyType">
                     <sequence>
                            <any namespace="##any" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
                     </sequence>
                     <attribute name="NotBefore" type="dateTime"
use="optional"/>
                     <attribute name="NotOnOrAfter" type="dateTime"
use="optional"/>
                     <attribute name="Recipient" type="anyURI"
use="optional"/>
                     <attribute name="InResponseTo" type="NCName"
use="optional"/>
                     <attribute name="Address" type="string"
use="optional"/>
                     <anyAttribute namespace="##other"
processContents="lax"/>
              </restriction>
       </complexContent>
</complexType>
```

8.1.4.3 Complex type KeyInfoConfirmationDataType

The **KeyInfoConfirmationDataType** complex type constrains a `<SubjectConfirmationData>` element to contain one or more `<ds:KeyInfo>` elements that identify cryptographic keys that are used in some way to authenticate an attesting entity. The particular confirmation method must define the exact mechanism by which the confirmation data can be used. The optional attributes defined by the **SubjectConfirmationDataType** complex type may also appear.

This complex type, or a type derived from it, should be used by any confirmation method that defines its confirmation data in terms of the `<ds:KeyInfo>` element.

In accordance with W3C Encryption, each `<ds:KeyInfo>` element must identify a single cryptographic key. Multiple keys may be identified with separate `<ds:KeyInfo>` elements, such as when a principal uses different keys to confirm itself to different relying parties.

The following schema fragment defines the **KeyInfoConfirmationDataType** complex type:

```
<complexType name="KeyInfoConfirmationDataType" mixed="false">
       <complexContent>
              <restriction base="saml:SubjectConfirmationDataType">
                     <sequence>
                            <element ref="ds:KeyInfo" maxOccurs="unbounded"/>
                     </sequence>
              </restriction>
       </complexContent>
</complexType>
```

8.1.4.4 Example of a key-confirmed <Subject>

To illustrate the way in which the various elements and types fit together, below is an example of a `<Subject>` element containing a name identifier and a subject confirmation based on proof of possession of a key. Here, the use of the **KeyInfoConfirmationDataType** to identify the confirmation data syntax is a `<ds:KeyInfo>` element:

```
<Subject>
       <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">
       scott@example.org
       </NameID>
       <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-
of-key">
              <SubjectConfirmationData
xsi:type="saml:KeyInfoConfirmationDataType">
                     <ds:KeyInfo>
                            <ds:KeyName>Scott's Key</ds:KeyName>
                     </ds:KeyInfo>
              </SubjectConfirmationData>
       </SubjectConfirmation>
</Subject>
```

8.1.5 Conditions

This clause defines the SAML constructs that place constraints on the acceptable use of SAML assertions. The `<Conditions>` element may contain the following elements and attributes:

- `NotBefore` [Optional]

  Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC, as described in 7.3.

- `NotOnOrAfter` [Optional]

  Specifies the time instant at which the assertion has expired. The time value is encoded in UTC, as described in 7.3.

- `<Condition>` [Any Number]

  A condition of a type defined in an extension schema. An `xsi:type` attribute must be used to indicate the actual condition type.

- `<AudienceRestriction>` [Any Number]

  Specifies that the assertion is addressed to a particular audience.

- `<OneTimeUse>` [Optional]

  Specifies that the assertion should be used immediately and must not be retained for future use. Although the schema permits multiple occurrences, there must be at most one instance of this element.

- `<ProxyRestriction>` [Optional]

  Specifies limitations that the asserting party imposes on relying parties that wish to subsequently act as asserting parties themselves and issue assertions of their own on the basis of the information contained in the original assertion. Although the schema permits multiple occurrences, there must be at most one instance of this element.

Because the use of the `xsi:type` attribute would permit an assertion to contain more than one instance of a SAML-defined subtype of **ConditionsType** (such as **OneTimeUseType**), the schema does not explicitly limit the number of times particular conditions may be included. A particular type of condition may define limits on such use, as shown above.

The following schema fragment defines the `<Conditions>` element and its **ConditionsType** complex type:

```
<element name="Conditions" type="saml:ConditionsType"/>
<complexType name="ConditionsType">
       <choice minOccurs="0" maxOccurs="unbounded">
              <element ref="saml:Condition"/>
              <element ref="saml:AudienceRestriction"/>
              <element ref="saml:OneTimeUse"/>
              <element ref="saml:ProxyRestriction"/>
       </choice>
       <attribute name="NotBefore" type="dateTime" use="optional"/>
       <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
</complexType>
```

8.1.5.1 General processing rules

If an assertion contains a `<Conditions>` element, then the validity of the assertion is dependent on the sub-elements and attributes provided, using the following rules in the order shown below.

An assertion that has condition validity status Valid may nonetheless be untrustworthy or invalid for reasons such as not being well-formed or schema-valid, not being issued by a trustworthy SAML authority, or not being authenticated by a trustworthy means.

Some conditions may not directly impact the validity of the containing assertion (they always evaluate to Valid), but may restrict the behaviour of relying parties with respect to the use of the assertion:

– If no sub-elements or attributes are supplied in the `<Conditions>` element, then the assertion is considered to be Valid with respect to condition processing.

– If any sub-element or attribute of the `<Conditions>` element is determined to be invalid, then the assertion is considered to be Invalid.

– If any sub-element or attribute of the `<Conditions>` element cannot be evaluated, or if an element is encountered that is not understood, then the validity of the assertion cannot be determined and is considered to be Indeterminate.

– If all sub-elements and attributes of the `<Conditions>` element are determined to be Valid, then the assertion is considered to be Valid with respect to condition processing.

The first rule that applies terminates condition processing; thus a determination that an assertion is Invalid takes precedence over that of Indeterminate.

An assertion that is determined to be Invalid or Indeterminate must be rejected by a relying party (within whatever context or profile it was being processed), just as if the assertion were malformed or otherwise unusable.

8.1.5.2 Attributes NotBefore and NotOnOrAfter

The `NotBefore` and `NotOnOrAfter` attributes specify time limits on the validity of the assertion within the context of its profile(s) of use. They do not guarantee that the statements in the assertion will be correct or accurate throughout the validity period.

The `NotBefore` attribute specifies the time instant at which the validity interval begins. The `NotOnOrAfter` attribute specifies the time instant at which the validity interval has ended.

If the value for either `NotBefore` or `NotOnOrAfter` is omitted, then it is considered unspecified. If the `NotBefore` attribute is unspecified (and if all other conditions that are supplied evaluate to Valid), then the assertion is Valid with respect to conditions at any time before the time instant specified by the `NotOnOrAfter` attribute. If the `NotOnOrAfter` attribute is unspecified (and if all other conditions that are supplied evaluate to Valid), the assertion is Valid with respect to conditions from the time instant specified by the `NotBefore` attribute with no expiry. If neither attribute is specified (and if any other conditions that are supplied evaluate to Valid), the assertion is Valid with respect to conditions at any time.

If both attributes are present, the value for `NotBefore` must be less than (earlier than) the value for `NotOnOrAfter`.

8.1.5.3 Element <Condition>

The `<Condition>` element serves as an extension point for new conditions. Its **ConditionAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

The following schema fragment defines the `<Condition>` element and its **ConditionAbstractType** complex type:

```
<element name="Condition" type="saml:ConditionAbstractType"/>
<complexType name="ConditionAbstractType" abstract="true"/>
```

8.1.5.4 Elements <AudienceRestriction> and <Audience>

The `<AudienceRestriction>` element specifies that the assertion is addressed to one or more specific audiences identified by `<Audience>` elements. Although a SAML relying party that is outside the audiences specified is capable of drawing conclusions from an assertion, the SAML asserting party explicitly makes no representation as to accuracy or trustworthiness to such a party. It contains the following element:

– `<Audience>`

  A URI reference that identifies an intended audience. The URI reference may identify a document that describes the terms and conditions of audience membership. It may also contain the unique identifier URI from a SAML name identifier that describes a system entity.

The audience restriction condition evaluates to Valid if, and only if, the SAML relying party is a member of one or more of the audiences specified.

The SAML asserting party cannot prevent a party to whom the assertion is disclosed from taking action on the basis of the information provided. However, the `<AudienceRestriction>` element allows the SAML asserting party to state explicitly that no warranty is provided to such a party in a machine- and human-readable form. While there can be no guarantee that a court would uphold such a warranty exclusion in every circumstance, the probability of upholding the warranty exclusion is considerably improved.

Multiple `<AudienceRestriction>` elements may be included in a single assertion, and each must be evaluated independently. The effect of this requirement and the preceding definition is that within a given condition, the audiences form a disjunction (an "OR") while multiple conditions form a conjunction (an "AND").

The following schema fragment defines the `<AudienceRestriction>` element and its **AudienceRestrictionType** complex type:

```
<element name="AudienceRestriction"
       type="saml:AudienceRestrictionType"/>
<complexType name="AudienceRestrictionType">
       <complexContent>
              <extension base="saml:ConditionAbstractType">
                     <sequence>
                            <element ref="saml:Audience"
maxOccurs="unbounded"/>
                     </sequence>
              </extension>
       </complexContent>
</complexType>
<element name="Audience" type="anyURI"/>
```

8.1.5.5 Element <OneTimeUse>

In general, relying parties may choose to retain assertions, or the information they contain in some other form, for reuse. The `<OneTimeUse>` condition element allows an authority to indicate that the information in the assertion is likely to change very soon and fresh information should be obtained for each use. An example would be an assertion containing an `<AuthzDecisionStatement>` which was the result of a policy which specified access control which was a function of the time of day.

If system clocks in a distributed environment could be precisely synchronized, then this requirement could be met by careful use of the validity interval. However, since some clock skew between systems will always be present and will be combined with possible transmission delays, there is no convenient way for the issuer to appropriately limit the lifetime of an assertion without running a substantial risk that it will already have expired before it arrives.

The `<OneTimeUse>` element indicates that the assertion should be used immediately by the relying party and must not be retained for future use. Relying parties are always free to request a fresh assertion for every use. However, implementations that choose to retain assertions for future use must observe the `<OneTimeUse>` element. This condition is independent from the `NotBefore` and `NotOnOrAfter` condition information.

To support the single use constraint, a relying party should maintain a cache of the assertions it has processed containing such a condition. Whenever an assertion with this condition is processed, the cache should be checked to ensure that the same assertion has not been previously received and processed by the relying party.

A SAML authority must not include more than one `<OneTimeUse>` element within a `<Conditions>` element of an assertion.

For the purposes of determining the validity of the `<Conditions>` element, the `<OneTimeUse>` is considered to always be valid. That is, this condition does not affect validity but is a condition on use.

The following schema fragment defines the `<OneTimeUse>` element and its **OneTimeUseType** complex type:

```
<element name="OneTimeUse" type="saml:OneTimeUseType"/>
<complexType name="OneTimeUseType">
       <complexContent>
              <extension base="saml:ConditionAbstractType"/>
</complexContent>
</complexType>
```

8.1.5.6 Element <ProxyRestriction>

Specifies limitations that the asserting party imposes on relying parties that in turn wish to act as asserting parties and issue subsequent assertions of their own on the basis of the information contained in the original assertion. A relying party acting as an asserting party must not issue an assertion that itself violates the restrictions specified in this condition on the basis of an assertion containing such a condition.

The `<ProxyRestriction>` element contains the following elements and attributes:

– `Count` [Optional]

  Specifies the maximum number of indirections that the asserting party permits to exist between this assertion and an assertion which has ultimately been issued on the basis of it.

– `<Audience>` [Zero or More]

  Specifies the set of audiences to whom the asserting party permits new assertions to be issued on the basis of this assertion.

A `Count` value of zero indicates that a relying party must not issue an assertion to another relying party on the basis of this assertion. If greater than zero, any assertions so issued must themselves contain a `<ProxyRestriction>` element with a `Count` value of at most one less than this value.

If no `<Audience>` elements are specified, then no audience restrictions are imposed on the relying parties to whom subsequent assertions can be issued. Otherwise, any assertions so issued must themselves contain an `<AudienceRestriction>` element with at least one of the `<Audience>` elements present in the previous `<ProxyRestriction>` element, and no `<Audience>` elements present that were not in the previous `<ProxyRestriction>` element.

A SAML authority must not include more than one `<ProxyRestriction>` element within a `<Conditions>` element of an assertion.

For the purposes of determining the validity of the `<Conditions>` element, the `<ProxyRestriction>` condition is considered to always be valid. That is, this condition does not affect validity but is a condition on use.

The following schema fragment defines the `<ProxyRestriction>` element and its **ProxyRestrictionType** complex type:

```
<element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
<complexType name="ProxyRestrictionType">
       <complexContent>
              <extension base="saml:ConditionAbstractType">
                     <sequence>
                            <element ref="saml:Audience" minOccurs="0"
maxOccurs="unbounded"/>
                     </sequence>
                     <attribute name="Count" type="nonNegativeInteger"
use="optional"/>
```

```
                </extension>
        </complexContent>
</complexType>
```

8.1.6 Advice

This clause defines the SAML constructs that contain additional information about an assertion that an asserting party wishes to provide to a relying party.

The `<Advice>` element contains any additional information that the SAML authority wishes to provide. This information may be ignored by applications without affecting either the semantics or the validity of the assertion.

The `<Advice>` element contains a mixture of zero or more `<Assertion>`, `<EncryptedAssertion>`, `<AssertionIDRef>`, and `<AssertionURIRef>` elements, and namespace-qualified elements in other non-SAML namespaces.

Following are some potential uses of the `<Advice>` element:

- Include evidence supporting the assertion claims to be cited, either directly (through incorporating the claims) or indirectly (by reference to the supporting assertions).
- State a proof of the assertion claims.
- Specify the timing and distribution points for updates to the assertion.

The following schema fragment defines the `<Advice>` element and its **AdviceType** complex type:

```
<element name="Advice" type="saml:AdviceType"/>
<complexType name="AdviceType">
        <choice minOccurs="0" maxOccurs="unbounded">
                <element ref="saml:AssertionIDRef"/>
                <element ref="saml:AssertionURIRef"/>
                <element ref="saml:Assertion"/>
                <element ref="saml:EncryptedAssertion"/>
                <any namespace="##other" processContents="lax"/>
        </choice>
</complexType>
```

8.1.7 Statements

All SAML defined statements are associated with a subject. SAML assertions are usually made about a **subject**, represented by the `<Subject>` element. However, the `<Subject>` element is optional, and other specifications and profiles may utilize the SAML assertion structure to make similar statements without specifying a subject, or possibly specifying the subject in an alternate way. The following subclauses define the SAML constructs that contain statement information.

8.1.7.1 Element <Statement>

The `<Statement>` element is an extension point that allows other assertion-based applications to reuse the SAML assertion framework. SAML itself derives its core statements from this extension point. Its **StatementAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

The following schema fragment defines the `<Statement>` element and its **StatementAbstractType** complex type:

```
<element name="Statement" type="saml:StatementAbstractType"/>
<complexType name="StatementAbstractType" abstract="true"/>
```

8.1.7.2 Element <AuthnStatement>

The `<AuthnStatement>` element describes a statement by the SAML authority asserting that the assertion subject was authenticated by a particular means at a particular time. Assertions containing `<AuthnStatement>` elements must contain a `<Subject>` element.

It is of type **AuthnStatementType**, which extends **StatementAbstractType** with the addition of the following elements and attributes:

NOTE – The `<AuthorityBinding>` element and its corresponding type were removed from `<AuthnStatement>` for V2.0 of SAML.

– `AuthnInstant` [Required]

Specifies the time at which the authentication took place. The time value is encoded in UTC, as described in 7.3.

– SessionIndex [Optional]

Specifies the index of a particular session between the principal identified by the subject and the authenticating authority.

– SessionNotOnOrAfter [Optional]

Specifies a time instant at which the session between the principal identified by the subject and the SAML authority issuing this statement must be considered ended. The time value is encoded in UTC, as described in 7.3. There is no required relationship between this attribute and a NotOnOrAfter condition attribute that may be present in the assertion.

– <SubjectLocality> [Optional]

Specifies the DNS domain name and IP address for the system from which the assertion subject was apparently authenticated.

– <AuthnContext> [Required]

The context used by the authenticating authority up to and including the authentication event that yielded this statement. Contains an authentication context class reference, an authentication context declaration or declaration reference, or both. See clause 12 (Authentication Context) for a full description of authentication context information.

In general, any string value may be used as a SessionIndex value. However, when privacy is a consideration, care must be taken to ensure that the SessionIndex value does not invalidate other privacy mechanisms. Accordingly, the value should not be usable to correlate activity by a principal across different session participants. Two solutions that achieve this goal are provided below and are recommended:

- Use small positive integers (or reoccurring constants in a list) for the SessionIndex. The SAML authority should choose the range of values such that the cardinality of any one integer will be sufficiently high to prevent a particular principal's actions from being correlated across multiple session participants. The SAML authority should choose values for SessionIndex randomly from within this range (except when required to ensure unique values for subsequent statements given to the same session participant but as part of a distinct session).
- Use the enclosing assertion's ID value in the SessionIndex.

The following schema fragment defines the <AuthnStatement> element and its **AuthnStatementType** complex type:

```
<element name="AuthnStatement" type="saml:AuthnStatementType"/>
<complexType name="AuthnStatementType">
       <complexContent>
              <extension base="saml:StatementAbstractType">
                     <sequence>
                            <element ref="saml:SubjectLocality"
minOccurs="0"/>
                            <element ref="saml:AuthnContext"/>
                     </sequence>
                     <attribute name="AuthnInstant" type="dateTime"
use="required"/>
                     <attribute name="SessionIndex" type="string"
use="optional"/>
                     <attribute name="SessionNotOnOrAfter" type="dateTime"
use="optional"/>
              </extension>
       </complexContent>
</complexType>
```

8.1.7.2.1 Element <SubjectLocality>

The <SubjectLocality> element specifies the DNS domain name and IP address for the system from which the assertion subject was authenticated. It has the following attributes:

– Address [Optional]

The network address of the system from which the principal identified by the subject was authenticated. IPv4 addresses should be represented in dotted-decimal format (e.g., "1.2.3.4"). IPv6 addresses should be represented as defined in IETF RFC 3513, 2.2 (e.g., "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

– `DNSName` [Optional]

The DNS name of the system from which the principal identified by the subject was authenticated.

This element is entirely advisory, since both of these fields are quite easily "spoofed", but may be useful information in some applications.

The following schema fragment defines the `<SubjectLocality>` element and its **SubjectLocalityType** complex type:

```
<element name="SubjectLocality" type="saml:SubjectLocalityType"/>
<complexType name="SubjectLocalityType">
       <attribute name="Address" type="string" use="optional"/>
       <attribute name="DNSName" type="string" use="optional"/>
</complexType>
```

8.1.7.2.2 Element <AuthnContext>

The `<AuthnContext>` element specifies the context of an authentication event. The element can contain an authentication context class reference, an authentication context declaration or declaration reference, or both. Its complex **AuthnContextType** has the following elements:

– `<AuthnContextClassRef>` [Optional]

A URI reference identifying an authentication context class that describes the authentication context declaration that follows.

– `<AuthnContextDecl>` or `<AuthnContextDeclRef>` [Optional]

Either an authentication context declaration provided by value, or a URI reference that identifies such a declaration. The URI reference may directly resolve into an XML document containing the referenced declaration.

– `<AuthenticatingAuthority>` [Zero or More]

Zero or more unique identifiers of authentication authorities that were involved in the authentication of the principal (not including the assertion issuer, who is presumed to have been involved without being explicitly named here).

See clause 12 for a full description of authentication context information.

The following schema fragment defines the `<AuthnContext>` element and its **AuthnContextType** complex type:

```
<element name="AuthnContext" type="saml:AuthnContextType"/>
<complexType name="AuthnContextType">
       <sequence>
              <choice>
                     <sequence>
                            <element ref="saml:AuthnContextClassRef"/>
                            <choice minOccurs="0">
                                   <element ref="saml:AuthnContextDecl"/>
                                   <element ref="saml:AuthnContextDeclRef"/>
                            </choice>
                     </sequence>
                     <choice>
                            <element ref="saml:AuthnContextDecl"/>
                            <element ref="saml:AuthnContextDeclRef"/>
                     </choice>
              </choice>
              <element ref="saml:AuthenticatingAuthority" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
</complexType>
<element name="AuthnContextClassRef" type="anyURI"/>
<element name="AuthnContextDeclRef" type="anyURI"/>
<element name="AuthnContextDecl" type="anyType"/>
<element name="AuthenticatingAuthority" type="anyURI"/>
```

8.1.7.3 Element <AttributeStatement>

The `<AttributeStatement>` element describes a statement by the SAML authority asserting that the assertion subject is associated with the specified attributes. Assertions containing `<AttributeStatement>` elements must contain a `<Subject>` element.

It is of type **AttributeStatementType**, which extends **StatementAbstractType** with the addition of the following elements:

– `<Attribute>` or `<EncryptedAttribute>` [One or More]

The `<Attribute>` element specifies an attribute of the assertion subject. An encrypted SAML attribute may be included with the `<EncryptedAttribute>` element.

The following schema fragment defines the `<AttributeStatement>` element and its **AttributeStatementType** complex type:

```
<element name="AttributeStatement" type="saml:AttributeStatementType"/>
<complexType name="AttributeStatementType">
      <complexContent>
            <extension base="saml:StatementAbstractType">
                  <choice maxOccurs="unbounded">
                        <element ref="saml:Attribute"/>
                        <element ref="saml:EncryptedAttribute"/>
                  </choice>
            </extension>
      </complexContent>
</complexType>
```

8.1.7.3.1 Element <Attribute>

The `<Attribute>` element identifies an attribute by name and optionally includes its value(s). It has the **AttributeType** complex type. It is used within an attribute statement to express particular attributes and values associated with an assertion subject, as described in the previous clause. It is also used in an attribute query to request that the values of specific SAML attributes be returned. The `<Attribute>` element contains the following XML attributes:

– `Name` [Required]

The name of the attribute.

– `NameFormat` [Optional]

A URI reference representing the classification of the attribute name for purposes of interpreting the name. See 8.7.2 for some URI references that may be used as the value of the `NameFormat` attribute and their associated descriptions and processing rules. If no `NameFormat` value is provided, the identifier `urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified` is in effect.

– `FriendlyName` [Optional]

A string that provides a more human-readable form of the attribute's name, which may be useful in cases in which the actual `Name` is complex or opaque, such as an OID (as defined in ITU-T Rec. X.660) or a UUID (as defined in ITU-T Rec. X.667). This attribute's value must not be used as a basis for formally identifying SAML attributes.

– Arbitrary attributes

This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary XML attributes to be added to `<Attribute>` constructs without the need for an explicit schema extension. This allows additional fields to be added as needed to supply additional parameters to be used, for example, in an attribute query. SAML extensions must not add local (non-namespace-qualified) XML attributes or XML attributes qualified by a SAML-defined namespace to the **AttributeType** complex type or a derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

– `<AttributeValue>` [Any Number]

Contains a value of the attribute. If an attribute contains more than one discrete value, it is recommended that each value appear in its own `<AttributeValue>` element. If more than one `<AttributeValue>` element is supplied for an attribute, and any of the elements have a datatype assigned through `xsi:type`, then all of the `<AttributeValue>` elements must have the identical datatype assigned.

The meaning of an <Attribute> element that contains no <AttributeValue> elements depends on its context. Within an <AttributeStatement>, if the SAML attribute exists but has no values, then the <AttributeValue> element must be omitted. Within a <samlp:AttributeQuery>, the absence of values indicates that the requester is interested in any or all of the named attribute's values (see also 8.2).

Any other uses of the <Attribute> element by profiles or other specifications must define the semantics of specifying or omitting <AttributeValue> elements.

The following schema fragment defines the <Attribute> element and its **AttributeType** complex type:

```
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
       <sequence>
              <element ref="saml:AttributeValue" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
       <attribute name="Name" type="string" use="required"/>
       <attribute name="NameFormat" type="anyURI" use="optional"/>
       <attribute name="FriendlyName" type="string" use="optional"/>
       <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

The <AttributeValue> element supplies the value of a specified SAML attribute. It is of the **xs:anyType** type, which allows any well-formed XML to appear as the content of the element.

If the data content of an <AttributeValue> element is of an XML Schema simple type (such as **xs:integer** or **xs:string**), the datatype may be declared explicitly by means of an xsi:type declaration in the <AttributeValue> element. If the attribute value contains structured data, the necessary data elements may be defined in an extension schema.

NOTE – Specifying a datatype other than an XML Schema simple type on <AttributeValue> using xsi:type will require the presence of the extension schema that defines the datatype in order for schema processing to proceed.

If a SAML attribute includes an empty value, such as the empty string, the corresponding <AttributeValue> element must be empty (generally this is serialized as <AttributeValue/>). This overrides the requirement in 7.1 that string values in SAML content contain at least one non-whitespace character.

If a SAML attribute includes a "null" value, the corresponding <AttributeValue> element must be empty and must contain the reserved xsi:nil XML attribute with a value of "true" or "1".

The following schema fragment defines the <AttributeValue> element:

```
<element name="AttributeValue" type="anyType" nillable="true"/>
```

8.1.7.3.2 Element <EncryptedAttribute>

The <EncryptedAttribute> element represents a SAML attribute in encrypted fashion, as defined in W3C Encryption. The <EncryptedAttribute> element contains the following elements:

– <xenc:EncryptedData> [Required]

The encrypted content and associated encryption details, as defined in W3C Encryption. The Type attribute should be present and, if present, must contain a value of http://www.w3.org/2001/04/xmlenc#Element. The encrypted content must contain an element that has a type of or derived from **AttributeType**.

– <xenc:EncryptedKey> [Zero or More]

Wrapped decryption keys, as defined in W3C Encryption. Each wrapped key should include a Recipient attribute that specifies the entity for whom the key has been encrypted. The value of the Recipient attribute should be the URI identifier of a system entity with a SAML name identifier, as defined by 8.7.

Encrypted attributes are intended as a confidentiality protection when the plain-text value passes through an intermediary.

The following schema fragment defines the <EncryptedAttribute> element:

```
<element name="EncryptedAttribute" type="saml:EncryptedElementType"/>
```

8.1.7.4 Element <AuthzDecisionStatement>

The `<AuthzDecisionStatement>` element describes a statement by the SAML authority asserting that a request for access by the assertion subject to the specified resource has resulted in the specified authorization decision on the basis of some optionally specified evidence. Assertions containing `<AuthzDecisionStatement>` elements must contain a `<Subject>` element.

The resource is identified by means of a URI reference. In order for the assertion to be interpreted correctly and securely, the SAML authority and SAML relying party must interpret each URI reference in a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing URI references are to be found in IETF RFC 2396, clause 6.

To avoid ambiguity resulting from variations in URI encoding, SAML system entities should employ the URI normalized form wherever possible as follows:

- SAML authorities should encode all resource URI references in normalized form.
- Relying parties should convert resource URI references to normalized form prior to processing.

Inconsistent URI reference interpretation can also result from differences between the URI reference syntax and the semantics of an underlying file system. Particular care is required if URI references are employed to specify an access control policy language. The following security conditions should be satisfied by the system which employs SAML assertions:

- Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive, a requester should not be able to gain access to a denied resource by changing the case of a part of the resource URI reference.
- Many file systems support mechanisms such as logical paths and symbolic links, which allow users to establish logical equivalences between file system entries. A requester should not be able to gain access to a denied resource by creating such an equivalence.

The `<AuthzDecisionStatement>` element is of type **AuthzDecisionStatementType**, which extends **StatementAbstractType** with the addition of the following elements and attributes:

– `Resource` [Required]

A URI reference identifying the resource to which access authorization is sought. This attribute may have the value of the empty URI reference (""), and the meaning is defined to be "the start of the current document", as specified in IETF RFC 2396, 4.2.

– `Decision` [Required]

The decision rendered by the SAML authority with respect to the specified resource. The value is of the **DecisionType** simple type.

– `<Action>` [One or more]

The set of actions authorized to be performed on the specified resource.

– `<Evidence>` [Optional]

A set of assertions that the SAML authority relied on in making the decision.

The following schema fragment defines the `<AuthzDecisionStatement>` element and its **AuthzDecisionStatementType** complex type:

```
<element name="AuthzDecisionStatement"
 type="saml:AuthzDecisionStatementType"/>
<complexType name="AuthzDecisionStatementType">
 <complexContent>
  <extension base="saml:StatementAbstractType">
         <sequence>
                <element ref="saml:Action" maxOccurs="unbounded"/>
                <element ref="saml:Evidence" minOccurs="0"/>
         </sequence>
         <attribute name="Resource" type="anyURI" use="required"/>
         <attribute name="Decision" type="saml:DecisionType" use="required"/>
  </extension>
 </complexContent>
</complexType>
```

8.1.7.4.1 Simple type DecisionType

The **DecisionType** simple type defines the possible values to be reported as the status of an authorization decision statement.

– `Permit`

  The specified action is permitted.

– `Deny`

  The specified action is denied.

– `Indeterminate`

  The SAML authority cannot determine whether the specified action is permitted or denied.

The `Indeterminate` decision value is used in situations where the SAML authority requires the ability to provide an affirmative statement but where it is not able to issue a decision. Additional information as to the reason for the refusal or inability to provide a decision may be returned as `<StatusDetail>` elements in the enclosing `<Response>`.

The following schema fragment defines the **DecisionType** simple type:

```
<simpleType name="DecisionType">
      <restriction base="string">
            <enumeration value="Permit"/>
            <enumeration value="Deny"/>
            <enumeration value="Indeterminate"/>
      </restriction>
</simpleType>
```

8.1.7.4.2 Element <Action>

The `<Action>` element specifies an action on the specified resource for which permission is sought. Its string-data content provides the label for an action sought to be performed on the specified resource, and it has the following attribute:

– `Namespace` [Optional]

  A URI reference representing the namespace in which the name of the specified action is to be interpreted. If this element is absent, the namespace `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation` specified in 8.7 is in effect.

  NOTE (informative) – PE 36 (see OASIS PE:2006) suggests to replace the above text with:

    `Namespace` [Required]

    A URI reference representing the namespace in which the name of the specified action is to be interpreted.

The following schema fragment defines the `<Action>` element and its **ActionType** complex type:

```
<element name="Action" type="saml:ActionType"/>
<complexType name="ActionType">
 <simpleContent>
       <extension base="string">
             <attribute name="Namespace" type="anyURI" use="required"/>
       </extension>
 </simpleContent>
</complexType>
```

8.1.7.4.3 Element <Evidence>

The `<Evidence>` element contains one or more assertions or assertion references that the SAML authority relied on in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one or more of the following elements:

– `<AssertionIDRef>` [Any number]

  Specifies an assertion by reference to the value of the assertion's `ID` attribute.

– `<AssertionURIRef>` [Any number]

  Specifies an assertion by means of a URI reference.

– `<Assertion>` [Any number]

Specifies an assertion by value.

– `<EncryptedAssertion>` [Any number]

Specifies an encrypted assertion by value.

Providing an assertion as evidence may affect the reliance agreement between the SAML relying party and the SAML authority making the authorization decision. For example, in the case that the SAML relying party presented an assertion to the SAML authority in a request, the SAML authority may use that assertion as evidence in making its authorization decision without endorsing the `<Evidence>` element's assertion as valid either to the relying party or any other third party.

The following schema fragment defines the `<Evidence>` element and its **EvidenceType** complex type:

```
<element name="Evidence" type="saml:EvidenceType"/>
<complexType name="EvidenceType">
      <choice maxOccurs="unbounded">
            <element ref="saml:AssertionIDRef"/>
            <element ref="saml:AssertionURIRef"/>
            <element ref="saml:Assertion"/>
            <element ref="saml:EncryptedAssertion"/>
      </choice>
</complexType>
```

8.2 SAML protocols

SAML protocol messages can be generated and exchanged using a variety of protocols. The SAML bindings in clause 10 describe specific means of transporting protocol messages using existing widely deployed transport protocols. The SAML profile in clause 11 describes a number of applications of the protocols defined in this clause together with additional processing rules, restrictions, and requirements that facilitate interoperability.

Specific SAML request and response messages derive from common types. The requester sends an element derived from **RequestAbstractType** to a SAML responder, and the responder generates an element adhering to or deriving from **StatusResponseType**, as shown in Figure 8-1.

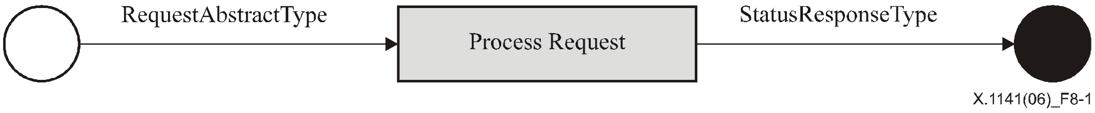

Figure 8-1/X.1141 – SAML request-response protocol

In certain cases, when permitted by profiles, a SAML response may be generated and sent without the responder having received a corresponding request.

The protocols defined by SAML achieve the following actions:

- Returning one or more requested assertions. This can occur in response to either a direct request for specific assertions or a query for assertions that meet particular criteria.
- Performing authentication on request and returning the corresponding assertion.
- Registering a name identifier or terminating a name registration on request.
- Retrieving a protocol message that has been requested by means of an artifact.
- Performing a near-simultaneous logout of a collection of related sessions ("single logout") on request.
- Providing a name identifier mapping on request.

Throughout this clause, text descriptions of elements and types in the SAML protocol namespace are not shown with the conventional namespace prefix `samlp:`. For clarity, text descriptions of elements and types in the SAML assertion namespace are indicated with the conventional namespace prefix `saml:`.

8.2.1 Schema header and namespace Declarations

The following schema fragment defines the XML namespaces and other header information for the protocol schema:

```
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="saml-schema-assertion-2.0.xsd"/>
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
20020212/xmldsig-core-schema.xsd"/>
    <annotation>
        <documentation>
            Document identifier: saml-schema-protocol-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
            V1.0 (November, 2002):
              Initial Standard Schema.
            V1.1 (September, 2003):
              Updates within the same V1.0 namespace.
            V2.0 (March, 2005):
              New protocol schema based in a SAML V2.0 namespace.
     </documentation>
    </annotation>
…
</schema>
```

8.2.2 Requests and responses

The following subclauses define the SAML constructs and basic requirements that underlie all of the request and response messages used in SAML protocols.

8.2.2.1 Complex type RequestAbstractType

All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type. This type defines common attributes and elements that are associated with all SAML requests:

NOTE – The `<RespondWith>` element has been removed from **`RequestAbstractType`** for V2.0 of SAML.

– `ID` [Required]

An identifier for the request. It is of type **xs:ID** and must follow the requirements specified in 7.4 for identifier uniqueness. The values of the `ID` attribute in a request and the `InResponseTo` attribute in the corresponding response must match.

– `Version` [Required]

The version of this request. The identifier for the version of SAML defined in this Recommendation is "2.0".

– `IssueInstant` [Required]

The time instant of issue of the request. The time value is encoded in UTC, as described in 7.3.

– `Destination` [Optional]

A URI reference indicating the address to which this request has been sent. This is useful to prevent malicious forwarding of requests to unintended recipients, a protection that is required by some protocol bindings. If it is present, the actual recipient must check that the URI reference identifies the location at which the message was received. If it does not, the request must be discarded. Some protocol bindings may require the use of this attribute (see clause 10).

– `Consent` [Optional]

Indicates whether or not (and under what conditions) consent has been obtained from a principal in the sending of this request. See 8.7.4 for some URI references that may be used as the value of the `Consent` attribute and their associated descriptions. If no `Consent` value is provided, the identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` is in effect.

– `<saml:Issuer>` [Optional]

Identifies the entity that generated the request message.

– `<ds:Signature>` [Optional]

An XML Signature that authenticates the requester and provides message integrity, as described below and in 8.4.

– `<Extensions>` [Optional]

This extension point contains optional protocol message extension elements that are agreed on between the communicating parties. No extension schema is required in order to make use of this extension point, and even if one is provided, the lax validation setting does not impose a requirement for the extension to be valid. SAML extension elements must be namespace-qualified in a non-SAML-defined namespace.

Depending on the requirements of particular protocols or profiles, a SAML requester may often need to authenticate itself, and message integrity may often be required. Authentication and message integrity may be provided by mechanisms provided by the protocol binding (see clause 10). The SAML request may be signed, which provides both authentication of the requester and message integrity.

If such a signature is used, then the `<ds:Signature>` element must be present, and the SAML responder must verify that the signature is valid (that is, that the message has not been tampered with) in accordance with W3C Signature. If it is invalid, then the responder must not rely on the contents of the request and should respond with an error. If it is valid, then the responder should evaluate the signature to determine the identity and appropriateness of the signer and may continue to process the request or respond with an error (if the request is invalid for some other reason).

If a `Consent` attribute is included and the value indicates that some form of principal consent has been obtained, then the request should be signed.

If a SAML responder deems a request to be invalid according to SAML syntax or processing rules, then if it responds, it must return a SAML response message with a `<StatusCode>` element with the value `urn:oasis:names:tc:SAML:2.0:status:Requester`. In some cases, for example during a suspected denial-of-service attack, not responding at all may be warranted.

The following schema fragment defines the **RequestAbstractType** complex type:

```
<complexType name="RequestAbstractType" abstract="true">
	<sequence>
		<element ref="saml:Issuer" minOccurs="0"/>
		<element ref="ds:Signature" minOccurs="0"/>
		<element ref="samlp:Extensions" minOccurs="0"/>
	</sequence>
	<attribute name="ID" type="ID" use="required"/>
	<attribute name="Version" type="string" use="required"/>
	<attribute name="IssueInstant" type="dateTime" use="required"/>
	<attribute name="Destination" type="anyURI" use="optional"/>
	<attribute name="Consent" type="anyURI" use="optional"/>
</complexType>
<element name="Extensions" type="samlp:ExtensionsType"/>
<complexType name="ExtensionsType">
	<sequence>
		<any namespace="##other" processContents="lax"
maxOccurs="unbounded"/>
	</sequence>
</complexType>
```

8.2.2.2 Complex type StatusResponseType

All SAML responses are of types that are derived from the **StatusResponseType** complex type. This type defines common attributes and elements that are associated with all SAML responses:

– `ID` [Required]

An identifier for the response. It is of type **xs:ID**, and must follow the requirements specified in 7.4 for identifier uniqueness.

– `InResponseTo` [Optional]

A reference to the identifier of the request to which the response corresponds, if any. If the response is not generated in response to a request, or if the `ID` attribute value of a request cannot be determined (for example, the request is malformed), then this attribute must not be present. Otherwise, it must be present and its value must match the value of the corresponding request's `ID` attribute.

– `Version` [Required]

The version of this response. The identifier for the version of SAML defined in this Recommendation is "2.0".

– `IssueInstant` [Required]

The time instant of issue of the response. The time value is encoded in UTC, as described in 7.3.

– `Destination` [Optional]

A URI reference indicating the address to which this response has been sent. This is useful to prevent malicious forwarding of responses to unintended recipients, a protection that is required by some protocol bindings. If it is present, the actual recipient must check that the URI reference identifies the location at which the message was received. If it does not, the response must be discarded. Some protocol bindings may require the use of this attribute (see clause 10).

– `Consent` [Optional]

Indicates whether or not (and under what conditions) consent has been obtained from a principal in the sending of this response. See 8.7.4 for some URI references that may be used as the value of the `Consent` attribute and their associated descriptions. If no `Consent` value is provided, the identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see 8.7.4) is in effect.

– `<saml:Issuer>` [Optional]

Identifies the entity that generated the response message. (For more information on this element, see 8.1.2.5).

– `<ds:Signature>` [Optional]

An XML Signature that authenticates the responder and provides message integrity, as described below and in 8.4.

– `<Extensions>` [Optional]

This extension point contains optional protocol message extension elements that are agreed on between the communicating parties. No extension schema is required in order to make use of this extension point, and even if one is provided, the lax validation setting does not impose a requirement for the extension to be valid. SAML extension elements must be namespace-qualified in a non-SAML-defined namespace.

– `<Status>` [Required]

A code representing the status of the corresponding request.

Depending on the requirements of particular protocols or profiles, a SAML responder may often need to authenticate itself, and message integrity may often be required. Authentication and message integrity may be provided by mechanisms provided by the protocol binding. The SAML response may be signed, which provides both authentication of the responder and message integrity.

If such a signature is used, then the `<ds:Signature>` element must be present, and the SAML requester receiving the response must verify that the signature is valid (that is, that the message has not been tampered with), in accordance with W3C XML Signature. If it is invalid, then the requester must not rely on the contents of the response and should treat it as an error. If it is valid, then the requester should evaluate the signature to determine the identity and appropriateness of the signer and may continue to process the response as it deems appropriate.

If a `Consent` attribute is included and the value indicates that some form of principal consent has been obtained, then the response should be signed.

The following schema fragment defines the **StatusResponseType** complex type:

```
<complexType name="StatusResponseType">
      <sequence>
            <element ref="saml:Issuer" minOccurs="0"/>
            <element ref="ds:Signature" minOccurs="0"/>
            <element ref="samlp:Extensions" minOccurs="0"/>
            <element ref="samlp:Status"/>
      </sequence>
      <attribute name="ID" type="ID" use="required"/>
      <attribute name="InResponseTo" type="NCName" use="optional"/>
      <attribute name="Version" type="string" use="required"/>
      <attribute name="IssueInstant" type="dateTime" use="required"/>
      <attribute name="Destination" type="anyURI" use="optional"/>
      <attribute name="Consent" type="anyURI" use="optional"/>
</complexType>
```

1) Element <Status>

The `<Status>` element contains the following elements:

– `<StatusCode>` [Required]

  A code representing the status of the activity carried out in response to the corresponding request.

– `<StatusMessage>` [Optional]

  A message which may be returned to an operator.

– `<StatusDetail>` [Optional]

  Additional information concerning the status of the request.

The following schema fragment defines the `<Status>` element and its **StatusType** complex type:

```
<element name="Status" type="samlp:StatusType"/>
<complexType name="StatusType">
   <sequence>
      <element ref="samlp:StatusCode"/>
      <element ref="samlp:StatusMessage" minOccurs="0"/>
      <element ref="samlp:StatusDetail" minOccurs="0"/>
   </sequence>
</complexType>
```

2) Element <StatusCode>

The `<StatusCode>` element specifies a code or a set of nested codes representing the status of the corresponding request. The `<StatusCode>` element has the following element and attribute:

– `Value` [Required]

  The status code value. This attribute contains a URI reference. The value of the topmost `<StatusCode>` element must be from the top-level list provided in this clause.

– `<StatusCode>` [Optional]

  A subordinate status code that provides more specific information on an error condition. Responders may omit subordinate status codes in order to prevent attacks that seek to probe for additional information by intentionally presenting erroneous requests.

The permissible top-level `<StatusCode>` values are as follows:

```
urn:oasis:names:tc:SAML:2.0:status:Success
```

The request succeeded. Additional information may be returned in the `<StatusMessage>` and/or `<StatusDetail>` elements.

```
urn:oasis:names:tc:SAML:2.0:status:Requester
```

The request could not be performed due to an error on the part of the requester.

```
urn:oasis:names:tc:SAML:2.0:status:Responder
```

The request could not be performed due to an error on the part of the SAML responder or SAML authority.

```
urn:oasis:names:tc:SAML:2.0:status:VersionMismatch
```

The SAML responder could not process the request because the version of the request message was incorrect.

The following second-level status codes are referenced at various places in this Recommendation. Additional second-level status codes may be defined in future versions of the SAML Recommendation. System entities are free to define more specific status codes by defining appropriate URI references.

```
urn:oasis:names:tc:SAML:2.0:status:AuthnFailed
```

The responding provider was unable to successfully authenticate the principal.

```
urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue
```

Unexpected or invalid content was encountered within a `<saml:Attribute>` or `<saml:AttributeValue>` element.

```
urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy
```

The responding provider cannot or will not support the requested name identifier policy.

```
urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext
```

The specified authentication context requirements cannot be met by the responder.

```
urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP
```

Used by an intermediary to indicate that none of the supported identity provider `<Loc>` elements in an `<IDPList>` can be resolved or that none of the supported identity providers are available.

```
urn:oasis:names:tc:SAML:2.0:status:NoPassive
```

Indicates the responding provider cannot authenticate the principal passively, as has been requested.

```
urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP
```

Used by an intermediary to indicate that none of the identity providers in an `<IDPList>` are supported by the intermediary.

```
urn:oasis:names:tc:SAML:2.0:status:PartialLogout
```

Used by a session authority to indicate to a session participant that it was not able to propagate logout to all other session participants.

```
urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded
```

Indicates that a responding provider cannot authenticate the principal directly and is not permitted to proxy the request further.

```
urn:oasis:names:tc:SAML:2.0:status:RequestDenied
```

The SAML responder or SAML authority is able to process the request but has chosen not to respond. This status code may be used when there is concern about the security context of the request message or the sequence of request messages received from a particular requester.

```
urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported
```

The SAML responder or SAML authority does not support the request.

```
urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated
```

The SAML responder cannot process any requests with the protocol version specified in the request.

```
urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh
```

The SAML responder cannot process the request because the protocol version specified in the request message is a major upgrade from the highest protocol version supported by the responder.

```
urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow
```

The SAML responder cannot process the request because the protocol version specified in the request message is too low.

```
urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized
```

The resource value provided in the request message is invalid or unrecognized.

```
urn:oasis:names:tc:SAML:2.0:status:TooManyResponses
```

The response message would contain more elements than the SAML responder is able to return.

```
urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
```

An entity that has no knowledge of a particular attribute profile has been presented with an attribute drawn from that profile.

```
urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal
```

The responding provider does not recognize the principal specified or implied by the request.

```
urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding
```

The SAML responder cannot properly fulfil the request using the protocol binding specified in the request.

The following schema fragment defines the `<StatusCode>` element and its **StatusCodeType** complex type:

```
<element name="StatusCode" type="samlp:StatusCodeType"/>
<complexType name="StatusCodeType">
    <sequence>
       <element ref="samlp:StatusCode" minOccurs="0"/>
    </sequence>
    <attribute name="Value" type="anyURI" use="required"/>
</complexType>
```

3) Element <StatusMessage>

The `<StatusMessage>` element specifies a message that may be returned to an operator:

The following schema fragment defines the `<StatusMessage>` element:

```
<element name="StatusMessage" type="string"/>
```

4) Element <StatusDetail>

The `<StatusDetail>` element may be used to specify additional information concerning the status of the request. The additional information consists of zero or more elements from any namespace, with no requirement for a schema to be present or for schema validation of the `<StatusDetail>` contents.

The following schema fragment defines the `<StatusDetail>` element and its **StatusDetailType** complex type:

```
<element name="StatusDetail" type="samlp:StatusDetailType"/>
<complexType name="StatusDetailType">
    <sequence>
       <any namespace="##any" processContents="lax" minOccurs="0"
       maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

8.2.3 Assertion query and request protocol

This clause defines messages and processing rules for requesting existing assertions by reference or querying for assertions by subject and statement type.

8.2.3.1 Element <AssertionIDRequest>

If the requester knows the unique identifier of one or more assertions, the `<AssertionIDRequest>` message element can be used to request that they be returned in a `<Response>` message. The `<saml:AssertionIDRef>` element is used to specify each assertion to return.

The following schema fragment defines the `<AssertionIDRequest>` element:

```
<element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>
<complexType name="AssertionIDRequestType">
      <complexContent>
            <extension base="samlp:RequestAbstractType">
                  <sequence>
                        <element ref="saml:AssertionIDRef"
maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

8.2.3.2 Queries

The following subclauses define the SAML query request messages.

8.2.3.2.1 Element <SubjectQuery>

The `<SubjectQuery>` message element is an extension point that allows new SAML queries to be defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the `<saml:Subject>` element (defined in 8.1.4) to **RequestAbstractType**.

The following schema fragment defines the `<SubjectQuery>` element and its **SubjectQueryAbstractType** complex type:

```
<element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
<complexType name="SubjectQueryAbstractType" abstract="true">
      <complexContent>
             <extension base="samlp:RequestAbstractType">
                     <sequence>
                            <element ref="saml:Subject"/>
                     </sequence>
             </extension>
      </complexContent>
</complexType>
```

8.2.3.2.2 Element <AuthnQuery>

The `<AuthnQuery>` message element is used to make the query "What assertions containing authentication statements are available for this subject?" A successful `<Response>` will contain one or more assertions containing authentication statements.

The `<AuthnQuery>` message must not be used as a request for a new authentication using credentials provided in the request. `<AuthnQuery>` is a request for statements about authentication acts that have occurred in a previous interaction between the indicated subject and the authentication authority.

This element is of type **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following element and attribute:

– `SessionIndex` [Optional]

  If present, specifies a filter for possible responses. Such a query asks the question "What assertions containing authentication statements do you have for this subject within the context of the supplied session information?"

– `<RequestedAuthnContext>` [Optional]

  If present, specifies a filter for possible responses. Such a query asks the question "What assertions containing authentication statements do you have for this subject that satisfy the authentication context requirements in this element?"

In response to an authentication query, a SAML authority returns assertions with authentication statements as follows:

- Rules given in 8.2.3.4 for matching against the `<Subject>` element of the query identify the assertions that may be returned.
- If the `SessionIndex` attribute is present in the query, at least one `<AuthnStatement>` element in the set of returned assertions must contain a `SessionIndex` attribute that matches the `SessionIndex` attribute in the query. It is optional for the complete set of all such matching assertions to be returned in the response.
- If the `<RequestedAuthnContext>` element is present in the query, at least one `<AuthnStatement>` element in the set of returned assertions must contain an `<AuthnContext>` element that satisfies the element in the query. It is optional for the complete set of all such matching assertions to be returned in the response.

The following schema fragment defines the `<AuthnQuery>` element and its **AuthnQueryType** complex type:

```
<element name="AuthnQuery" type="samlp:AuthnQueryType"/>
<complexType name="AuthnQueryType">
       <complexContent>
              <extension base="samlp:SubjectQueryAbstractType">
                     <sequence>
                            <element ref="samlp:RequestedAuthnContext"
minOccurs="0"/>
                     </sequence>
                     <attribute name="SessionIndex" type="string"
use="optional"/>
              </extension>
       </complexContent>
</complexType>
```

1) Element <RequestedAuthnContext>

The `<RequestedAuthnContext>` element specifies the authentication context requirements of authentication statements returned in response to a request or query. Its **RequestedAuthnContextType** complex type defines the following elements and attributes:

– `<saml:AuthnContextClassRef>` or `<saml:AuthnContextDeclRef>` [One or More]

Specifies one or more URI references identifying authentication context classes or declarations. These elements are defined in 8.1.7.2.2. For more information about authentication context classes, see clause 12.

– `Comparison` [Optional]

Specifies the comparison method used to evaluate the requested context classes or statements, one of "`exact`", "`minimum`", "`maximum`", or "`better`". The default is "`exact`".

Either a set of class references or a set of declaration references can be used. The set of supplied references must be evaluated as an ordered set, where the first element is the most preferred authentication context class or declaration. If none of the specified classes or declarations can be satisfied in accordance with the rules below, then the responder must return a `<Response>` message with a second-level `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext`.

If `Comparison` is set to "`exact`" or omitted, then the resulting authentication context in the authentication statement must be the exact match of at least one of the authentication contexts specified.

If `Comparison` is set to "`minimum`", then the resulting authentication context in the authentication statement must be at least as strong (as deemed by the responder) as one of the authentication contexts specified.

If `Comparison` is set to "`better`", then the resulting authentication context in the authentication statement must be stronger (as deemed by the responder) than any one of the authentication contexts specified.

If `Comparison` is set to "`maximum`", then the resulting authentication context in the authentication statement must be as strong as possible (as deemed by the responder) without exceeding the strength of at least one of the authentication contexts specified.

The following schema fragment defines the `<RequestedAuthnContext>` element and its **RequestedAuthnContextType** complex type:

```
<element name="RequestedAuthnContext" type="samlp:RequestedAuthnContextType"/>
<complexType name="RequestedAuthnContextType">
 <choice>
        <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
        <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>
 </choice>
 <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
use="optional"/>
</complexType>
<simpleType name="AuthnContextComparisonType">
 <restriction base="string">
        <enumeration value="exact"/>
        <enumeration value="minimum"/>
        <enumeration value="maximum"/>
        <enumeration value="better"/>
 </restriction>
</simpleType>
```

8.2.3.2.3 Element <AttributeQuery>

The `<AttributeQuery>` element is used to make the query "Return the requested attributes for this subject". A successful response will be in the form of assertions containing attribute statements, to the extent allowed by policy. This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following element:

– `<saml:Attribute>` [Any Number]

Each `<saml:Attribute>` element specifies an attribute whose value(s) are to be returned. If no attributes are specified, it indicates that all attributes allowed by policy are requested. If a given `<saml:Attribute>` element contains one or more `<saml:AttributeValue>` elements, then if that attribute is returned in the response, it must not contain any values that are not equal to the values specified in the query. In the absence of equality rules specified by particular profiles or attributes, equality is defined as an identical XML representation of the value. For more information on `<saml:Attribute>`, see 8.1.7.3.1.

A single query must not contain two `<saml:Attribute>` elements with the same `Name` and `NameFormat` values (that is, a given attribute must be named only once in a query).

In response to an attribute query, a SAML authority returns assertions with attribute statements as follows:

- Rules given in 8.2.3.4 for matching against the `<Subject>` element of the query identify the assertions that may be returned.
- If any `<Attribute>` elements are present in the query, they constrain/filter the attributes and optionally the values returned as noted above.
- The attributes and values returned may also be constrained by application-specific policy considerations.

The second-level status codes `urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile and urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue` may be used to indicate problems with the interpretation of attribute or value information in a query.

The following schema fragment defines the `<AttributeQuery>` element and its **AttributeQueryType** complex type:

```
<element name="AttributeQuery" type="samlp:AttributeQueryType"/>
<complexType name="AttributeQueryType">
       <complexContent>
              <extension base="samlp:SubjectQueryAbstractType">
                     <sequence>
                            <element ref="saml:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
                     </sequence>
              </extension>
       </complexContent>
</complexType>
```

8.2.3.2.4 Element <AuthzDecisionQuery>

The `<AuthzDecisionQuery>` element is used to make the query "Should these actions on this resource be allowed for this subject, given this evidence?" A successful response will be in the form of assertions containing authorization decision statements.

NOTE – The `<AuthzDecisionQuery>` feature has been frozen as of SAML V2.0, with no future enhancements planned. Users who require additional functionality may want to consider the eXtensible Access Control Markup Language (see ITU-T Rec. X.1142), which offers enhanced authorization decision features.

This element is of type **AuthzDecisionQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following elements and attribute:

– `Resource` [Required]

A URI reference indicating the resource for which authorization is requested.

– `<saml:Action>` [One or More]

The actions for which authorization is requested. For more information on this element, see 8.1.7.4.2.

– `<saml:Evidence>` [Optional]

A set of assertions that the SAML authority may rely on in making its authorization decision. For more information on this element, see 8.1.7.4.3.

In response to an authorization decision query, a SAML authority returns assertions with authorization decision statements as follows:

- Rules given in 8.2.3.4 for matching against the `<Subject>` element of the query identify the assertions that may be returned.

The following schema fragment defines the `<AuthzDecisionQuery>` element and its **AuthzDecisionQueryType** complex type:

```
<element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>
<complexType name="AuthzDecisionQueryType">
 <complexContent>
       <extension base="samlp:SubjectQueryAbstractType">
              <sequence>
                     <element ref="saml:Action" maxOccurs="unbounded"/>
                     <element ref="saml:Evidence" minOccurs="0"/>
              </sequence>
              <attribute name="Resource" type="anyURI" use="required"/>
       </extension>
 </complexContent>
</complexType>
```

8.2.3.3 Element <Response>

The `<Response>` message element is used when a response consists of a list of zero or more assertions that satisfy the request. It has the complex type **ResponseType**, which extends **StatusResponseType** and adds the following elements:

– `<saml:Assertion>` or `<saml:EncryptedAssertion>` [Any Number]

Specifies an assertion by value, or optionally an encrypted assertion by value. See 8.1.3.3 for more information on these elements.

The following schema fragment defines the `<Response>` element and its **ResponseType** complex type:

```
<element name="Response" type="samlp:ResponseType"/>
<complexType name="ResponseType">
       <complexContent>
              <extension base="samlp:StatusResponseType">
                     <choice minOccurs="0" maxOccurs="unbounded">
                            <element ref="saml:Assertion"/>
                            <element ref="saml:EncryptedAssertion"/>
                     </choice>
              </extension>
       </complexContent>
</complexType>
```

8.2.3.4 Processing rules

In response to a SAML-defined query message, every assertion returned by a SAML authority must contain a `<saml:Subject>` element that **strongly matches** the `<saml:Subject>` element found in the query.

A `<saml:Subject>` element S1 strongly matches S2 if and only if the following two conditions both apply:

- If S2 includes an identifier element (`<BaseID>`, `<NameID>`, or `<EncryptedID>`), then S1 must include an identical identifier element, but the element may be encrypted (or not) in either S1 or S2. In other words, the decrypted form of the identifier must be identical in S1 and S2. "Identical" means that the identifier element's content and attribute values must be the same. An encrypted identifier will be identical to the original according to this definition, once decrypted.
- If S2 includes one or more `<saml:SubjectConfirmation>` elements, then S1 must include at least one `<saml:SubjectConfirmation>` element such that S1 can be confirmed in the manner described by at least one `<saml:SubjectConfirmation>` element in S2.

As an example of what is and is not permitted, S1 could contain a `<saml:NameID>` with a particular `Format` value, and S2 could contain a `<saml:EncryptedID>` element that is the result of encrypting S1's `<saml:NameID>` element. However, S1 and S2 cannot contain a `<saml:NameID>` element with different `Format` values and element content, even if the two identifiers are considered to refer to the same principal.

If the SAML authority cannot provide an assertion with any statements satisfying the constraints expressed by a query or assertion reference, the `<Response>` element must not contain an `<Assertion>` element and must include a `<StatusCode>` element with the value `urn:oasis:names:tc:SAML:2.0:status:Success`.

All other processing rules associated with the underlying request and response messages must be observed.

8.2.4 Authentication request protocol

When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing authentication statements to establish a security context at one or more relying parties, it can use the authentication request protocol to send an `<AuthnRequest>` message element to a SAML authority and request that it return a `<Response>` message containing one or more such assertions. Such assertions may contain additional statements of any type, but at least one assertion must contain at least one authentication statement. A SAML authority that supports this protocol is also termed an identity provider.

Apart from this requirement, the specific contents of the returned assertions depend on the profile or context of use. Also, the exact means by which the principal or agent authenticates to the identity provider is not specified, though the means of authentication might impact the content of the response. Other issues related to the validation of authentication credentials by the identity provider or any communication between the identity provider and any other entities involved in the authentication process are also out of scope of this protocol.

The descriptions and processing rules in the following subclauses reference the following actors, many of whom might be the same entity in a particular profile of use:

– Requester

  The entity who creates the authentication request and to whom the response is to be returned.

– Presenter

  The entity who presents the request to the identity provider and either authenticates itself during the transmission of the message, or relies on an existing security context to establish its identity. If not the requester, the presenter acts as an intermediary between the requester and the responding identity provider.

– Requested Subject

  The entity about whom one or more assertions are being requested.

– Attesting Entity

  The entity or entities expected to be able to satisfy one of the `<SubjectConfirmation>` elements of the resulting assertion(s).

– Relying Party

  The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by the profile or context of use, generally to establish a security context.

– Identity Provider

  The entity to whom the presenter gives the request and from whom the presenter receives the response.

Element <AuthnRequest>

To request that an identity provider issue an assertion with an authentication statement, a presenter authenticates to that identity provider (or relies on an existing security context) and sends it an `<AuthnRequest>` message that describes the properties that the resulting assertion needs to have to satisfy its purpose. Among these properties may be information that relates to the content of the assertion and/or information that relates to how the resulting `<Response>` message should be delivered to the requester. The process of authentication of the presenter may take place before, during, or after the initial delivery of the `<AuthnRequest>` message.

The requester might not be the same as the presenter of the request if, for example, the requester is a relying party that intends to use the resulting assertion to authenticate or authorize the requested subject so that the relying party can decide whether to provide a service.

The `<AuthnRequest>` message should be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and adds the following elements and attributes, all of which are optional in general, but may be required by specific profiles:

– `<saml:Subject>` [Optional]

  Specifies the requested subject of the resulting assertion(s). This may include one or more `<saml:SubjectConfirmation>` elements to indicate how and/or by whom the resulting assertions can be confirmed. For more information on this element, see 8.1.4.

If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the requested subject. If no <saml:SubjectConfirmation> elements are included, then the presenter is presumed to be the only attesting entity required and the method is implied by the profile of use and/or the policies of the identity provider.

– <NameIDPolicy> [Optional]

Specifies constraints on the name identifier to be used to represent the requested subject. If omitted, then any type of identifier supported by the identity provider for the requested subject can be used, constrained by any relevant deployment-specific policies, with respect to privacy, for example.

– <saml:Conditions> [Optional]

Specifies the SAML conditions the requester expects to limit the validity and/or use of the resulting assertion(s). The responder may modify or supplement this set as it deems necessary. The information in this element is used as input to the process of constructing the assertion, rather than as conditions on the use of the request itself. (For more information on this element, see 8.1.5.)

– <RequestedAuthnContext> [Optional]

Specifies the requirements, if any, that the requester places on the authentication context that applies to the responding provider's authentication of the presenter.

– <Scoping> [Optional]

Specifies a set of identity providers trusted by the requester to authenticate the presenter, as well as limitations and context related to proxying of the <AuthnRequest> message to subsequent identity providers by the responder.

– ForceAuthn [Optional]

A boolean value. If "true", the identity provider must authenticate the presenter directly rather than rely on a previous security context. If a value is not provided, the default is "false". However, if both ForceAuthn and IsPassive are "true", the identity provider must not freshly authenticate the presenter unless the constraints of IsPassive can be met.

– IsPassive [Optional]

A boolean value. If "true", the identity provider and the user agent itself must not visibly take control of the user interface from the requester and interact with the presenter in a noticeable fashion. If a value is not provided, the default is "false".

– AssertionConsumerServiceIndex [Optional]

Indirectly identifies the location to which the <Response> message should be returned to the requester. It applies only to profiles in which the requester is different from the presenter, such as the Web Browser SSO profile in this Recommendation. The identity provider must have a trusted means to map the index value in the attribute to a location associated with the requester. Clause 9 provides one possible mechanism. If omitted, then the identity provider must return the <Response> message to the default location associated with the requester for the profile of use. If the index specified is invalid, then the identity provider may return an error <Response> or it may use the default location. This attribute is mutually exclusive with the AssertionConsumerServiceURL and ProtocolBinding attributes.

– AssertionConsumerServiceURL [Optional]

Specifies by value the location to which the <Response> message must be returned to the requester. The responder must ensure by some means that the value specified is in fact associated with the requester. Clause 9 provides one possible mechanism; signing the enclosing <AuthnRequest> message is another. This attribute is mutually exclusive with the AssertionConsumerServiceIndex attribute and is typically accompanied by the ProtocolBinding attribute.

– ProtocolBinding [Optional]

A URI reference that identifies a SAML protocol binding to be used when returning the <Response> message. See clause 10 for more information about protocol bindings and URI references defined for them. This attribute is mutually exclusive with the AssertionConsumerServiceIndex attribute and is typically accompanied by the AssertionConsumerServiceURL attribute.

– `AttributeConsumingServiceIndex` [Optional]

Indirectly identifies information associated with the requester describing the SAML attributes the requester desires or requires to be supplied by the identity provider in the `<Response>` message. The identity provider must have a trusted means to map the index value in the attribute to information associated with the requester. Clause 9 provides one possible mechanism. The identity provider may use this information to populate one or more `<saml:AttributeStatement>` elements in the assertion(s) it returns.

– `ProviderName` [Optional]

Specifies the human-readable name of the requester for use by the presenter's user agent or the identity provider.

See 8.2.4.4 for general processing rules regarding this message.

The following schema fragment defines the `<AuthnRequest>` element and its **AuthnRequestType** complex type:

```
<element name="AuthnRequest" type="samlp:AuthnRequestType"/>
<complexType name="AuthnRequestType">
      <complexContent>
             <extension base="samlp:RequestAbstractType">
                    <sequence>
                           <element ref="saml:Subject" minOccurs="0"/>
                           <element ref="samlp:NameIDPolicy" minOccurs="0"/>
                           <element ref="saml:Conditions" minOccurs="0"/>
                           <element ref="samlp:RequestedAuthnContext"
minOccurs="0"/>
                           <element ref="samlp:Scoping" minOccurs="0"/>
                    </sequence>
                    <attribute name="ForceAuthn" type="boolean"
use="optional"/>
                    <attribute name="IsPassive" type="boolean"
use="optional"/>
                    <attribute name="ProtocolBinding" type="anyURI"
use="optional"/>
                    <attribute name="AssertionConsumerServiceIndex"
type="unsignedShort" use="optional"/>
                    <attribute name="AssertionConsumerServiceURL"
type="anyURI" use="optional"/>
                    <attribute name="AttributeConsumingServiceIndex"
type="unsignedShort" use="optional"/>
                    <attribute name="ProviderName" type="string"
use="optional"/>
             </extension>
      </complexContent>
</complexType>
```

8.2.4.1 Element <NameIDPolicy>

The `<NameIDPolicy>` element tailors the name identifier in the subjects of assertions resulting from an `<AuthnRequest>`. Its **NameIDPolicyType** complex type defines the following attributes:

– `Format` [Optional]

Specifies the URI reference corresponding to a name identifier format defined in this or another Recommendation (see 8.7.3 for examples). The additional value of `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` is defined specifically for use within this attribute to indicate a request that the resulting identifier be encrypted.

– `SPNameQualifier` [Optional]

Optionally specifies that the assertion subject's identifier be returned (or created) in the namespace of a service provider other than the requester, or in the namespace of an affiliation group of service providers. See for example the definition of `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` in this Recommendation.

– `AllowCreate` [Optional]

A boolean value used to indicate whether the identity provider is allowed, in the course of fulfilling the request, to create a new identifier to represent the principal. Defaults to "false". When "false", the requester constrains the identity provider to only issue an assertion to it if an acceptable identifier for the principal has already been established. This does not prevent the identity provider from creating such identifiers outside the context of this specific request (for example, in advance for a large number of principals).

NOTE 1 (informative) – PE14 (see OASIS PE:2006) clarifies the above definition as below:

A boolean value used to indicate whether the requester grants to the identity provider, in the course of fulfilling the request, permission to create a new identifier or to associate an existing identifier representing the principal with the relying party. Defaults to "false" if not present or the entire element is omitted.

NOTE 2 (informative) – PE14 (see OASIS PE:2006) suggests to add the following text to the paragraph below:

The AllowCreate attribute may be used by some deployments to influence the creation of state maintained by the identity provider pertaining to the use of a name identifier (or any other persistent, uniquely identifying attributes) by a particular relying party, for purposes such as dynamic identifier or attribute creation, tracking of consent, subsequent use of the Name Identifier Management protocol, or other related purposes.

When "false", the requester tries to constrain the identity provider to issue an assertion only if such state has already been established or is not deemed applicable by the identity provider to the use of an identifier. Thus, this does not prevent the identity provider from assuming such information exists outside the context of this specific request (for example, establishing it in advance for a large number of principals).

A value of "true" permits the identity provider to take any related actions it wishes to fulfil the request, subject to any other constraints imposed by the request and policy (the IsPassive attribute, for example).

Generally, requesters cannot assume specific behaviour from identity providers regarding the initial creation or association of identifiers on their behalf, as these are details left to implementations or deployments. Absent specific profiles governing the use of this attribute, it might be used as a hint to identity providers about the requester's intention to store the identifier or link it to a local value.

A value of "false" might be used to indicate that the requester is not prepared or able to do so and save the identity provider wasted effort.

Requesters that do not make specific use of this attribute should generally set it to "true" to maximize interoperability. The use of the AllowCreate attribute must not be used and should be ignored in conjunction with requests for or assertions issued with name identifiers with a Format of urn:oasis:names:tc:SAML:2.0:nameid-format:transient (they preclude any such state in and of themselves).

When this element is used, if the content is not understood by or acceptable to the identity provider, then a <Response> message element must be returned with an error <Status>, and may contain a second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy.

If the Format value is omitted or set to urn:oasis:names:tc:SAML:2.0:nameid-format:unspecified, then the identity provider is free to return any kind of identifier, subject to any additional constraints due to the content of this element or the policies of the identity provider or principal.

The special Format value urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted indicates that the resulting assertion(s) must contain <EncryptedID> elements instead of plaintext. The underlying name identifier's unencrypted form can be of any type supported by the identity provider for the requested subject.

NOTE 3 (informative) – PE6 (see OASIS PE:2006) suggests to add the following text to the end of the above paragraph:

It is not possible for the service provider to specifically request that a particular kind of identifier be returned if it asks for encryption. The <md:NameIDFormat> metadata element in clause 9 or other out-of-band means may be used to determine what kind of identifier to encrypt and return.

NOTE 4 (informative) – PE15 (see OASIS PE:2006) suggests to add the following paragraph:

When a Format defined in 8.7.3.7 is used other than urn:oasis:names:TC:SAML:2.0:nameid-format:unspecified or urn:oasis:names:TC:SAML:2.0:nameid-format:encrypted, then if the identity provider returns any assertions:

- the Format value of the <NameID> within the <Subject> of any <Assertion> must be identical to the Format value supplied in the <NameIDPolicy>; and
- if SPNameQualifier is not omitted in <NameIDPolicy>, the SPNameQualifier value of the <NameID> within the <Subject> of any <Assertion> must be identical to the SPNameQualifier value supplied in the <NameIDPolicy>.

Regardless of the Format in the <NameIDPolicy>, the identity provider may return an <EncryptedID> in the resulting assertion subject if the policies in effect at the identity provider (possibly specific to the service provider) require that an encrypted identifier be used.

If the requester wishes to permit the identity provider to establish a new identifier for the principal if none exists, it must include this element with the AllowCreate attribute set to "true". Otherwise, only a principal for whom the identity provider has previously established an identifier usable by the requester can be authenticated successfully. This is primarily useful in conjunction with the urn:oasis:names:tc:SAML:2.0:nameid-format:persistent Format value, (see clause 12).

NOTE 5 (informative) – PE14 (see OASIS PE:2006) suggests to ignore the above paragraph.

The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType** complex type:

```
<element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
<complexType name="NameIDPolicyType">
      <attribute name="Format" type="anyURI" use="optional"/>
      <attribute name="SPNameQualifier" type="string" use="optional"/>
      <attribute name="AllowCreate" type="boolean" use="optional"/>
</complexType>
```

8.2.4.2 Element <Scoping>

The `<Scoping>` element specifies the identity providers trusted by the requester to authenticate the presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to subsequent identity providers by the responder. Its **ScopingType** complex type defines the following elements and attribute:

– `ProxyCount` [Optional]

Specifies the number of proxying indirections permissible between the identity provider that receives this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count of zero permits no proxying, while omitting this attribute expresses no such restriction.

– `<IDPList>` [Optional]

An advisory list of identity providers and associated information that the requester deems acceptable to respond to the request.

– `<RequesterID>` [Zero or More]

Identifies the set of requesting entities on whose behalf the requester is acting. Used to communicate the chain of requesters when proxying occurs, as described in 8.2.4.5. See 8.7.3.6 for a description of entity identifiers.

In profiles specifying an active intermediary, the intermediary may examine the list and return a `<Response>` message with an error `<Status>` and a second-level `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP` or `urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP` if it cannot contact or does not support any of the specified identity providers.

The following schema fragment defines the `<Scoping>` element and its **ScopingType** complex type:

```
<element name="Scoping" type="samlp:ScopingType"/>
<complexType name="ScopingType">
      <sequence>
             <element ref="samlp:IDPList" minOccurs="0"/>
             <element ref="samlp:RequesterID" minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
      <attribute name="ProxyCount" type="nonNegativeInteger"
use="optional"/>
</complexType>
<element name="RequesterID" type="anyURI"/>
```

8.2.4.3 Element <IDPList>

The `<IDPList>` element specifies the identity providers trusted by the requester to authenticate the presenter. Its **IDPListType** complex type defines the following elements:

– `<IDPEntry>` [One or More]

Information about a single identity provider.

– `<GetComplete>` [Optional]

If the `<IDPList>` is not complete, using this element specifies a URI reference that can be used to retrieve the complete list. Retrieving the resource associated with the URI must result in an XML instance whose root element is an `<IDPList>` that does not itself contain a `<GetComplete>` element.

The following schema fragment defines the `<IDPList>` element and its **IDPListType** complex type:

```
<element name="IDPList" type="samlp:IDPListType"/>
<complexType name="IDPListType">
      <sequence>
            <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
            <element ref="samlp:GetComplete" minOccurs="0"/>
      </sequence>
</complexType>
<element name="GetComplete" type="anyURI"/>
```

The `<IDPEntry>` element specifies a single identity provider trusted by the requester to authenticate the presenter. Its **IDPEntryType** complex type defines the following attributes:

– `ProviderID` [Required]

  The unique identifier of the identity provider. See 8.7.3.6 for a description of such identifiers.

– `Name` [Optional]

  A human-readable name for the identity provider.

– `Loc` [Optional]

  A URI reference representing the location of a profile-specific endpoint supporting the authentication request protocol. The binding to be used must be understood from the profile of use.

  The following schema fragment defines the `<IDPEntry>` element and its **IDPEntryType** complex type:

```
<element name="IDPEntry" type="samlp:IDPEntryType"/>
<complexType name="IDPEntryType">
      <attribute name="ProviderID" type="anyURI" use="required"/>
      <attribute name="Name" type="string" use="optional"/>
      <attribute name="Loc" type="anyURI" use="optional"/>
</complexType>
```

8.2.4.4 Processing rules

The `<AuthnRequest>` and `<Response>` exchange supports a variety of usage scenarios and is therefore typically profiled for use in a specific context in which this optionality is constrained and specific kinds of input and output are required or prohibited. The following processing rules apply as invariant behaviour across any profile of this protocol exchange. All other processing rules associated with the underlying request and response messages must also be observed.

The responder must ultimately reply to an `<AuthnRequest>` with a `<Response>` message containing one or more assertions that meet the specifications defined by the request, or with a `<Response>` message containing a `<Status>` describing the error that occurred. The responder may conduct additional message exchanges with the presenter as needed to initiate or complete the authentication process, subject to the nature of the protocol binding and the authentication mechanism. As described in the next clause, this includes proxying the request by directing the presenter to another identity provider by issuing its own `<AuthnRequest>` message, so that the resulting assertion can be used to authenticate the presenter to the original responder, in effect using SAML as the authentication mechanism.

If the responder is unable to authenticate the presenter or does not recognize the requested subject, or if prevented from providing an assertion by policies in effect at the identity provider (for example the intended subject has prohibited the identity provider from providing assertions to the relying party), then it must return a `<Response>` with an error `<Status>`, and may return a second-level `<StatusCode>` of:
`urn:oasis:names:tc:SAML:2.0:status:AuthnFailed`; or
`urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal`.

If the `<saml:Subject>` element in the request is present, then the resulting assertions' `<saml:Subject>` must **strongly match** the request `<saml:Subject>`, as described in 8.2.3.4, except that the identifier may be in a different format if specified by `<NameIDPolicy>`. In such a case, the identifier's physical content may be different, but it must refer to the same principal.

All of the content defined specifically within `<AuthnRequest>` is optional, although some may be required by certain profiles. In the absence of any specific content at all, the following behaviour is implied:

- The assertion(s) returned must contain a `<saml:Subject>` element that represents the presenter. The identifier type and format are determined by the identity provider. At least one statement in at least one assertion must be a `<saml:AuthnStatement>` that describes the authentication performed by the responder or authentication service associated with it.

- The request presenter should, to the extent possible, be the only attesting entity able to satisfy the `<saml:SubjectConfirmation>` of the assertion(s). In the case of weaker confirmation methods, binding-specific or other mechanisms will be used to help satisfy this requirement.
- The resulting assertion(s) must contain a `<saml:AudienceRestriction>` element referencing the requester as an acceptable relying party. Other audiences may be included as deemed appropriate by the identity provider.

8.2.4.5 Proxying

If an identity provider that receives an `<AuthnRequest>` has not yet authenticated the presenter or cannot directly authenticate the presenter, but believes that the presenter has already authenticated to another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new `<AuthnRequest>` on its own behalf to be presented to the other identity provider, or a request in whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying identity provider.

Upon the successful return of a `<Response>` (or non-SAML equivalent) to the proxying provider, the enclosed assertion or non-SAML equivalent may be used to authenticate the presenter so that the proxying provider can issue an assertion of its own in response to the original `<AuthnRequest>,` completing the overall message exchange. Both the proxying and authenticating identity providers may include constraints on proxying activity in the messages and assertions they issue, as described in previous subclauses and below.

The requester can influence proxy behaviour by including a `<Scoping>` element where the provider sets a desired `ProxyCount` value and/or indicates a list of preferred identity providers which may be proxied by including an ordered `<IDPList>` of preferred providers.

An identity provider can control secondary use of its assertions by proxying identity providers using a `<ProxyRestriction>` element in the assertions it issues.

An identity provider may proxy an `<AuthnRequest>` if the `<ProxyCount>` attribute is omitted or is greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider may choose to proxy for a provider specified in the `<IDPList>`, if provided, but is not required to do so.

An identity provider must not proxy a request where `<ProxyCount>` is set to zero. The identity provider must return an error `<Status>` containing a second-level `<StatusCode>` value of `urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded`, unless it can directly authenticate the presenter.

If it chooses to proxy to a SAML identity provider, when creating the new `<AuthnRequest>`, the proxying identity provider must include equivalent or stricter forms of all the information included in the original request (such as authentication context policy). Note, however, that the proxying provider is free to specify whatever `<NameIDPolicy>` it wishes to maximize the chances of a successful response.

If the authenticating identity provider is not a SAML identity provider, then the proxying provider must have some other way to ensure that the elements governing user agent interaction (`<IsPassive>`, for example) will be honoured by the authenticating provider.

The new `<AuthnRequest>` must contain a `<ProxyCount>` attribute with a value of at most one less than the original value. If the original request does not contain a `<ProxyCount>` attribute, then the new request should contain a `<ProxyCount>` attribute.

If an `<IDPList>` was specified in the original request, the new request must also contain an `<IDPList>`. The proxying identity provider may add additional identity providers to the end of the `<IDPList>`, but must not remove any from the list.

The authentication request and response are processed in normal fashion, in accordance with the rules given in this clause and the profile of use. Once the presenter has authenticated to the proxying identity provider (in the case of SAML by delivering a `<Response>)`, the following steps are followed:

- The proxying identity provider prepares a new assertion on its own behalf by copying in the relevant information from the original assertion or non-SAML equivalent.
- The new assertion's `<saml:Subject>` must contain an identifier that satisfies the original requester's preferences, as defined by its `<NameIDPolicy>` element.

- The `<saml:AuthnStatement>` in the new assertion must include a `<saml:AuthnContext>` element containing a `<saml:AuthenticatingAuthority>` element referencing the identity provider to which the proxying identity provider referred the presenter. If the original assertion contains `<saml:AuthnContext>` information that includes one or more `<saml:AuthenticatingAuthority>` elements, those elements should be included in the new assertion, with the new element placed after them.
- If the authenticating identity provider is not a SAML provider, then the proxying identity provider must generate a unique identifier value for the authenticating provider. This value should be consistent over time across different requests. The value must not conflict with values used or generated by other SAML providers.
- Any other `<saml:AuthnContext>` information may be copied, translated, or omitted in accordance with the policies of the proxying identity provider, provided that the original requirements dictated by the requester are met.

If, in the future, the identity provider is asked to authenticate the same presenter for a second requester, and this request is equally or less strict than the original request (as determined by the proxying identity provider), the identity provider may skip the creation of a new `<AuthnRequest>` to the authenticating identity provider and immediately issue another assertion (assuming the original assertion or non-SAML equivalent it received is still valid).

8.2.5 Artifact resolution protocol

The artifact resolution protocol provides a mechanism by which SAML protocol messages can be transported in a SAML binding by reference instead of by value. Both requests and responses can be obtained by reference using this specialized protocol. A message sender, instead of binding a message to a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a variety of forms, but must support a means by which the receiver can determine who sent it. If the receiver wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML binding protocol to resolve the artifact into the original protocol message.

The most common use for this mechanism is with bindings that cannot easily carry a message because of size constraints, or to enable a message to be communicated via a secure channel between the SAML requester and responder, avoiding the need for a signature.

Depending on the characteristics of the underlying message being passed by reference, the artifact resolution protocol may require protections such as mutual authentication, integrity protection, confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact must exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used by any party.

Regardless of the protocol message obtained, the result of resolving an artifact must be treated exactly as if the message so obtained had been sent originally in place of the artifact.

8.2.5.1 Element <ArtifactResolve>

The `<ArtifactResolve>` message is used to request that a SAML protocol message be returned in an `<ArtifactResponse>` message by specifying an artifact that represents the SAML protocol message. The original transmission of the artifact is governed by the specific protocol binding that is being used.

The `<ArtifactResolve>` message should be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and adds the following element:

– `<Artifact>` [Required]

The artifact value that the requester received and now wishes to translate into the protocol message it represents.

The following schema fragment defines the `<ArtifactResolve>` element and its **ArtifactResolveType** complex type:

```
<element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
<complexType name="ArtifactResolveType">
       <complexContent>
              <extension base="samlp:RequestAbstractType">
                     <sequence>
                            <element ref="samlp:Artifact"/>
                     </sequence>
              </extension>
       </complexContent>
</complexType>
<element name="Artifact" type="string"/>
```

8.2.5.2 Element <ArtifactResponse>

The recipient of an `<ArtifactResolve>` message must respond with an `<ArtifactResponse>` message element. This element is of complex type **ArtifactResponseType**, which extends **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol message being returned. This wrapped message element can be a request or a response.

The `<ArtifactResponse>` message should be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the `<ArtifactResponse>` element and its **ArtifactResponseType** complex type:

```
<element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
<complexType name="ArtifactResponseType">
       <complexContent>
              <extension base="samlp:StatusResponseType">
                     <sequence>
                            <any namespace="##any" processContents="lax"
minOccurs="0"/>
                     </sequence>
              </extension>
       </complexContent>
</complexType>
```

8.2.5.3 Processing rules

If the responder recognizes the artifact as valid, then it responds with the associated protocol message in an `<ArtifactResponse>` message element. Otherwise, it responds with an `<ArtifactResponse>` element with no embedded message. In both cases, the `<Status>` element must include a `<StatusCode>` element with the code value `urn:oasis:names:tc:SAML:2.0:status:Success`. A response message with no embedded message inside it is termed an empty response in the remainder of this clause.

The responder must enforce a one-time-use property on the artifact by ensuring that any subsequent request with the same artifact by any requester results in an empty response as described above.

Some SAML protocol messages, most particularly the `<AuthnRequest>` message in some profiles, may be intended for consumption by any party that receives it and can respond appropriately. In most other cases, however, a message is intended for a specific entity. In such cases, the artifact when issued must be associated with the intended recipient of the message that the artifact represents. If the artifact issuer receives an `<ArtifactResolve>` message from a requester that cannot authenticate itself as the original intended recipient, then the artifact issuer must return an empty response.

The artifact issuer should enforce the shortest practical time limit on the usability of an artifact, such that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and return it in an `<ArtifactResolve>` message to the issuer.

The `<ArtifactResponse>` message's `InResponseTo` attribute must contain the value of the corresponding `<ArtifactResolve>` message's `ID` attribute, but the embedded protocol message will contain its own message identifier, and in the case of an embedded response, may contain a different `InResponseTo` value that corresponds to the original request message to which the embedded message is responding.

All other processing rules associated with the underlying request and response messages must be observed.

8.2.6 Name identifier management protocol

After establishing a name identifier for a principal, an identity provider wishing to change the value and/or format of the identifier that it will use when referring to the principal, or to indicate that a name identifier will no longer be used to refer to the principal, informs service providers of the change by sending them a <ManageNameIDRequest> message.

NOTE 1 (informative) – PE12 (see OASIS PE:2006) identifies the intent of the above paragraph by re-writing it as:

> After establishing a name identifier for a principal, an identity provider wishing to change the value of the identifier that it will use when referring to the principal or to indicate that a name identifier will no longer be used to refer to the principal, informs service providers of the change by sending them a <ManageNameIDRequest> message.

A service provider also uses this message to register or change the SPProvidedID value to be included when the underlying name identifier is used to communicate with it, or to terminate the use of a name identifier between itself and the identity provider.

This protocol is typically not used with "transient" name identifiers, since their value is not intended to be managed on a long term basis.

NOTE 2 (informative) – PE14 (see OASIS PE:2006) clarifies the above text as below:

> This protocol must not be used in conjunction with the
> urn:oasis:names:tc:SAML:2.0:nameidformat:transient <NameID> Format.

8.2.6.1 Element <ManageNameIDRequest>

A provider sends a <ManageNameIDRequest> message to inform the recipient of a changed name identifier or to indicate the termination of the use of a name identifier.

The <ManageNameIDRequest> message should be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **ManageNameIDRequestType**, which extends **RequestAbstractType** and adds the following elements:

– <saml:NameID> or <saml:EncryptedID> [Required]

The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the principal as currently recognized by the identity and service providers prior to this request (for more information on these elements, see 8.1.2).

– <NewID> or <NewEncryptedID> or <Terminate> [Required]

The new identifier value (in plaintext or encrypted form) to be used when communicating with the requesting provider concerning this principal, or an indication that the use of the old identifier has been terminated. In the former case, if the requester is the service provider, the new identifier must appear in subsequent <NameID> elements in the SPProvidedID attribute. If the requester is the identity provider, the new value will appear in subsequent <NameID> elements as the element's content.

NOTE (informative) – PE12 (see OASIS PE:2006) suggests to append to the above paragraph with the following:

> In either case, if the <NewEncryptedID> is used, its encrypted content is just a <NewID> element containing only the new value for the identifier (format and qualifiers cannot be changed once established).

The following schema fragment defines the <ManageNameIDRequest> element and its **ManageNameIDRequestType** complex type:

```
<element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
<complexType name="ManageNameIDRequestType">
      <complexContent>
            <extension base="samlp:RequestAbstractType">
                  <sequence>
                        <choice>
                              <element ref="saml:NameID"/>
                              <element ref="saml:EncryptedID"/>
                        </choice>
                        <choice>
                              <element ref="samlp:NewID"/>
                              <element ref="samlp:NewEncryptedID"/>
                              <element ref="samlp:Terminate"/>
                        </choice>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

```
<element name="NewID" type="string"/>
<element name="NewEncryptedID" type="saml:EncryptedElementType"/>
<element name="Terminate" type="samlp:TerminateType"/>
<complexType name="TerminateType"/>
```

8.2.6.2 Element <ManageNameIDResponse>

The recipient of a `<ManageNameIDRequest>` message must respond with a `<ManageNameIDResponse>` message, which is of type **StatusResponseType** with no additional content.

The `<ManageNameIDResponse>` message should be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the `<ManageNameIDResponse>` element:

```
<element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>
```

8.2.6.3 Processing rules

If the request includes a `<saml:NameID>` (or encrypted version) that the recipient does not recognize, the responding provider must respond with an error `<Status>` and may respond with a second-level `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal`.

NOTE 1 (informative) – PE14 (see OASIS PE:2006) further clarifies the paragraph below. Refer to Appendix VIII for more details.

If the `<Terminate>` element is included in the request, the requesting provider is indicating that (in the case of a service provider) it will no longer accept assertions from the identity provider or (in the case of an identity provider) it will no longer issue assertions to the service provider about the principal. The receiving provider can perform any maintenance with the knowledge that the relationship represented by the name identifier has been terminated. It can choose to invalidate the active session(s) of a principal for whom a relationship has been terminated.

NOTE 2 (informative) – PE8 (see OASIS PE:2006) suggests to replace the last sentence of this paragraph with:

In general it should not invalidate any active session(s) of the principal for whom the relationship has been terminated. If the receiving provider is an identity provider, it should not invalidate any active session(s) of the principal established with other service providers. A requesting provider may send a `<LogoutRequest>` message prior to initiating a name identifier termination by sending a `<ManageNameIDRequest>` message if that is the requesting provider's intent (e.g., the name identifier termination is initiated via an administrator who wished to terminate all user activity). The requesting provider must not send a `<LogoutRequest>` message after the `<ManageNameIDRequest>` message is sent.

If the service provider requests that its identifier for the principal be changed by including a `<NewID>` (or `<NewEncryptedID>`) element, the identity provider must include the element's content as the `SPProvidedID` when subsequently communicating to the service provider regarding this principal.

If the identity provider requests that its identifier for the principal be changed by including a `<NewID>` (or `<NewEncryptedID>`) element, the service provider must use the element's content as the `<saml:NameID>` element content when subsequently communicating with the identity provider regarding this principal.

Neither, either, or both of the original and new identifier may be encrypted (using the `<EncryptedID>` and `<NewEncryptedID>` elements).

In any case, the `<saml:NameID>` content in the request and its associated `SPProvidedID` attribute must contain the most recent name identifier information established between the providers for the principal.

In the case of an identifier with a `Format` of `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`, the `NameQualifier` attribute must contain the unique identifier of the identity provider that created the identifier. If the identifier was established between the identity provider and an affiliation group of which the service provider is a member, then the `SPNameQualifier` attribute must contain the unique identifier of the affiliation group. Otherwise, it must contain the unique identifier of the service provider. These attributes may be omitted if they would otherwise match the value of the containing protocol message's `<Issuer>` element, but this is not recommended due to the opportunity for confusion.

Changes to these identifiers may take a potentially significant amount of time to propagate through the systems at both the requester and the responder. Implementations might wish to allow each party to accept either identifier for some period of time following the successful completion of a name identifier change. Not doing so could result in the inability of the principal to access resources.

All other processing rules associated with the underlying request and response messages must be observed.

8.2.7 Single logout protocol

The single logout protocol provides a message exchange protocol by which all sessions provided by a particular session authority are near-simultaneously terminated. The single logout protocol is used either when a principal logs out at a session participant or when the principal logs out directly at the session authority. This protocol may also be used to log out a principal due to a timeout. The reason for the logout event can be indicated through the Reason attribute.

The principal may have established authenticated sessions with both the session authority and individual session participants, based on assertions containing authentication statements supplied by the session authority.

When the principal invokes the single logout process at a session participant, the session participant must send a `<LogoutRequest>` message to the session authority that provided the assertion containing the authentication statement related to that session at the session participant.

When either the principal invokes a logout at the session authority, or a session participant sends a logout request to the session authority specifying that principal, the session authority should send a `<LogoutRequest>` message to each session participant to which it provided assertions containing authentication statements under its current session with the principal, with the exception of the session participant that sent the `<LogoutRequest>` message to the session authority. It should attempt to contact as many of these participants as it can using this protocol, terminate its own session with the principal, and finally return a `<LogoutResponse>` message to the requesting session participant, if any.

8.2.7.1 Element <LogoutRequest>

A session participant or session authority sends a `<LogoutRequest>` message to indicate that a session has been terminated.

The `<LogoutRequest>` message should be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **LogoutRequestType**, which extends **RequestAbstractType** and adds the following elements and attributes:

– `NotOnOrAfter` [Optional]

The time at which the request expires, after which the recipient may discard the message. The time value is encoded in UTC, as described in 7.3.

– `Reason` [Optional]

An indication of the reason for the logout, in the form of a URI reference.

NOTE 1 (informative) – PE10 (see OASIS PE:2006) suggests to replace the above text with:

The `Reason` attribute is specified as a string in the schema. This specification further restricts the schema by requiring that the `Reason` attribute must be in the form of a URI reference.

– `<saml:BaseID>` or `<saml:NameID>` or `<saml:EncryptedID>` [Required]

The identifier and associated attributes (in plaintext or encrypted form) that specify the principal as currently recognized by the identity and service providers prior to this request. (For more information on this element, see 8.1.2.)

– `<SessionIndex>` [Optional]

The identifier that indexes this session at the message recipient.

NOTE 2 (informative) – PE38 (see OASIS PE:2006) clarifies the above text as below:

The index of the session between the principal identified by the `<saml:BaseID>`, `<saml:NameID>`, or `<saml:EncryptedID>` element, and the session authority. This must correlate to the `SessionIndex` attribute, if any, in the `<saml:AuthnStatement>` of the assertion used to establish the session that is being terminated.

The following schema fragment defines the `<LogoutRequest>` element and associated **LogoutRequestType** complex type:

```
<element name="LogoutRequest" type="samlp:LogoutRequestType"/>
    <complexType name="LogoutRequestType">
        <complexContent>
            <extension base="samlp:RequestAbstractType">
                <sequence>
                    <choice>
                        <element ref="saml:BaseID"/>
                        <element ref="saml:NameID"/>
                        <element ref="saml:EncryptedID"/>
                    </choice>
                    <element ref="samlp:SessionIndex" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
                <attribute name="Reason" type="string" use="optional"/>
                <attribute name="NotOnOrAfter" type="dateTime"
use="optional"/>
            </extension>
        </complexContent>
    </complexType>
    <element name="SessionIndex" type="string"/>
```

8.2.7.2 Element <LogoutResponse>

The recipient of a `<LogoutRequest>` message must respond with a `<LogoutResponse>` message, of type **StatusResponseType**, with no additional content specified.

The `<LogoutResponse>` message should be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the `<LogoutResponse>` element:

```
<element name="LogoutResponse" type="samlp:StatusResponseType"/>
```

8.2.7.3 Processing rules

The message sender may use the `Reason` attribute to indicate the reason for sending the `<LogoutRequest>`. The following values are defined by this Recommendation for use by all message senders; other values may be agreed on between participants:

`urn:oasis:names:tc:SAML:2.0:logout:user`

Specifies that the message is being sent because the principal wishes to terminate the indicated session.

`urn:oasis:names:tc:SAML:2.0:logout:admin`

Specifies that the message is being sent because an administrator wishes to terminate the indicated session for that principal.

All other processing rules associated with the underlying request and response messages must be observed.

Additional processing rules are provided in the following subclauses.

1) Session participant rules

When a session participant receives a `<LogoutRequest>` message, the session participant must authenticate the message. If the sender is the authority that provided an assertion containing an authentication statement linked to the principal's current session, the session participant must invalidate the principal's session(s) referred to by the `<saml:BaseID>`, `<saml:NameID>`, or `<saml:EncryptedID>` element, and any `<SessionIndex>` elements supplied in the message. If no `<SessionIndex>` elements are supplied, then all sessions associated with the principal must be invalidated.

The session participant must apply the logout request message to any assertion that meets the following conditions, even if the assertion arrives after the logout request:

- The subject of the assertion **strongly matches** the `<saml:BaseID>`, `<saml:NameID>`, or `<saml:EncryptedID>` element in the `<LogoutRequest>`, as defined in 8.2.3.4.

- The SessionIndex attribute of one of the assertion's authentication statements matches one of the <SessionIndex> elements specified in the logout request, or the logout request contains no <SessionIndex> elements.
- The assertion would otherwise be valid, based on the time conditions specified in the assertion itself (in particular, the value of any specified NotOnOrAfter attributes in conditions or subject confirmation data).

The logout request has not yet expired (determined by examining the NotOnOrAfter attribute on the message).

NOTE – This rule is intended to prevent a situation in which a session participant receives a logout request targeted at a single, or multiple, assertion(s) (as identified by the <SessionIndex> element(s)) *before* it receives the actual – and possibly still valid – assertion(s) targeted by the logout request. It should honour the logout request until the logout request itself may be discarded (the NotOnOrAfter value on the request has been exceeded) or the assertion targeted by the logout request has been received and has been handled appropriately.

2) Session authority rules

When a session authority receives a <LogoutRequest> message, the session authority must authenticate the sender. If the sender is a session participant to which the session authority provided an assertion containing an authentication statement for the current session, then the session authority should do the following in the specified order:

– Send a <LogoutRequest> message to any session authority on behalf of whom the session authority proxied the principal's authentication, unless the second authority is the originator of the <LogoutRequest>.

– Send a <LogoutRequest> message to each session participant for which the session authority provided assertions in the current session, *other than* the originator of a current <LogoutRequest>.

– Terminate the principal's current session as specified by the <saml:BaseID>, <saml:NameID>, or <saml:EncryptedID> element, and any <SessionIndex> elements present in the logout request message.

If the session authority successfully terminates the principal's session with respect to itself, then it must respond to the original requester, if any, with a <LogoutResponse> message containing a top-level status code of urn:oasis:names:tc:SAML:2.0:status:Success. If it cannot do so, then it must respond with a <LogoutResponse> message containing a top-level status code indicating the error. Thus, the top-level status indicates the state of the logout operation only with respect to the session authority itself.

The session authority should attempt to contact each session participant using any applicable/usable protocol binding, even if one or more of these attempts fails or cannot be attempted (for example because the original request takes place using a protocol binding that does not enable the logout to be propagated to all participants).

In the event that not all session participants successfully respond to these <LogoutRequest> messages (or if not all participants can be contacted), then the session authority must include in its <LogoutResponse> message a second-level status code of urn:oasis:names:tc:SAML:2.0:status:PartialLogout to indicate that not all other session participants successfully responded with confirmation of the logout.

A session authority may initiate a logout for reasons other than having received a <LogoutRequest> from a session participant – these include, but are not limited to:

- If some timeout period was agreed out-of-band with an individual session participant, the session authority may send a <LogoutRequest> to that individual participant alone.
- An agreed global timeout period has been exceeded.
- The principal or some other trusted entity has requested logout of the principal directly at the session authority.
- The session authority has determined that the principal's credentials may have been compromised.

When constructing a logout request message, the session authority must set the value of the NotOnOrAfter attribute of the message to a time value, indicating an expiration time for the message, after which the logout request may be discarded by the recipient. This value should be set to a time value equal to or greater than the value of any NotOnOrAfter attribute specified in the assertion most recently issued as part of the targeted session (as indicated by the SessionIndex attribute on the logout request).

In addition to the values specified in 8.2.6.3 for the Reason attribute, the following values are also available for use by the session authority only:

```
urn:oasis:names:tc:SAML:2.0:logout:global-timeout
```

Specifies that the message is being sent because of the global session timeout interval period being exceeded.

`urn:oasis:names:tc:SAML:2.0:logout:sp-timeout`

Specifies that the message is being sent because a timeout interval period agreed between a participant and the session authority has been exceeded.

8.2.8 Name identifier mapping protocol

When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name identifier for the same principal in a particular format or federation namespace, it can send a request to the identity provider using this protocol.

For example, a service provider that wishes to communicate with another service provider with whom it does not share an identifier for the principal can use an identity provider that shares an identifier for the principal with both service providers to map from its own identifier to a new identifier, generally encrypted, with which it can communicate with the second service provider.

Regardless of the type of identifier involved, the mapped identifier should be encrypted into a `<saml:EncryptedID>` element unless a specific deployment dictates such protection is unnecessary.

8.2.8.1 Element <NameIDMappingRequest>

To request an alternate name identifier for a principal from an identity provider, a requester sends an `<NameIDMappingRequest>` message. This message has the complex type **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following elements:

– `<saml:BaseID>` or `<saml:NameID>` or `<saml:EncryptedID>` [Required]

The identifier and associated descriptive data that specify the principal as currently recognized by the requester and the responder. (For more information on this element, see 8.1.2.)

– `<NameIDPolicy>` [Required]

The requirements regarding the format and optional name qualifier for the identifier to be returned.

The message should be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the `<NameIDMappingRequest>` element and its **NameIDMappingRequestType** complex type:

```
<element name="NameIDMappingRequest"
type="samlp:NameIDMappingRequestType"/>
<complexType name="NameIDMappingRequestType">
       <complexContent>
              <extension base="samlp:RequestAbstractType">
                     <sequence>
                            <choice>
                                   <element ref="saml:BaseID"/>
                                   <element ref="saml:NameID"/>
                                   <element ref="saml:EncryptedID"/>
                            </choice>
                            <element ref="samlp:NameIDPolicy"/>
                     </sequence>
              </extension>
       </complexContent>
</complexType>
```

8.2.8.2 Element <NameIDMappingResponse>

The recipient of a `<NameIDMappingRequest>` message must respond with a `<NameIDMappingResponse>` message. This message has the complex type **NameIDMappingResponseType**, which extends **StatusResponseType** and adds the following element:

– `<saml:NameID>` or `<saml:EncryptedID>` [Required]

The identifier and associated attributes that specify the principal in the manner requested, usually in encrypted form. (For more information on this element, see 8.1.2.)

The message should be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the `<NameIDMappingResponse>` element and its **NameIDMappingResponseType** complex type:

```
<element name="NameIDMappingResponse"
type="samlp:NameIDMappingResponseType"/>
<complexType name="NameIDMappingResponseType">
       <complexContent>
              <extension base="samlp:StatusResponseType">
                     <choice>
                            <element ref="saml:NameID"/>
                            <element ref="saml:EncryptedID"/>
                     </choice>
              </extension>
       </complexContent>
</complexType>
```

8.2.8.3 Processing rules

If the responder does not recognize the principal identified in the request, it may respond with an error `<Status>` containing a second-level `<StatusCode>` of:

`urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal`

At the responder's discretion, the `urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy` status code may be returned to indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

All other processing rules associated with the underlying request and response messages must be observed.

8.3 SAML versioning

The SAML Recommendation set is versioned in two independent ways. Each is discussed in the following subclauses, along with processing rules for detecting and handling version differences. Also included are guidelines on when and why specific version information is expected to change in future revisions of SAML.

When version information is expressed as both a Major and Minor version, it is expressed in the form *Major.Minor*. The version number $Major_B.Minor_B$ is higher than the version number $Major_A.Minor_A$ if and only if:

$$(\mathrm{Major}_B > \mathrm{Major}_A) \text{ OR } ((\mathrm{Major}_B = \mathrm{Major}_A) \text{ AND } (\mathrm{Minor}_B > \mathrm{Minor}_A))$$

8.3.1 SAML specification set version

Each release of the SAML Recommendation will contain a major and minor version designation describing its relationship to earlier and later versions of the Recommendation. The version will be expressed in the content in the Recommendation. The overall size and scope of changes to the Recommendation will informally dictate whether a set of changes constitutes a major or minor revision. In general, if the cumulative changes is backward compatible with an earlier version then the new version will be a minor revision. Otherwise, the changes will constitute a major revision.

This Recommendation defines V2.0.

8.3.1.1 Schema version

As a non-normative documentation mechanism, any XML schema documents published as part of the specification set will contain a version attribute on the `<xs:schema>` element whose value is in the form *Major.Minor*, reflecting the specification set version in which it has been published. Validating implementations may use the attribute as a means of distinguishing which version of a schema is being used to validate messages, or to support multiple versions of the same logical schema.

8.3.1.2 SAML assertion version

The SAML `<Assertion>` element contains an attribute for expressing the major and minor version of the assertion in a string of the form *Major.Minor*. Each version of the SAML specification set will be construed so as to document the syntax, semantics, and processing rules of the assertions of the same version. That is, specification set version 1.0 describes assertion version 1.0, and so on.

There is explicitly NO relationship between the assertion version and the target XML namespace specified for the schema definitions for that assertion version.

The following processing rules apply:

- A SAML asserting party must not issue any assertion with an overall *Major.Minor* assertion version number not supported by the authority.
- A SAML relying party must not process any assertion with a major assertion version number not supported by the relying party.
- A SAML relying party may process or may reject an assertion whose minor assertion version number is higher than the minor assertion version number supported by the relying party. However, all assertions that share a major assertion version number must share the same general processing rules and semantics, and may be treated in a uniform way by an implementation. For example, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation may treat the assertion as a V1.0 assertion without ill effect.

8.3.1.3 SAML protocol version

The various SAML protocols' request and response elements contain an attribute for expressing the major and minor version of the request or response message using a string of the form *Major.Minor*. Each version of the SAML specification set will be construed so as to document the syntax, semantics, and processing rules of the protocol messages of the same version. That is, specification set version 1.0 describes request and response version V1.0, and so on.

There is explicitly NO relationship between the protocol version and the target XML namespace specified for the schema definitions for that protocol version.

The version numbers used in SAML protocol request and response elements will match for any particular revision of the SAML specification set.

1) Request version

The following processing rules apply to requests:

- A SAML requester should issue requests with the highest request version supported by both the SAML requester and the SAML responder.
- If the SAML requester does not know the capabilities of the SAML responder, then it should assume that the responder supports requests with the highest request version supported by the requester.
- A SAML requester must not issue a request message with an overall *Major.Minor* request version number matching a response version number that the requester does not support.
- A SAML responder must reject any request with a major request version number not supported by the responder.

A SAML responder may process or may reject any request whose minor request version number is higher than the highest supported request version that it supports. However, all requests that share a major request version number must share the same general processing rules and semantics, and may be treated in a uniform way by an implementation. That is, if a V1.1 request shares the syntax of a V1.0 request, a responder may treat the request message as a V1.0 request without ill effect.

2) Response version

The following processing rules apply to responses:

- A SAML responder must not issue a response message with a response version number higher than the request version number of the corresponding request message.
- A SAML responder must not issue a response message with a major response version number lower than the major request version number of the corresponding request message except to report the error `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- An error response resulting from incompatible SAML protocol versions must result in reporting a top-level `<StatusCode>` value of `urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`, and may result in reporting one of the following second-level values:
  - `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`;
  - `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow`; or
  - `urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated`.

3) Permissible version combinations

Assertions of a particular major version appear only in response messages of the same major version, as permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For example, a V1.1 assertion may appear in a V1.0 response message, and a V1.0 assertion in a V1.1 response message, if the appropriate assertion schema is referenced during namespace importation. But a V1.0 assertion must not appear in a V2.0 response message because they are of different major versions.

8.3.2 SAML namespace version

XML schema documents published as part of the specification set contain one or more target namespaces into which the type, element, and attribute definitions are placed. Each namespace is distinct from the others, and represents, in shorthand, the structural and syntactic definitions that make up that part of the specification.

The namespace URI references defined by the specification set will generally contain version information of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI must correspond to the major and minor version of the specification set in which the namespace is first introduced and defined. This information is not typically consumed by an XML processor, which treats the namespace opaquely, but is intended to communicate the relationship between the specification set and the namespaces it defines. This pattern is also followed by the SAML-defined URI-based identifiers that are listed in 8.7.

As a general rule, implementers can expect the namespaces and the associated schema definitions defined by a major revision of the specification set to remain valid and stable across minor revisions of the specification. New namespaces may be introduced, and when necessary, old namespaces replaced, but this is expected to be rare. In such cases, the older namespaces and their associated definitions should be expected to remain valid until a major specification set revision.

In general, maintaining namespace stability while adding or changing the content of a schema are competing goals. While certain design strategies can facilitate such changes, it is complex to predict how older implementations will react to any given change, making forward compatibility difficult to achieve. Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace stability. Except in special circumstances (for example, to correct major deficiencies or to fix errors), implementations should expect forward-compatible schema changes in minor revisions, allowing new messages to validate against older schemas.

Implementations should expect and be prepared to deal with new extensions and message types in accordance with the processing rules laid out for those types. Minor revisions may introduce new types that leverage the extension facilities as described in this Recommendation. Older implementations should reject such extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples include new query, statement, or condition types.

8.4 SAML and XML signature syntax and processing

SAML assertions and SAML protocol request and response messages may be signed, with the following benefits. An assertion signed by the asserting party supports assertion integrity, authentication of the asserting party to a SAML relying party, and, if the signature is based on the SAML authority's public-private key pair, non-repudiation of origin. A SAML protocol request or response message signed by the message originator supports message integrity, authentication of message origin to a destination, and, if the signature is based on the originator's public-private key pair, non-repudiation of origin.

A digital signature is not always required in SAML. For example, in some circumstances, signatures may be "inherited," such as when an unsigned assertion gains protection from a signature on the containing protocol response message. "Inherited" signatures should be used with care when the contained object (such as the assertion) is intended to have a non-transitory lifetime. The reason is that the entire context must be retained to allow validation, exposing the XML content and adding potentially unnecessary overhead. As another example, the SAML relying party or SAML requester may have obtained an assertion or protocol message from the SAML asserting party or SAML responder directly (with no intermediaries) through a secure channel, with the asserting party or SAML responder having authenticated to the relying party or SAML responder by some means other than a digital signature.

Many different techniques are available for "direct" authentication and secure channel establishment between two parties. The list includes TLS, HMAC, password-based mechanisms, and so on. In addition, the applicable security requirements depend on the communicating applications and the nature of the assertion or message transported. It is recommended that, in all other contexts, digital signatures be used for assertions and request and response messages. Specifically:

- A SAML assertion obtained by a SAML relying party from an entity other than the SAML asserting party should be signed by the SAML asserting party.
- A SAML protocol message arriving at a destination from an entity other than the originating sender should be signed by the sender.

- Profiles may specify alternative signature mechanisms such as S/MIME or signed Java objects that contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures are intended to be the primary SAML signature mechanism, but this Recommendation attempts to ensure compatibility with profiles that may require other mechanisms.
- Unless a profile specifies an alternative signature mechanism, any XML Digital Signatures must be enveloped.

8.4.1 Signing assertions

All SAML assertions may be signed using XML Signature. This is reflected in the assertion schema as described in clause 8.

8.4.2 Request/response signing

All SAML protocol request and response messages may be signed using XML Signature. This is reflected in the schema as described in Annex A.

8.4.3 Signature inheritance

A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>` or a request or response, which may be signed. When a SAML assertion does not contain a `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children, then the assertion can be considered to inherit the signature from the enclosing element. The resulting interpretation should be equivalent to the case where the assertion itself was signed with the same key and signature options.

Many SAML use cases involve SAML XML data enclosed within other protected data structures such as signed SOAP messages, S/MIME packages, and authenticated TLS connections. SAML profiles may define additional rules for interpreting SAML elements as inheriting signatures or other authentication information from the surrounding context, but no such inheritance should be inferred unless specifically identified by the profile.

8.4.4 XML signature profile

W3C XML Signature:2002 provides a general XML syntax for signing data with flexibility and many choices. This clause details constraints on these facilities so that SAML processors do not have to deal with the full generality of XML Signature processing. This usage makes specific use of the **xs:ID**-typed attributes present on the root elements to which signatures can apply, specifically the `ID` attribute on `<Assertion>` and the various request and response elements. These attributes are collectively referred to in this clause as the identifier attributes.

This profile only applies to the use of the `<ds:Signature>` elements found directly within SAML assertions, requests, and responses. Other profiles in which signatures appear elsewhere but apply to SAML content are free to define other approaches.

8.4.4.1 Signing formats and algorithms

XML Signature has three ways of relating a signature to a document: enveloping, enveloped and detached.

SAML assertions and protocols must use enveloped signatures when signing assertions and protocol messages. SAML processors should support the use of RSA signing and verification for public key operations in accordance with the algorithm identified in 6.4 of http://www.w3.org/2000/09/xmldsig#rsa-sha1.

8.4.4.2 References

SAML assertions and protocol messages must supply a value for the `ID` attribute on the root element of the assertion or protocol message being signed. The assertion's or protocol message's root element may or may not be the root element of the actual XML document containing the signed assertion or protocol message (e.g., it might be contained within a SOAP envelope).

Signatures must contain a single `<ds:Reference>` containing a same-document reference to the `ID` attribute value of the root element of the assertion or protocol message being signed. For example, if the `ID` attribute value is "foo", then the `URI` attribute in the `<ds:Reference>` element must be "#foo".

8.4.4.3 Canonicalization method

SAML implementations should use Exclusive Canonicalization, with or without comments, both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML messages embedded in an XML context can be verified independent of that context.

8.4.4.4 Transforms

Signatures in SAML messages should not contain transforms other than the enveloped signature transform (with the identifier http://www.w3.org/2000/09/xmldsig#enveloped-signature) or the exclusive canonicalization transforms (with the identifier http://www.w3.org/2001/10/xml-exc-c14n# or http://www.w3.org/2001/10/xml-exc-c14n#WithComments).

Verifiers of signatures may reject signatures that contain other transform algorithms as invalid. If they do not, verifiers must ensure that no content of the SAML message is excluded from the signature. This can be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by applying the transforms manually to the content and reverifying the result as consisting of the same SAML message.

8.4.4.5 KeyInfo

W3C Signature defines usage of the `<ds:KeyInfo>` element. SAML does not require the use of `<ds:KeyInfo>`, nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` may be absent.

8.4.4.6 Example

Following is an example of a signed response containing a signed assertion. Line breaks have been added for readability; the signatures are not valid and cannot be successfully verified.

```
<Response
  IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
  ID="_c7055387-af61-4fce-8b98-e2927324b306"
  xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <saml:Issuer>https://www.opensaml.org/IDP"</saml:Issuer>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
             <ds:SignedInfo>
                    <ds:CanonicalizationMethod
                          Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
                    <ds:SignatureMethod
                          Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
                    <ds:Reference URI="#_c7055387-af61-4fce-8b98-
e2927324b306">
                          <ds:Transforms>
                                 <ds:Transform

      Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                                 <ds:Transform

      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                                        <InclusiveNamespaces
PrefixList="#default saml ds xs xsi"

      xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                                 </ds:Transform>
                          </ds:Transforms>
                          <ds:DigestMethod

      Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

      <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
                    </ds:Reference>
             </ds:SignedInfo>
             <ds:SignatureValue>

      x/GyPbzmFEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5

      EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D
                    w6vKhaqled10BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
             </ds:SignatureValue>
             <ds:KeyInfo>
                    <ds:X509Data>
                          <ds:X509Certificate>
```

```
        MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakxCzAJBgNVBAYTAlVT

        MRIwEAYDVQQIEwlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT

        F1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ

        bmZvcm1hdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgQ0Eg

        LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx

        CzAJBgNVBAYTAlVTMREwDwYDVQQIEwhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy

        Ym9yMQ4wDAYDVQQKEwVVQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk

        dTEnMCUGCSqGSIb3DQEJARYYcm9vdEBzaGliMS5pbnRlcm5ldDIuZWR1MIGfMA0G

        CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj

        IHRYQgIv6IqaGG04eTcyVMhoekE0b45QgvBIaOAPSZBl13R6+KYiE7x4XAWIrCP+

        c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE

        pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq

        hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n

        qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX

        8I3bsbmRAUg4UP9hH6ABVq4KQKMknxu1xQxLhpR1ylGPdiowMNTrEG8cCx3w/w==
                                </ds:X509Certificate>
                        </ds:X509Data>
                </ds:KeyInfo>
        </ds:Signature>
        <Status>
                <StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
        </Status>
        <Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
                IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
                xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
                <Issuer>https://www.opensaml.org/IDP</Issuer>
                <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                        <ds:SignedInfo>
                                <ds:CanonicalizationMethod
                                        Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
                                <ds:SignatureMethod

        Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                                <ds:Reference URI="#_a75adf55-01d7-40cc-929f-
dbd8372ebdfc">
                                        <ds:Transforms>
                                                <ds:Transform

        Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                                                <ds:Transform

        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                                                        <InclusiveNamespaces
                                                                PrefixList="#default
saml ds xs xsi"

        xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                                                </ds:Transform>
                                        </ds:Transforms>
                                        <ds:DigestMethod

        Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

        <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
```

```
                                </ds:Reference>
                        </ds:SignedInfo>
                        <ds:SignatureValue>

        hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n

        7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtp3TD
                                MwuL/cBUj2OtBZOQMFn7jQ9YB7klIz3RqVL+wNmeWI4=
                        </ds:SignatureValue>
                        <ds:KeyInfo>
                                <ds:X509Data>
                                        <ds:X509Certificate>

        MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakxCzAJBgNVBAYTAlVT

        MRIwEAYDVQQIEwlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT

        F1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ

        bmZvcm1hdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgQ0Eg

        LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx

        CzAJBgNVBAYTAlVTMREwDwYDVQQIEwhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy

        Ym9yMQ4wDAYDVQQKEwVVQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk

        dTEnMCUGCSqGSIb3DQEJARYYcm9vdEBzaGliMS5pbnRlcm5ldDIuZWR1MIGfMA0G

        CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj

        IHRYQgIv6IqaGG04eTcyVMhoekE0b45QgvBIaOAPSZBl13R6+KYiE7x4XAWIrCP+

        c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE

        pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq

        hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n

        qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX

        8I3bsbmRAUg4UP9hH6ABVq4KQKMknxu1xQxLhpR1ylGPdiowMNTrEG8cCx3w/w==
                                        </ds:X509Certificate>
                                </ds:X509Data>
                        </ds:KeyInfo>
                </ds:Signature>
                <Subject>
                        <NameID
                                Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">
                                scott@example.org
                        </NameID>
                        <SubjectConfirmation
                                Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
                </Subject>
                <Conditions NotBefore="2003-04-17T00:46:02Z"
                                NotOnOrAfter="2003-04-17T00:51:02Z">
                        <AudienceRestriction>
                                <Audience>http://www.opensaml.org/SP</Audience>
                        </AudienceRestriction>
                </Conditions>
                <AuthnStatement AuthnInstant="2003-04-17T00:46:00Z">
                        <AuthnContext>
                                <AuthnContextClassRef>

        urn:oasis:names:tc:SAML:2.0:ac:classes:Password
                                </AuthnContextClassRef>
                        </AuthnContext>
                </AuthnStatement>
        </Assertion>
</Response>
```

8.5 SAML AND XML encryption syntax and processing

Encryption is used as the means to implement confidentiality. The most common motives for confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for competitive advantage or similar reasons. Confidentiality may also be required to ensure the effectiveness of some other security mechanism. For example, a secret password or key may be encrypted.

Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

- Communications confidentiality may be provided by mechanisms associated with a particular binding or profile. For example, the SOAP binding supports the use of TLS (see IETF RFC 2246) or SOAP message security mechanisms for confidentiality.
- A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>` element within `<SubjectConfirmationData>`, which permits keys or other secrets to be encrypted.
- An entire `<Assertion>` element may be encrypted, as described in 8.1.3.4.
- The `<BaseID>` or `<NameID>` element may be encrypted, as described in 8.1.2.4.
- An `<Attribute>` element may be encrypted, as described in 8.1.7.3.2.

8.5.1 General considerations

Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by use of XML encryption. Encrypted data and optionally one or more encrypted keys must replace the plaintext information in the same location within the XML instance. The `<EncryptedData>` element's Type attribute should be used and, if it is present, must have the value http://www.w3.org/2001/04/xmlenc#Element.

NOTE (informative) – PE30 (see OASIS PE:2006) suggests to replace one or more in the second line with zero or more.

Any of the algorithms defined for use with XML Encryption may be used to perform the encryption. The SAML schema is defined so that the inclusion of the encrypted data yields a valid instance.

8.5.2 Combining signatures and encryption

Use of XML encryption and XML signature may be combined. When an assertion is to be signed and encrypted, the following rules apply. A relying party must perform signature validation and decryption in the reverse order that signing and encryption were performed.

- When a signed `<Assertion>` element is encrypted, the signature must first be calculated and placed within the `<Assertion>` element before the element is encrypted.
- When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption must be performed first and then the signature calculated over the assertion or message containing the encrypted element.

8.6 SAML extensibility

SAML supports extensibility in a number of ways, including extending the assertion and protocol schemas. See the SAML Profiles clause in this Recommendation for information on how to define new profiles, which can be combined with extensions to put the SAML framework to new uses.

8.6.1 Schema extension

Elements in SAML schemas are blocked from substitution, which means that no SAML elements can serve as the head element of a substitution group. However, SAML types are not defined as final, so that all SAML types may be extended and restricted. As a practical matter, this means that extensions are typically defined only as types rather than elements, and are included in SAML instances by means of an `xsi:type` attribute.

The following subclauses discuss only elements and types that have been specifically designed to support extensibility.

8.6.1.1 Assertion schema extension

The SAML assertion schema is designed to permit separate processing of the assertion package and the statements it contains, if the extension mechanism is used for either part.

The following elements are intended specifically for use as extension points in an extension schema; their types are set to `abstract`, and are thus usable only as the base of a derived type:

- `<BaseID>` and **BaseIDAbstractType**
- `<Condition>` and **ConditionAbstractType**

- `<Statement>` and **StatementAbstractType**

The following constructs that are directly usable as part of SAML are particularly interesting targets for extension:

- `<AuthnStatement>` and **AuthnStatementType**
- `<AttributeStatement>` and **AttributeStatementType**
- `<AuthzDecisionStatement>` and **AuthzDecisionStatementType**
- `<AudienceRestriction>` and **AudienceRestrictionType**
- `<ProxyRestriction>` and **ProxyRestrictionType**
- `<OneTimeUse>` and **OneTimeUseType**

8.6.1.2 Protocol schema extension

The following SAML protocol elements are intended specifically for use as extension points in an extension schema; their types are set to abstract, and are thus usable only as the base of a derived type:

- `<Request>` and **RequestAbstractType**
- `<SubjectQuery>` and **SubjectQueryAbstractType**

The following constructs that are directly usable as part of SAML are particularly interesting targets for extension:

- `<AuthnQuery>` and **AuthnQueryType**
- `<AuthzDecisionQuery>` and **AuthzDecisionQueryType**
- `<AttributeQuery>` and **AttributeQueryType**
- **StatusResponseType**

8.6.2 Schema wildcard extension points

The SAML schemas use wildcard constructs in some locations to allow the use of elements and attributes from arbitrary namespaces, which serves as a built-in extension point without requiring an extension schema.

8.6.2.1 Assertion extension points

The following constructs in the assertion schema allow constructs from arbitrary namespaces within them:

- `<SubjectConfirmationData>`: Uses **xs:anyType**, which allows any sub-elements and attributes.
- `<AuthnContextDecl>`: Uses **xs:anyType**, which allows any sub-elements and attributes.
- `<AttributeValue>`: Uses **xs:anyType**, which allows any sub-elements and attributes.
- `<Advice>` and **AdviceType**: In addition to SAML-native elements, allows elements from other namespaces with lax schema validation processing.

The following construct in the assertion schema allows arbitrary global attributes:

- `<Attribute>` and **AttributeType**

8.6.2.2 Protocol extension points

The following construct in the protocol schema allows constructs from arbitrary namespaces within them:

- `<Extensions>` and **ExtensionsType**: Allows elements from other namespaces with lax schema validation processing.
- `<StatusDetail>` and **StatusDetailType**: Allows elements from other namespaces with lax schema validation processing.
- `<ArtifactResponse>` and **ArtifactResponseType**: Allows elements from any namespaces with lax schema validation processing. (It is specifically intended to carry a SAML request or response message element, however.)

8.6.3 Identifier extension

SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier formats, and defines some identifiers that may be used for these purposes; most are listed in 8.7. However, it is always possible to define additional URI-based identifiers for these purposes. It is recommended that these additional identifiers be defined in a formal profile of use. In no case should the meaning of a given URI used as such an identifier significantly change, or be used to mean two different things.

8.7 SAML-defined identifiers

The following subclauses define URI-based identifiers for common resource access actions, subject name identifier formats, and attribute name formats.

Where possible an existing URN is used to specify a protocol. In the case of IETF protocols, the URN of the most current RFC that specifies the protocol is used. URI references created specifically for SAML have one of the following stems, according to the specification set version in which they were first introduced:

```
urn:oasis:names:tc:SAML:1.0:
urn:oasis:names:tc:SAML:1.1:
urn:oasis:names:tc:SAML:2.0:
```

This Recommendation introduces the last stem.

8.7.1 Action namespace identifiers

The following identifiers may be used in the `Namespace` attribute of the `<Action>` element to refer to common sets of actions to perform on resources.

8.7.1.1 Read/Write/Execute/Delete/Control

URI: `urn:oasis:names:tc:SAML:1.0:action:rwedc`

Defined actions: `Read Write Execute Delete Control`

These actions are interpreted as follows:

`Read`: The subject may read the resource.

`Write`: The subject may modify the resource.

`Execute`: The subject may execute the resource.

`Delete`: The subject may delete the resource.

`Control`: The subject may specify the access control policy for the resource.

8.7.1.2 Read/Write/Execute/Delete/Control with negation

URI: `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation`

Defined actions: `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

The actions specified in 8.7.1.1 are interpreted in the same manner described there. Actions prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated permission is denied. Thus a subject described as being authorized to perform the action ~Read is affirmatively denied read permission.

A SAML authority must not authorize both an action and its negated form.

8.7.1.3 Get/Head/Put/Post

URI: `urn:oasis:names:tc:SAML:1.0:action:ghpp`

Defined actions: `GET HEAD PUT POST`

These actions bind to the corresponding HTTP operations. For example, a subject authorized to perform the `GET` action on a resource is authorized to retrieve it.

The `GET` and `HEAD` actions loosely correspond to the conventional read permission and the `PUT` and `POST` actions to the write permission. The correspondence is not exact however since an HTTP GET operation may cause data to be modified and a POST operation may cause modification to a resource other than the one specified in the request. For this reason, a separate Action URI reference specifier is provided.

8.7.1.4 UNIX file permissions

URI: `urn:oasis:names:tc:SAML:1.0:action:unix`

The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

The action string is a four-digit numeric code:

extended user group world

Where the *extended* access permission has the value:

+2 if sgid is set

+4 if suid is set

The *user group* and *world* access permissions have the value:

+1 if execute permission is granted

+2 if write permission is granted

+4 if read permission is granted

For example, 0754 denotes the UNIX file access permission: user read, write, and execute; group read and execute; and world read.

8.7.2 Attribute name format identifiers

The following identifiers may be used in the `NameFormat` attribute defined on the **AttributeType** complex type to refer to the classification of the attribute name for purposes of interpreting the name.

8.7.2.1 Unspecified

URI: `urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified`

The interpretation of the attribute name is left to individual implementations.

8.7.2.2 URI reference

URI: `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`

The attribute name follows the convention for URI references, for example as used in XACML attribute identifiers. The interpretation of the URI content or naming scheme is application-specific. See clause 11 for attribute profiles that make use of this identifier.

8.7.2.3 Basic

URI: `urn:oasis:names:tc:SAML:2.0:attrname-format:basic`

The class of strings acceptable as the attribute name must be drawn from the set of values belonging to the primitive type **xs:Name**, as defined in W3C XML Datatypes, 3.3.6. See clause 13 for attribute profiles that make use of this identifier.

8.7.3 Name identifier format identifiers

The following identifiers may be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or `<Issuer>` elements (see 8.1.2) to refer to common formats for the content of the elements and the associated processing rules, if any.

NOTE – Several identifiers that were deprecated in SAML V1.1 have been removed for SAML V2.0.

8.7.3.1 Unspecified

URI: `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`

The interpretation of the content of the element is left to individual implementations.

8.7.3.2 Email address

URI: `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`

Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as defined in IETF RFC 2822, 3.4.1. An addr-spec has the form local-part@domain. Note that an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in parentheses) after it, and is not surrounded by "<" and ">".

8.7.3.3 X.509 subject name

URI: `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`

Indicates that the content of the element is in the form specified for the contents of the `<ds:X509SubjectName>` element in W3C Signature. Implementers should note that W3C XML Signature specifies encoding rules for X.509 subject names that differ from the rules given in IETF RFC 2253.

8.7.3.4 Windows domain qualified name

URI: `urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName`

Indicates that the content of the element is a Windows domain qualified name. A Windows domain qualified user name is a string of the form "DomainName\UserName". The domain name and "\" separator may be omitted.

8.7.3.5 Kerberos principal name

URI: `urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos`

Indicates that the content of the element is in the form of a Kerberos principal name using the format `name[/instance]@REALM`. The syntax, format and characters allowed for the name, instance, and realm are described in IETF RFC 1510.

8.7.3.6 Entity identifier

URI: `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`

Indicates that the content of the element is the identifier of an entity that provides SAML-based services (such as a SAML authority, requester, or responder) or is a participant in SAML profiles (such as a service provider supporting the browser SSO profile). Such an identifier can be used in the `<Issuer>` element to identify the issuer of a SAML request, response, or assertion, or within the `<NameID>` element to make assertions about system entities that can issue SAML requests, responses, and assertions. It can also be used in other elements and attributes whose purpose is to identify a system entity in various protocol exchanges.

The syntax of such an identifier is a URI of not more than 1024 characters in length. It is recommended that a system entity use a URL containing its own domain name to identify itself.

The `NameQualifier`, `SPNameQualifier`, and `SPProvidedID` attributes must be omitted.

8.7.3.7 Persistent identifier

URI: `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`

Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to an identity provider and a service provider or affiliation of service providers. Persistent name identifiers generated by identity providers must be constructed using pseudo-random values that have no discernible correspondence with the subject's actual identifier (for example, username). The intent is to create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities. Persistent name identifier values must not exceed a length of 256 characters.

The element's `NameQualifier` attribute, if present, must contain the unique identifier of the identity provider that generated the identifier (see 8.7.3.6). It may be omitted if the value can be derived from the context of the message containing the element, such as the issuer of a protocol message or an assertion containing the identifier in its subject. A different system entity might later issue its own protocol message or assertion containing the identifier; the `NameQualifier` attribute does not change in this case, but must continue to identify the entity that originally created the identifier (and must not be omitted in such a case).

The element's `SPNameQualifier` attribute, if present, must contain the unique identifier of the service provider or affiliation of providers for whom the identifier was generated (see 8.7.3.6). It may be omitted if the element is contained in a message intended only for consumption directly by the service provider, and the value would be the unique identifier of that service provider.

The element's `SPProvidedID` attribute must contain the alternative identifier of the principal most recently set by the service provider or affiliation, if any (see 8.2.6). If no such identifier has been established, then the attribute must be omitted.

Persistent identifiers are intended as a privacy protection mechanism; as such they must not be shared in clear text with providers other than the providers that have established the shared identifier. Furthermore, they must not appear in log files or similar locations without appropriate controls and protections. Deployments without such requirements are free to use other kinds of identifiers in their SAML exchanges, but must not overload this format with persistent but non-opaque values

While persistent identifiers are typically used to reflect an account linking relationship between a pair of providers, a service provider is not obligated to recognize or make use of the long term nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly different and does not affect the behaviour of the identity provider or any processing rules specific to persistent identifiers in the protocols defined in this Recommendation.

`NameQualifier` and `SPNameQualifier` attributes indicate directionality of creation, but not its use. If a persistent identifier is created by a particular identity provider, the `NameQualifier` attribute value is permanently established at that time. If a service provider that receives such an identifier takes on the role of an identity provider and issues its own assertion containing that identifier, the `NameQualifier` attribute value does not change (and would of course not be omitted). It might alternatively choose to create its own persistent identifier to represent the principal and link the two values. This is a deployment decision.

8.7.3.8 Transient identifier

URI: `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`

Indicates that the content of the element is an identifier with transient semantics and should be treated as an opaque and temporary value by the relying party. Transient identifier values must be generated in accordance with the rules for SAML identifiers (see 7.4), and must not exceed a length of 256 characters.

The `NameQualifier` and `SPNameQualifier` attributes may be used to signify that the identifier represents a transient and temporary pair-wise identifier. In such a case, they may be omitted in accordance with the rules specified in 8.7.3.7.

8.7.4 Consent identifiers

The following identifiers may be used in the `Consent` attribute defined on the **RequestAbstractType** and **StatusResponseType** complex types to communicate whether a principal gave consent, and under what conditions, for the message.

8.7.4.1 Unspecified

URI: `urn:oasis:names:tc:SAML:2.0:consent:unspecified`

No claim as to principal consent is being made.

8.7.4.2 Obtained

URI: `urn:oasis:names:tc:SAML:2.0:consent:obtained`

Indicates that a principal's consent has been obtained by the issuer of the message.

8.7.4.3 Prior

URI: `urn:oasis:names:tc:SAML:2.0:consent:prior`

Indicates that a principal's consent has been obtained by the issuer of the message at some point prior to the action that initiated the message.

8.7.4.4 Implicit

URI: `urn:oasis:names:tc:SAML:2.0:consent:current-implicit`

Indicates that a principal's consent has been implicitly obtained by the issuer of the message during the action that initiated the message, as part of a broader indication of consent. Implicit consent is typically more proximal to the action in time and presentation than prior consent, such as part of a session of activities.

8.7.4.5 Explicit

URI: `urn:oasis:names:tc:SAML:2.0:consent:current-explicit`

Indicates that a principal's consent has been explicitly obtained by the issuer of the message during the action that initiated the message.

8.7.4.6 Unavailable

URI: `urn:oasis:names:tc:SAML:2.0:consent:unavailable`

Indicates that the issuer of the message did not obtain consent.

8.7.4.7 Inapplicable

URI: `urn:oasis:names:tc:SAML:2.0:consent:inapplicable`

Indicates that the issuer of the message does not believe that they need to obtain or report consent.

9 SAML metadata

SAML profiles require agreements between system entities regarding identifiers, binding support and endpoints, certificates and keys, and so forth. This clause defines an extensible metadata format for SAML system entities, organized by roles that reflect SAML profiles. Such roles include that of SSO Identity Provider, SSO Service Provider, Affiliation, Attribute Authority, Attribute Requester, and Policy Decision Point.

9.1 Metadata

SAML metadata is organized around an extensible collection of roles representing common combinations of SAML protocols and profiles supported by system entities. Each role is described by an element derived from the extensible base type of `RoleDescriptor`. Such descriptors are in turn collected into the `<EntityDescriptor>` container element, the primary unit of SAML metadata. An entity might alternatively represent an affiliation of other entities, such as an affiliation of service providers. The `<AffiliationDescriptor>` is provided for this purpose.

Such descriptors may in turn be aggregated into nested groups using the `<EntitiesDescriptor>` element.

A variety of security mechanisms for establishing the trustworthiness of metadata can be supported, particularly with the ability to individually sign most of the elements defined in this Recommendation.

When elements with a parent/child relationship contain common attributes, such as caching or expiration information, the parent element takes precedence.

> NOTE – As a general matter, SAML metadata is not to be taken as an authoritative statement about the capabilities or options of a given system entity. That is, while it should be accurate, it need not be exhaustive. The omission of a particular option does not imply that it is or is not unsupported, merely that it is not claimed. As an example, a SAML attribute authority might support any number of attributes not named in an `<AttributeAuthorityDescriptor>`. Omissions might reflect privacy or any number of other considerations. Conversely, indicating support for a given attribute does not imply that a given requester can or will receive it.

9.1.1 Namespaces

SAML Metadata uses the following namespace:

```
urn:oasis:names:tc:SAML:2.0:metadata
```

This Recommendation uses the namespace prefix `md:` to refer to the namespace above.

The following schema fragment illustrates the use of namespaces in SAML metadata documents:

```
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
20020212/xmldsig-core-schema.xsd"/>
    <import namespace="http://www.w3.org/2001/04/xmlenc#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-
20021210/xenc-schema.xsd"/>
    <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
```

```
        schemaLocation="saml-schema-assertion-2.0.xsd"/>
    <import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd"/>
    <annotation>
        <documentation>
            Document identifier: saml-schema-metadata-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
              V2.0 (March, 2005):
                Schema for SAML metadata, first published in SAML 2.0.
        </documentation>
    </annotation>
…
</schema>
```

9.1.2 Common types

This clause defines several Metadata types to be used in defining elements and attributes.

9.1.2.1 Simple type entityIDType

The simple type **entityIDType** restricts the XML schema data type **anyURI** to a maximum length of 1024 characters. **entityIDType** is used as a unique identifier for SAML entities. See also clause 8.7.3.6. An identifier of this type must be unique across all entities that interact within a given deployment. The use of a URI and holding to the rule that a single URI must not refer to different entities satisfies this requirement.

The following schema fragment defines the **entityIDType** simple type:

```
<simpleType name="entityIDType">
      <restriction base="anyURI">
             <maxLength value="1024"/>
      </restriction>
</simpleType>
```

9.1.2.2 Complex type EndpointType

The complex type **EndpointType** describes a SAML protocol binding endpoint at which a SAML entity can be sent protocol messages. Various protocol or profile-specific metadata elements are bound to this type. It consists of the following attributes:

– `Binding` [Required]

  A required attribute that specifies the SAML binding supported by the endpoint. Each binding is assigned a URI to identify it.

– `Location` [Required]

  A required URI attribute that specifies the location of the endpoint. The allowable syntax of this URI depends on the protocol binding.

– `ResponseLocation` [Optional]

  Optionally specifies a different location to which response messages sent as part of the protocol or profile should be sent. The allowable syntax of this URI depends on the protocol binding.

The `ResponseLocation` attribute is used to enable different endpoints to be specified for receiving request and response messages associated with a protocol or profile, not as a means of load-balancing or redundancy (multiple elements of this type can be included for this purpose). When a role contains an element of this type pertaining to a protocol or profile for which only a single type of message (request or response) is applicable, then the `ResponseLocation` attribute is unused.

NOTE (informative) – PE41 (see OASIS PE:2006) clarifies the above paragraph by appending the following sentence to the above text:

> If the `ResponseLocation` attribute is omitted, any response messages associated with a protocol or profile may be assumed to be handled at the URI indicated by the `Location` attribute.

In most contexts, elements of this type appear in unbounded sequences in the schema. This is to permit a protocol or profile to be offered by an entity at multiple endpoints, usually with different protocol bindings, allowing the metadata consumer to choose an appropriate endpoint for its needs. Multiple endpoints might also offer "client-side" load-balancing or failover, particular in the case of a synchronous protocol binding.

This element also permits the use of arbitrary elements and attributes defined in a non-SAML namespace. Any such content must be namespace-qualified.

The following schema fragment defines the **EndpointType** complex type:

```
<complexType name="EndpointType">
	<sequence>
		<any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
	</sequence>
	<attribute name="Binding" type="anyURI" use="required"/>
	<attribute name="Location" type="anyURI" use="required"/>
	<attribute name="ResponseLocation" type="anyURI" use="optional"/>
	<anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

9.1.2.3 Complex type IndexedEndpointType

The complex type **IndexedEndpointType** extends **EndpointType** with a pair of attributes to permit the indexing of otherwise identical endpoints so that they can be referenced by protocol messages. It consists of the following additional attributes:

– `index` [Required]

A required attribute that assigns a unique integer value to the endpoint so that it can be referenced in a protocol message. The index value need only be unique within a collection of like elements contained within the same parent element (i.e., they need not be unique across the entire instance).

– `isDefault` [Optional]

An optional boolean attribute used to designate the default endpoint among an indexed set. If omitted, the value is assumed to be false.

In any such sequence of like endpoints based on this type, the default endpoint is the first such endpoint with the `isDefault` attribute set to `true`. If no such endpoints exist, the default endpoint is the first such endpoint without the `isDefault` attribute set to `false`. If no such endpoints exist, the default endpoint is the first element in the sequence.

NOTE (informative) – PE37 (see OASIS PE:2006) suggests to clarify the above paragraph with:

> In any such sequence of indexed endpoints that share a common element name and namespace (i.e., all instances of `<md:AssertionConsumerService>` within a role), the default endpoint is the first such endpoint with the `isDefault` attribute set to `true`. If no such endpoints exist, the default endpoint is the first such endpoint without the `isDefault` attribute set to `false`. If no such endpoints exist, the default endpoint is the first element in the sequence.

The following schema fragment defines the **IndexedEndpointType** complex type:

```
<complexType name="IndexedEndpointType">
	<complexContent>
		<extension base="md:EndpointType">
			<attribute name="index" type="unsignedShort"
use="required"/>
			<attribute name="isDefault" type="boolean"
use="optional"/>
		</extension>
	</complexContent>
</complexType>
```

9.1.2.4 Complex type localizedNameType

The **localizedNameType** complex type extends a string-valued element with a standard XML language attribute. The following schema fragment defines the **localizedNameType** complex type:

```
<complexType name="localizedNameType">
	<simpleContent>
		<extension base="string">
			<attribute ref="xml:lang" use="required"/>
		</extension>
	</simpleContent>
</complexType>
```

9.1.2.5 Complex type localizedURIType

The **localizedURIType** complex type extends a URI-valued element with a standard XML language attribute.

The following schema fragment defines the **localizedURIType** complex type:

```
<complexType name="localizedURIType">
    <simpleContent>
             <extension base="anyURI">
                    <attribute ref="xml:lang" use="required"/>
             </extension>
       </simpleContent>
</complexType>
```

9.1.3 Root elements

A SAML metadata instance describes either a single entity or multiple entities. In the former case, the root element must be `<EntityDescriptor>`. In the latter case, the root element must be `<EntitiesDescriptor>`.

9.1.3.1 Element <EntitiesDescriptor>

The `<EntitiesDescriptor>` element contains the metadata for an optionally named group of SAML entities. Its **EntitiesDescriptorType** complex type contains a sequence of `<EntityDescriptor>` elements, `<EntitiesDescriptor>` elements, or both:

– `ID` [Optional]

  A document-unique identifier for the element, typically used as a reference point when signing.

– `validUntil` [Optional]

  Optional attribute indicates the expiration time of the metadata contained in the element and any contained elements.

– `cacheDuration` [Optional]

  Optional attribute indicates the maximum length of time a consumer should cache the metadata contained in the element and any contained elements.

– `Name` [Optional]

  A string name that identifies a group of SAML entities in the context of some deployment.

– `<ds:Signature>` [Optional]

  An XML signature that authenticates the containing element and its contents, as described in clause 8.

– `<Extensions>` [Optional]

  This contains optional metadata extensions that are agreed upon between a metadata publisher and consumer. Extension elements must be namespace-qualified by a non-SAML-defined namespace.

– `<EntitiesDescriptor>` or `<EntityDescriptor>` [One or More]

  Contains the metadata for one or more SAML entities, or a nested group of additional metadata.

When used as the root element of a metadata instance, this element must contain either a `validUntil` or `cacheDuration` attribute. It is recommended that only the root element of a metadata instance contain either attribute.

The following schema fragment defines the `<EntitiesDescriptor>` element and its **EntitiesDescriptorType** complex type:

```
<element name="EntitiesDescriptor" type="md:EntitiesDescriptorType"/>
<complexType name="EntitiesDescriptorType">
       <sequence>
             <element ref="ds:Signature" minOccurs="0"/>
             <element ref="md:Extensions" minOccurs="0"/>
             <choice minOccurs="1" maxOccurs="unbounded">
                    <element ref="md:EntityDescriptor"/>
                    <element ref="md:EntitiesDescriptor"/>
             </choice>
       </sequence>
       <attribute name="validUntil" type="dateTime" use="optional"/>
       <attribute name="cacheDuration" type="duration" use="optional"/>
       <attribute name="ID" type="ID" use="optional"/>
       <attribute name="Name" type="string" use="optional"/>
</complexType>
```

```
<element name="Extensions" type="md:ExtensionsType"/>
<complexType final="#all" name="ExtensionsType">
      <sequence>
            <any namespace="##other" processContents="lax"
maxOccurs="unbounded"/>
      </sequence>
</complexType>
```

9.1.3.2 Element <EntityDescriptor>

The `<EntityDescriptor>` element specifies metadata for a single SAML entity. A single entity may act in many different roles in the support of multiple profiles. This Recommendation directly supports the following concrete roles as well as the abstract `<RoleDescriptor>` element for extensibility:

- SSO Identity Provider;
- SSO Service Provider;
- Authentication Authority;
- Attribute Authority;
- Policy Decision Point;
- Affiliation.

Its **EntityDescriptorType** complex type consists of the following elements and attributes:

– `entityID` [Required]

Specifies the unique identifier of the SAML entity whose metadata is described by the element's contents.

– `ID` [Optional]

A document-unique identifier for the element, typically used as a reference point when signing.

– `validUntil` [Optional]

Optional attribute indicates the expiration time of the metadata contained in the element and any contained elements.

– `cacheDuration` [Optional]

Optional attribute indicates the maximum length of time a consumer should cache the metadata contained in the element and any contained elements.

– `<ds:Signature>` [Optional]

An XML signature that authenticates the containing element and its contents.

– `<Extensions>` [Optional]

This contains optional metadata extensions that are agreed upon between a metadata publisher and consumer. Extension elements must be namespace-qualified by a non-SAML-defined namespace.

– `<RoleDescriptor>`, `<IDPSSODescriptor>`, `<SPSSODescriptor>`, `<AuthnAuthorityDescriptor>`, `<AttributeAuthorityDescriptor>`, `<PDPDescriptor>` [One or More]; or

– `<AffiliationDescriptor>` [Required]

The primary content of the element is either a sequence of one or more role descriptor elements, or a specialized descriptor that defines an affiliation.

– `<Organization>` [Optional]

Optional element identifying the organization responsible for the SAML entity described by the element.

– `<ContactPerson>` [Zero or More]

Optional sequence of elements identifying various kinds of contact personnel.

– `<AdditionalMetadataLocation>` [Zero or More]

Optional sequence of namespace-qualified locations where additional metadata exists for the SAML entity. This may include metadata in alternate formats or describing adherence to other non-SAML Recommendations.

Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

When used as the root element of a metadata instance, this element must contain either a `validUntil` or `cacheDuration` attribute. It is recommended that only the root element of a metadata instance contain either attribute.

It is recommended that if multiple role descriptor elements of the same type appear, that they do not share overlapping `protocolSupportEnumeration` values. Selecting from among multiple role descriptor elements of the same type that do share a `protocolSupportEnumeration` value is undefined within this Recommendation, but may be defined by metadata profiles, possibly through the use of other distinguishing extension attributes.

The following schema fragment defines the `<EntityDescriptor>` element and its **EntityDescriptorType** complex type:

```
<element name="EntityDescriptor" type="md:EntityDescriptorType"/>
<complexType name="EntityDescriptorType">
        <sequence>
                <element ref="ds:Signature" minOccurs="0"/>
                <element ref="md:Extensions" minOccurs="0"/>
                <choice>
                        <choice maxOccurs="unbounded">
                                <element ref="md:RoleDescriptor"/>
                                <element ref="md:IDPSSODescriptor"/>
                                <element ref="md:SPSSODescriptor"/>
                                <element ref="md:AuthnAuthorityDescriptor"/>
                                <element ref="md:AttributeAuthorityDescriptor"/>
                                <element ref="md:PDPDescriptor"/>
                        </choice>
                        <element ref="md:AffiliationDescriptor"/>
                </choice>
                <element ref="md:Organization" minOccurs="0"/>
                <element ref="md:ContactPerson" minOccurs="0"
maxOccurs="unbounded"/>
                <element ref="md:AdditionalMetadataLocation" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
        <attribute name="entityID" type="md:entityIDType" use="required"/>
        <attribute name="validUntil" type="dateTime" use="optional"/>
        <attribute name="cacheDuration" type="duration" use="optional"/>
        <attribute name="ID" type="ID" use="optional"/>
        <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

9.1.3.2.1 Element <Organization>

The `<Organization>` element specifies basic information about an organization responsible for a SAML entity or role. The use of this element is always optional. Its content is informative in nature and does not directly map to any core SAML elements or attributes. Its **OrganizationType** complex type consists of the following elements:

– `<Extensions>` [Optional]

  This contains optional metadata extensions that are agreed upon between a metadata publisher and consumer. Extensions must not include global (non-namespace-qualified) elements or elements qualified by a SAML-defined namespace within this element.

– `<OrganizationName>` [One or More]

  One or more language-qualified names that may or may not be suitable for human consumption.

– `<OrganizationDisplayName>` [One or More]

  One or more language-qualified names that are suitable for human consumption.

– `<OrganizationURL>` [One or More]

  One or more language-qualified URIs that specify a location to which to direct a user for additional information. The language qualifier refers to the content of the material at the specified location.

Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

The following schema fragment defines the `<Organization>` element and its **OrganizationType** complex type:

```
<element name="Organization" type="md:OrganizationType"/>
<complexType name="OrganizationType">
       <sequence>
              <element ref="md:Extensions" minOccurs="0"/>
              <element ref="md:OrganizationName" maxOccurs="unbounded"/>
              <element ref="md:OrganizationDisplayName"
maxOccurs="unbounded"/>
              <element ref="md:OrganizationURL" maxOccurs="unbounded"/>
       </sequence>
       <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
<element name="OrganizationName" type="md:localizedNameType"/>
<element name="OrganizationDisplayName" type="md:localizedNameType"/>
<element name="OrganizationURL" type="md:localizedURIType"/>
```

9.1.3.2.2 Element <ContactPerson>

The `<ContactPerson>` element specifies basic contact information about a person responsible in some capacity for a SAML entity or role. The use of this element is always optional. Its content is informative in nature and does not directly map to any core SAML elements or attributes. Its **ContactType** complex type consists of the following elements and attributes:

– `contactType` [Required]

Specifies the type of contact using the **ContactTypeType** enumeration. The possible values are technical, support, administrative, billing, and other.

– `<Extensions>` [Optional]

This contains optional metadata extensions that are agreed upon between a metadata publisher and consumer. Extension elements must be namespace-qualified by a non-SAML-defined namespace.

– `<Company>` [Optional]

Optional string element that specifies the name of the company for the contact person.

– `<GivenName>` [Optional]

Optional string element that specifies the given (first) name of the contact person.

– `<SurName>` [Optional]

Optional string element that specifies the surname of the contact person.

– `<EmailAddress>` [Zero or More]

Zero or more elements containing mailto: URIs representing e-mail addresses belonging to the contact person.

– `<TelephoneNumber>` [Zero or More]

Zero or more string elements specifying a telephone number of the contact person.

Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

The following schema fragment defines the `<ContactPerson>` element and its **ContactType** complex type:

```
<element name="ContactPerson" type="md:ContactType"/>
<complexType name="ContactType">
       <sequence>
              <element ref="md:Extensions" minOccurs="0"/>
              <element ref="md:Company" minOccurs="0"/>
              <element ref="md:GivenName" minOccurs="0"/>
              <element ref="md:SurName" minOccurs="0"/>
              <element ref="md:EmailAddress" minOccurs="0"
maxOccurs="unbounded"/>
              <element ref="md:TelephoneNumber" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
       <attribute name="contactType" type="md:ContactTypeType"
use="required"/>
       <anyAttribute namespace="##other" processContents="lax"/>
```

```
</complexType>
<element name="Company" type="string"/>
<element name="GivenName" type="string"/>
<element name="SurName" type="string"/>
<element name="EmailAddress" type="anyURI"/>
<element name="TelephoneNumber" type="string"/>
<simpleType name="ContactTypeType">
	<restriction base="string">
		<enumeration value="technical"/>
		<enumeration value="support"/>
		<enumeration value="administrative"/>
		<enumeration value="billing"/>
		<enumeration value="other"/>
	</restriction>
</simpleType>
```

9.1.3.2.3 Element <AdditionalMetadataLocation>

The `<AdditionalMetadataLocation>` element is a namespace-qualified URI that specifies where additional XML-based metadata may exist for a SAML entity. Its **AdditionalMetadataLocationType** complex type extends the **anyURI** type with a namespace attribute (also of type **anyURI**). This required attribute must contain the XML namespace of the root element of the instance document found at the specified location.

The following schema fragment defines the `<AdditionalMetadataLocation>` element and its **AdditionalMetadataLocationType** complex type:

```
<element name="AdditionalMetadataLocation"
type="md:AdditionalMetadataLocationType"/>
<complexType name="AdditionalMetadataLocationType">
	<simpleContent>
		<extension base="anyURI">
			<attribute name="namespace" type="anyURI"
use="required"/>
		</extension>
	</simpleContent>
</complexType>
```

9.1.4 Role descriptor elements

The elements in this clause make up the bulk of the operational support component of the metadata. Each element (save for the abstract one) defines a specific collection of operational behaviours in support of SAML profiles.

9.1.4.1 Element <RoleDescriptor>

The `<RoleDescriptor>` element is an abstract extension point that contains common descriptive information intended to provide processing commonality across different roles. New roles can be defined by extending its abstract **RoleDescriptorType** complex type, which contains the following elements and attributes:

– `ID` [Optional]

A document-unique identifier for the element, typically used as a reference point when signing.

– `validUntil` [Optional]

Optional attribute indicates the expiration time of the metadata contained in the element and any contained elements.

– `cacheDuration` [Optional]

Optional attribute indicates the maximum length of time a consumer should cache the metadata contained in the element and any contained elements.

– `protocolSupportEnumeration` [Required]

A whitespace-delimited set of URIs that identify the set of protocol specifications supported by the role element. For SAML V2.0 entities, this set must include the SAML protocol namespace URI, `urn:oasis:names:tc:SAML:2.0:protocol`. Subsequent SAML Recommendations might share the same namespace URI, but should provide alternate "protocol support" identifiers to ensure discrimination when necessary.

– `errorURL` [Optional]

Optional URI attribute that specifies a location to direct a user for problem resolution and additional support related to this role.

– `<ds:Signature>` [Optional]

An XML signature that authenticates the containing element and its contents.

– `<Extensions>` [Optional]

This contains optional metadata extensions that are agreed upon between a metadata publisher and consumer. Extension elements must be namespace-qualified by a non-SAML-defined namespace.

– `<KeyDescriptor>` [Zero or More]

Optional sequence of elements that provides information about the cryptographic keys that the entity uses when acting in this role.

– `<Organization>` [Optional]

Optional element specifies the organization associated with this role. Identical to the element used within the `<EntityDescriptor>` element.

– `<ContactPerson>` [Zero or More]

Optional sequence of elements specifying contacts associated with this role. Identical to the element used within the `<EntityDescriptor>` element.

Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

The following schema fragment defines the `<RoleDescriptor>` element and its **RoleDescriptorType** complex type:

```
<element name="RoleDescriptor" type="md:RoleDescriptorType"/>
<complexType name="RoleDescriptorType" abstract="true">
       <sequence>
              <element ref="ds:Signature" minOccurs="0"/>
              <element ref="md:Extensions" minOccurs="0"/>
              <element ref="md:KeyDescriptor" minOccurs="0"
maxOccurs="unbounded"/>
              <element ref="md:Organization" minOccurs="0"/>
              <element ref="md:ContactPerson" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
       <attribute name="ID" type="ID" use="optional"/>
       <attribute name="validUntil" type="dateTime" use="optional"/>
       <attribute name="cacheDuration" type="duration" use="optional"/>
       <attribute name="protocolSupportEnumeration" type="md:anyURIListType"
use="required"/>
       <attribute name="errorURL" type="anyURI" use="optional"/>
       <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
<simpleType name="anyURIListType">
       <list itemType="anyURI"/>
</simpleType>
```

9.1.4.1.1 Element <KeyDescriptor>

The `<KeyDescriptor>` element provides information about the cryptographic key(s) that an entity uses to sign data or receive encrypted keys, along with additional cryptographic details. Its **KeyDescriptorType** complex type consists of the following elements and attributes:

– `use` [Optional]

Optional attribute specifying the purpose of the key being described. Values are drawn from the **KeyTypes** enumeration, and consist of the values `encryption` and `signing`.

– `<ds:KeyInfo>` [Required]

Optional element that directly or indirectly identifies a key. See W3C XML Signature for additional details on the use of this element.

– `<EncryptionMethod>` [Zero or More]

Optional element specifying an algorithm and algorithm-specific settings supported by the entity. The exact content varies based on the algorithm supported. See W3C Encryption for the definition of this element's **xenc:EncryptionMethodType** complex type.

The following schema fragment defines the `<KeyDescriptor>` element and its **KeyDescriptorType** complex type:

```
<element name="KeyDescriptor" type="md:KeyDescriptorType"/>
<complexType name="KeyDescriptorType">
    <sequence>
        <element ref="ds:KeyInfo"/>
        <element ref="md:EncryptionMethod" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
    <attribute name="use" type="md:KeyTypes" use="optional"/>
</complexType>
<simpleType name="KeyTypes">
    <restriction base="string">
        <enumeration value="encryption"/>
        <enumeration value="signing"/>
    </restriction>
</simpleType>
<element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>
```

9.1.4.2 Complex type SSODescriptorType

The **SSODescriptorType** abstract type is a common base type for the concrete types **SPSSODescriptorType** and **IDPSSODescriptorType**, described in subsequent clauses. It extends **RoleDescriptorType** with elements reflecting profiles common to both identity providers and service providers that support SSO, and contains the following additional elements:

– `<ArtifactResolutionService>` [Zero or More]

Zero or more elements of type **IndexedEndpointType** that describe indexed endpoints that support the Artifact Resolution profile, defined in clause 12. The `ResponseLocation` attribute must be omitted.

– `<SingleLogoutService>` [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the Single Logout profiles, defined in clause 12.

– `<ManageNameIDService>` [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the Name Identifier Management profiles, defined in clause 12.

– `<NameIDFormat>` [Zero or More]

Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by this system entity acting in this role.

The following schema fragment defines the **SSODescriptorType** complex type:

```
<complexType name="SSODescriptorType" abstract="true">
      <complexContent>
             <extension base="md:RoleDescriptorType">
                   <sequence>
                          <element ref="md:ArtifactResolutionService"
minOccurs="0" maxOccurs="unbounded"/>
                          <element ref="md:SingleLogoutService"
minOccurs="0" maxOccurs="unbounded"/>
                          <element ref="md:ManageNameIDService"
minOccurs="0" maxOccurs="unbounded"/>
                          <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
                   </sequence>
             </extension>
      </complexContent>
</complexType>
<element name="ArtifactResolutionService" type="md:IndexedEndpointType"/>
<element name="SingleLogoutService" type="md:EndpointType"/>
```

```
<element name="ManageNameIDService" type="md:EndpointType"/>
<element name="NameIDFormat" type="anyURI"/>
```

9.1.4.3 Element <IDPSSODescriptor>

The `<IDPSSODescriptor>` element extends **SSODescriptorType** with content reflecting profiles specific to identity providers supporting SSO. Its **IDPSSODescriptorType** complex type contains the following additional elements and attributes:

– `WantAuthnRequestsSigned` [Optional]

Optional attribute that indicates a requirement for the `<samlp:AuthnRequest>` messages received by this identity provider to be signed. If omitted, the value is assumed to be `false`.

– `<SingleSignOnService>` [One or More]

One or more elements of type **EndpointType** that describe endpoints that support the profiles of the Authentication Request protocol, defined in clause 12. All identity providers support at least one such endpoint, by definition. The `ResponseLocation` attribute must be omitted.

– `<NameIDMappingService>` [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the Name Identifier Mapping profile, defined in clause 12. The `ResponseLocation` attribute must be omitted.

– `<AssertionIDRequestService>` [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the profile of the Assertion Request protocol or the special URI binding for assertion requests defined in clause 10.

NOTE 1 (informative) – PE33 (see OASIS PE:2006) suggests to replace Assertion Request protocol with Assertion Query/Request.

– `<AttributeProfile>` [Zero or More]

Zero or more elements of type **anyURI** that enumerate the attribute profiles supported by this identity provider.

– `<saml:Attribute>` [Zero or More]

Zero or more elements that identify the SAML attributes supported by the identity provider. Specific values may optionally be included, indicating that only certain values permitted by the attribute's definition are supported. In this context, "support" for an attribute means that the identity provider has the capability to include it when delivering assertions during single sign-on.

NOTE 2 (informative) – PE7 (see OASIS PE:2006) suggests to add the following text to the end of the above paragraph:

The `WantAuthnRequestsSigned` attribute is intended to indicate to service providers whether or not they can expect an unsigned `<AuthnRequest>` message to be accepted by the identity provider. The identity provider is not obligated to reject unsigned requests nor is a service provider obligated to sign its requests, although it might reasonably expect an unsigned request will be rejected. In some cases, a service provider may not even know which identity provider will ultimately receive and respond to its requests, so the use of this attribute in such a case cannot be strictly defined. Furthermore, note that the specific method of signing that would be expected is binding dependent. The HTTP Redirect binding in 10.2.4 requires the signature be applied to the URL-encoded value rather than placed within the XML message, while other bindings generally permit the signature to be within the message in the usual fashion.

The following schema fragment defines the `<IDPSSODescriptor>` element and its **IDPSSODescriptorType** complex type:

```
<element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>
<complexType name="IDPSSODescriptorType">
       <complexContent>
              <extension base="md:SSODescriptorType">
                     <sequence>
                            <element ref="md:SingleSignOnService"
maxOccurs="unbounded"/>
                            <element ref="md:NameIDMappingService"
minOccurs="0" maxOccurs="unbounded"/>
                            <element ref="md:AssertionIDRequestService"
minOccurs="0" maxOccurs="unbounded"/>
                            <element ref="md:AttributeProfile" minOccurs="0"
maxOccurs="unbounded"/>
```

```
                            <element ref="saml:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
                     </sequence>
                     <attribute name="WantAuthnRequestsSigned"
type="boolean" use="optional"/>
              </extension>
       </complexContent>
</complexType>
<element name="SingleSignOnService" type="md:EndpointType"/>
<element name="NameIDMappingService" type="md:EndpointType"/>
<element name="AssertionIDRequestService" type="md:EndpointType"/>
<element name="AttributeProfile" type="anyURI"/>
```

9.1.4.4 Element <SPSSODescriptor>

The `<SPSSODescriptor>` element extends **SSODescriptorType** with content reflecting profiles specific to service providers. Its **SPSSODescriptorType** complex type contains the following additional elements and attributes:

– `AuthnRequestsSigned` [Optional]

Optional attribute that indicates whether the `<samlp:AuthnRequest>` messages sent by this service provider will be signed. If omitted, the value is assumed to be `false`.

NOTE 1 (informative) – PE7 (see OASIS PE:2006) suggests to add the following text to the end of the above paragraph:

A value of false (or omission of this attribute) does not imply that the service provider will never sign its requests or that a signed request should be considered an error. However, an identity provider that receives an unsigned `<samlp:AuthnRequest>` message from a service provider whose metadata contains this attribute with a value of `true` must return a SAML error response and must not fulfil the request.

– `WantAssertionsSigned` [Optional]

Optional attribute that indicates a requirement for the `<saml:Assertion>` elements received by this service provider to be signed. If omitted, the value is assumed to be `false`. This requirement is in addition to any requirement for signing derived from the use of a particular profile/binding combination.

NOTE 2 (informative) – PE7 (see OASIS PE:2006) suggests to add the following text to the end of the above paragraph:

Note that an enclosing signature at the SAML binding or protocol layer does not suffice to meet this requirement, for example signing a `<samlp:Response>` containing the assertion(s) or a TLS connection.

– `<AssertionConsumerService>` [One or More]

One or more elements that describe indexed endpoints that support the profiles of the Authentication Request protocol defined in this Recommendation. All service providers support at least one such endpoint, by definition.

– `<AttributeConsumingService>` [Zero or More]

Zero or more elements that describe an application or service provided by the service provider that requires or desires the use of SAML attributes.

At most one `<AttributeConsumingService>` element can have the attribute `isDefault` set to `true`. It is permissible for none of the included elements to contain an `isDefault` attribute set to `true`.

The following schema fragment defines the `<SPSSODescriptor>` element and its **SPSSODescriptorType** complex type:

```
<element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
<complexType name="SPSSODescriptorType">
       <complexContent>
              <extension base="md:SSODescriptorType">
                     <sequence>
                            <element ref="md:AssertionConsumerService"
maxOccurs="unbounded"/>
                            <element ref="md:AttributeConsumingService"
minOccurs="0" maxOccurs="unbounded"/>
                     </sequence>
                     <attribute name="AuthnRequestsSigned" type="boolean"
use="optional"/>
                     <attribute name="WantAssertionsSigned" type="boolean"
use="optional"/>
```

```
              </extension>
       </complexContent>
</complexType>
<element name="AssertionConsumerService" type="md:IndexedEndpointType"/>
```

9.1.4.4.1 Element <AttributeConsumingService>

The `<AttributeConsumingService>` element defines a particular service offered by the service provider in terms of the attributes the service requires or desires. Its **AttributeConsumingServiceType** complex type contains the following elements and attributes:

– `index` [Required]

  A required attribute that assigns a unique integer value to the element so that it can be referenced in a protocol message.

– `isDefault` [Optional]

  Identifies the default service supported by the service provider. Useful if the specific service is not otherwise indicated by application context. If omitted, the value is assumed to be `false`.

– `<ServiceName>` [One or More]

  One or more language-qualified names for the service.

– `<ServiceDescription>` [Zero or More]

  Zero or more language-qualified strings that describe the service.

– `<RequestedAttribute>` [One or More]

  One or more elements specifying attributes required or desired by this service.

The following schema fragment defines the `<AttributeConsumingService>` element and its **AttributeConsumingServiceType** complex type:

```
<element name="AttributeConsumingService"
type="md:AttributeConsumingServiceType"/>
<complexType name="AttributeConsumingServiceType">
       <sequence>
              <element ref="md:ServiceName" maxOccurs="unbounded"/>
              <element ref="md:ServiceDescription" minOccurs="0"
maxOccurs="unbounded"/>
              <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>
       </sequence>
       <attribute name="index" type="unsignedShort" use="required"/>
       <attribute name="isDefault" type="boolean" use="optional"/>
</complexType>
<element name="ServiceName" type="md:localizedNameType"/>
<element name="ServiceDescription" type="md:localizedNameType"/>
```

9.1.4.4.2 Element <RequestedAttribute>

The `<RequestedAttribute>` element specifies a service provider's interest in a specific SAML attribute, optionally including specific values. Its **RequestedAttributeType** complex type extends the **saml:AttributeType** with the following attribute:

– `isRequired` [Optional]

  Optional XML attribute indicates if the service requires the corresponding SAML attribute in order to function at all (as opposed to merely finding an attribute useful or desirable).

  If specific `<saml:AttributeValue>` elements are included, then only matching values are relevant to the service.

The following schema fragment defines the `<RequestedAttribute>` element and its **RequestedAttributeType** complex type:

```
<element name="RequestedAttribute" type="md:RequestedAttributeType"/>
<complexType name="RequestedAttributeType">
 <complexContent>
     <extension base="saml:AttributeType">
          <attribute name="isRequired" type="boolean"
use="optional"/>
     </extension>
 </complexContent>
</complexType>
```

9.1.4.5 Element <AuthnAuthorityDescriptor>

The `<AuthnAuthorityDescriptor>` element extends **RoleDescriptorType** with content reflecting profiles specific to authentication authorities, SAML authorities that respond to `<samlp:AuthnQuery>` messages. Its **AuthnAuthorityDescriptorType** complex type contains the following additional elements:

– `<AuthnQueryService>` [One or More]

One or more elements of type **EndpointType** that describe endpoints that support the profile of the Authentication Query protocol defined in clause 12. All authentication authorities support at least one such endpoint, by definition.

– `<AssertionIDRequestService>` [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the profile of the Assertion Request protocol defined in clause 12 or the special URI binding for assertion requests defined in clause 10.

– `<NameIDFormat>` [Zero or More]

Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by this authority (see 8.7.3 for possible values if this element).

The following schema fragment defines the `<AuthnAuthorityDescriptor>` element and its **AuthnAuthorityDescriptorType** complex type:

```
<element name="AuthnAuthorityDescriptor"
type="md:AuthnAuthorityDescriptorType"/>
<complexType name="AuthnAuthorityDescriptorType">
    <complexContent>
         <extension base="md:RoleDescriptorType">
             <sequence>
                  <element ref="md:AuthnQueryService"
maxOccurs="unbounded"/>
                  <element ref="md:AssertionIDRequestService"
minOccurs="0" maxOccurs="unbounded"/>
                  <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
             </sequence>
         </extension>
    </complexContent>
</complexType>
<element name="AuthnQueryService" type="md:EndpointType"/>
```

9.1.4.6 Element <PDPDescriptor>

The `<PDPDescriptor>` element extends **RoleDescriptorType** with content reflecting profiles specific to policy decision points, SAML authorities that respond to `<samlp:AuthzDecisionQuery>` messages. Its **PDPDescriptorType** complex type contains the following additional elements:

– `<AuthzService>` [One or More]

One or more elements of type **EndpointType** that describe endpoints that support the profile of the Authorization Decision Query protocol defined in clause 12. All policy decision points support at least one such endpoint, by definition.

– <AssertionIDRequestService> [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the profile of the Assertion Request protocol defined in clause 12 or the special URI binding for assertion requests defined in clause 10.

NOTE (informative) – PE33 (see OASIS PE:2006) suggests to Assertion Request protocol with Assertion Query/Request.

– <NameIDFormat> [Zero or More]

Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by this authority (see 8.7.3 for some possible values of this element).

The following schema fragment defines the <PDPDescriptor> element and its **PDPDescriptorType** complex type:

```
<element name="PDPDescriptor" type="md:PDPDescriptorType"/>
<complexType name="PDPDescriptorType">
      <complexContent>
            <extension base="md:RoleDescriptorType">
                  <sequence>
                        <element ref="md:AuthzService"
maxOccurs="unbounded"/>
                        <element ref="md:AssertionIDRequestService"
minOccurs="0" maxOccurs="unbounded"/>
                        <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
<element name="AuthzService" type="md:EndpointType"/>
```

9.1.4.7 Element <AttributeAuthorityDescriptor>

The <AttributeAuthorityDescriptor> element extends **RoleDescriptorType** with content reflecting profiles specific to attribute authorities, SAML authorities that respond to <samlp:AttributeQuery> messages. Its **AttributeAuthorityDescriptorType** complex type contains the following additional elements:

– <AttributeService> [One or More]

One or more elements of type **EndpointType** that describe endpoints that support the profile of the Attribute Query protocol defined in clause 12. All attribute authorities support at least one such endpoint, by definition.

– <AssertionIDRequestService> [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the profile of the Assertion Request protocol defined in clause 12 or the special URI binding for assertion requests defined in clause 10.

NOTE (informative) – PE33 (see OASIS PE:2006) suggests to replace Assertion Request protocol with Assertion Query/Request.

– <NameIDFormat> [Zero or More]

Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by this authority (see 8.7.3 for possible values of this element).

– <AttributeProfile> [Zero or More]

Zero or more elements of type **anyURI** that enumerate the attribute profiles supported by this authority (see 8.7.3 for possible values of this element).

– <saml:Attribute> [Zero or More]

Zero or more elements that identify the SAML attributes supported by the authority. Specific values may optionally be included, indicating that only certain values permitted by the attribute's definition are supported.

The following schema fragment defines the `<AttributeAuthorityDescriptor>` element and its **AttributeAuthorityDescriptorType** complex type:

```
<element name="AttributeAuthorityDescriptor"
type="md:AttributeAuthorityDescriptorType"/>
<complexType name="AttributeAuthorityDescriptorType">
       <complexContent>
              <extension base="md:RoleDescriptorType">
                     <sequence>
                            <element ref="md:AttributeService"
maxOccurs="unbounded"/>
                            <element ref="md:AssertionIDRequestService"
minOccurs="0" maxOccurs="unbounded"/>
                            <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
                            <element ref="md:AttributeProfile" minOccurs="0"
maxOccurs="unbounded"/>
                            <element ref="saml:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
                     </sequence>
              </extension>
       </complexContent>
</complexType>
<element name="AttributeService" type="md:EndpointType"/>
```

9.1.5 Element <AffiliationDescriptor>

The `<AffiliationDescriptor>` element is an alternative to the sequence of role descriptors that is used when an `<EntityDescriptor>` describes an affiliation of SAML entities (typically service providers) rather than a single entity. The `<AffiliationDescriptor>` element provides a summary of the individual entities that make up the affiliation along with general information about the affiliation itself. Its **AffiliationDescriptorType** complex type contains the following elements and attributes:

– `affiliationOwnerID` [Required]

Specifies the unique identifier of the entity responsible for the affiliation. The owner is not presumed to be a member of the affiliation; if it is a member, its identifier must also appear in an `<AffiliateMember>` element.

– `ID` [Optional]

A document-unique identifier for the element, typically used as a reference point when signing.

– `validUntil` [Optional]

Optional attribute indicates the expiration time of the metadata contained in the element and any contained elements.

– `cacheDuration` [Optional]

Optional attribute indicates the maximum length of time a consumer should cache the metadata contained in the element and any contained elements.

– `<ds:Signature>` [Optional]

An XML signature that authenticates the containing element and its contents (see clause 8).

– `<Extensions>` [Optional]

This contains optional metadata extensions that are agreed upon between a metadata publisher and consumer. Extension elements must be namespace-qualified by a non-SAML-defined namespace.

– `<AffiliateMember>` [One or More]

One or more elements enumerating the members of the affiliation by specifying each member's unique identifier (see also 8.7.3.6).

– `<KeyDescriptor>` [Zero or More]

Optional sequence of elements that provides information about the cryptographic keys that the affiliation uses as a whole, as distinct from keys used by individual members of the affiliation, which are published in the metadata for those entities.

Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

The following schema fragment defines the `<AffiliationDescriptor>` element and its **AffiliationDescriptorType** complex type:

```
<element name="AffiliationDescriptor" type="md:AffiliationDescriptorType"/>
<complexType name="AffiliationDescriptorType">
       <sequence>
              <element ref="ds:Signature" minOccurs="0"/>
              <element ref="md:Extensions" minOccurs="0"/>
              <element ref="md:AffiliateMember" maxOccurs="unbounded"/>
              <element ref="md:KeyDescriptor" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
       <attribute name="affiliationOwnerID" type="md:entityIDType"
use="required"/>
       <attribute name="validUntil" type="dateTime" use="optional"/>
       <attribute name="cacheDuration" type="duration" use="optional"/>
       <attribute name="ID" type="ID" use="optional"/>
       <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
<element name="AffiliateMember" type="md:entityIDType"/>
```

9.1.6 Examples

The following is an example of metadata for a SAML system entity acting as an identity provider and an attribute authority. A signature is shown as a placeholder, without the actual content.

```
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
       entityID="https://IdentityProvider.com/SAML">
       <ds:Signature>...</ds:Signature>
    <IDPSSODescriptor WantAuthnRequestsSigned="true"

       protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <KeyDescriptor use="signing">
            <ds:KeyInfo>
                <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>
            </ds:KeyInfo>
        </KeyDescriptor>
        <ArtifactResolutionService isDefault="true" index="0"
                    Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                    Location="https://IdentityProvider.com/SAML/Artifact"/>
        <SingleLogoutService
                    Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                    Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
        <SingleLogoutService
                    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect"

       Location="https://IdentityProvider.com/SAML/SLO/Browser"

       ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>
        <NameIDFormat>
              urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
        </NameIDFormat>
        <NameIDFormat>
              urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
        </NameIDFormat>
        <NameIDFormat>
              urn:oasis:names:tc:SAML:2.0:nameid-format:transient
        </NameIDFormat>
        <SingleSignOnService
                    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect"

       Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
        <SingleSignOnService
                    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST"
```

```
       Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
        <saml:Attribute
                    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"
                    Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
                    FriendlyName="eduPersonPrincipalName">
        </saml:Attribute>
        <saml:Attribute
                    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"
                    Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
                    FriendlyName="eduPersonAffiliation">
            <saml:AttributeValue>member</saml:AttributeValue>
            <saml:AttributeValue>student</saml:AttributeValue>
            <saml:AttributeValue>faculty</saml:AttributeValue>
            <saml:AttributeValue>employee</saml:AttributeValue>
            <saml:AttributeValue>staff</saml:AttributeValue>
        </saml:Attribute>
    </IDPSSODescriptor>
    <AttributeAuthorityDescriptor

       protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <KeyDescriptor use="signing">
            <ds:KeyInfo>
                <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
            </ds:KeyInfo>
        </KeyDescriptor>
        <AttributeService
                    Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                    Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
        <AssertionIDRequestService
                    Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
                    Location="https://IdentityProvider.com/SAML/AA/URI"/>
        <NameIDFormat>
             urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
        </NameIDFormat>
        <NameIDFormat>
             urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
        </NameIDFormat>
        <NameIDFormat>
             urn:oasis:names:tc:SAML:2.0:nameid-format:transient
        </NameIDFormat>
        <saml:Attribute
                    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"
                    Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
                    FriendlyName="eduPersonPrincipalName">
        </saml:Attribute>
        <saml:Attribute
                    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"
                    Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
                    FriendlyName="eduPersonAffiliation">
            <saml:AttributeValue>member</saml:AttributeValue>
            <saml:AttributeValue>student</saml:AttributeValue>
            <saml:AttributeValue>faculty</saml:AttributeValue>
            <saml:AttributeValue>employee</saml:AttributeValue>
            <saml:AttributeValue>staff</saml:AttributeValue>
        </saml:Attribute>
    </AttributeAuthorityDescriptor>
    <Organization>
        <OrganizationName xml:lang="en">Identity Providers R
US</OrganizationName>
        <OrganizationDisplayName xml:lang="en">
             Identity Providers R US, a Division of Lerxst Corp.
        </OrganizationDisplayName>
        <OrganizationURL
xml:lang="en">https://IdentityProvider.com</OrganizationURL>
    </Organization>
</EntityDescriptor>
```

The following is an example of metadata for a SAML system entity acting as a service provider. A signature is shown as a placeholder, without the actual content. For illustrative purposes, the service is one that does not require users to uniquely identify themselves, but rather authorizes access on the basis of a role-like attribute.

```
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
       entityID="https://ServiceProvider.com/SAML">
       <ds:Signature>...</ds:Signature>
    <SPSSODescriptor AuthnRequestsSigned="true"

       protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <KeyDescriptor use="signing">
            <ds:KeyInfo>
                <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>
            </ds:KeyInfo>
        </KeyDescriptor>
        <KeyDescriptor use="encryption">
            <ds:KeyInfo>
                <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>
            </ds:KeyInfo>
            <EncryptionMethod
       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        </KeyDescriptor>
        <SingleLogoutService
                    Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                    Location="https://ServiceProvider.com/SAML/SLO/SOAP"/>
        <SingleLogoutService
                    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect"
                    Location="https://ServiceProvider.com/SAML/SLO/Browser"

       ResponseLocation="https://ServiceProvider.com/SAML/SLO/Response"/>
        <NameIDFormat>
             urn:oasis:names:tc:SAML:2.0:nameid-format:transient
        </NameIDFormat>
        <AssertionConsumerService isDefault="true" index="0"
                    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Artifact"

       Location="https://ServiceProvider.com/SAML/SSO/Artifact"/>
        <AssertionConsumerService index="1"
                    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST"
                    Location="https://ServiceProvider.com/SAML/SSO/POST"/>
        <AttributeConsumingService index="0">
            <ServiceName xml:lang="en">Academic Journals R US</ServiceName>
            <RequestedAttribute
                           NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"
                           Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
                           FriendlyName="eduPersonEntitlement">
                <saml:AttributeValue>
                    https://ServiceProvider.com/entitlements/123456789
                </saml:AttributeValue>
            </RequestedAttribute>
        </AttributeConsumingService>
    </SPSSODescriptor>
    <Organization>
        <OrganizationName xml:lang="en">Academic Journals R
US</OrganizationName>
        <OrganizationDisplayName xml:lang="en">
               Academic Journals R US, a Division of Dirk Corp.
        </OrganizationDisplayName>
        <OrganizationURL
xml:lang="en">https://ServiceProvider.com</OrganizationURL>
    </Organization>
</EntityDescriptor>
```

9.2 Signature processing

Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion of a `<ds:Signature>` element), with the following benefits:

9.2.1 Metadata integrity

Authentication of the metadata by a trusted signer.

A digital signature is not always required, for example if the relying party obtains the information directly from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having authenticated to the relying party by some means other than a digital signature.

Many different techniques are available for "direct" authentication and secure channel establishment between two parties. The list includes TLS, HMAC, password-based mechanisms, etc. In addition, the applicable security requirements depend on the communicating applications.

Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

In the absence of such context, it is recommended that at least the root element of a metadata instance be signed.

9.2.2 XML Signature profile

W3C XML Signature specification calls out a general XML syntax for signing data with flexibility and many choices. This clause details the constraints on these facilities so that metadata processors do not have to deal with the full generality of XML Signature processing. This usage makes specific use of the **xs:ID**-typed attributes optionally present on the elements to which signatures can apply. These attributes are collectively referred to in this clause as the identifier attributes.

1) **Signing formats and algorithms**

   XML Signature has three ways of relating a signature to a document: enveloping, enveloped and detached.

   SAML metadata must use enveloped signatures when signing the elements defined in this Recommendation. SAML processors should support the use of RSA signing and verification for public key operations in accordance with the algorithm identified by http://www.w3.org/2000/09/xmldsig#rsa-sha1.

2) **References**

   Signed metadata elements must supply a value for the identifier attribute on the signed element. The element may or may not be the root element of the actual XML document containing the signed metadata element.

   Signatures must contain a single `<ds:Reference>` containing a URI reference to the identifier attribute value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the URI attribute in the `<ds:Reference>` element must be "#foo".

   As a consequence, a metadata element's signature must apply to the content of the signed element and any child elements it contains.

3) **Canonicalization method**

   SAML implementations should use Exclusive Canonicalization, with or without comments, both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML metadata embedded in an XML context can be verified independent of that context.

4) **Transforms**

   Signatures in SAML metadata should not contain transforms other than the enveloped signature transform (with the identifier http://www.w3.org/2000/09/xmldsig#enveloped-signature) or the exclusive canonicalization transforms (with the identifier http://www.w3.org/2001/10/xml-exc-c14n# or http://www.w3.org/2001/10/xml-exc-c14n#WithComments).

   Verifiers of signatures may reject signatures that contain other transform algorithms as invalid. If they do not, verifiers must ensure that no content of the signed metadata element is excluded from the signature. This can be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by applying the transforms manually to the content and reverifying the result as consisting of the same SAML metadata.

5) **KeyInfo**

W3C XML Signature defines usage of the `<ds:KeyInfo> element`. SAML does not require the use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` may be absent.

9.3 Metadata publication and resolution

In this Recommendation, two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of) metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both approaches must attempt resolution via DNS before using the "well-known-location" mechanism.

When retrieval requires network transport of the document, the transport should be protected with mechanisms providing server authentication and integrity protection. For example, HTTP-based resolution should be protected with TLS as defined in IETF RFC 2246 amended by IETF RFC 3546.

Various mechanisms are described in this clause to aid in establishing trust in the accuracy and legitimacy of metadata, including use of XML signatures, TLS server authentication, and DNS signatures. Regardless of the mechanism(s) used, relying parties should have some means by which to establish trust in metadata information before relying on it.

9.3.1 Publication and resolution via well-known location

The following subclauses describe publication and resolution of metadata by means of a well-known location.

9.3.1.1 Publication

Entities may publish their metadata documents at a well-known location by placing the document at the location denoted by its unique identifier, which must be in the form of a URL (rather than a URN). It is strongly recommended that `https` URLs be used for this purpose. An indirection mechanism supported by the URL scheme (such as an HTTP 1.1 302 redirect) may be used if the document is not placed directly at the location. If the publishing protocol permits MIME-based identification of content types, the content type of the metadata instance must be `application/samlmetadata+xml`.

The XML document provided at the well-known location must describe the metadata only for the entity represented by the unique identifier (that is, the root element must be an `<EntityDescriptor>` with an `entityID` matching the location). If other entities need to be described, the `<AdditionalMetadataLocation>` element must be used. Thus the `<EntitiesDescriptor>` element must not be used in documents published using this mechanism, since a group of entities are not defined by such an identifier.

9.3.1.2 Resolution

If an entity's unique identifier is a URL, metadata consumers may attempt to resolve an entity's unique identifier directly, in a scheme-specific manner, by dereferencing the identifier.

9.3.2 Publishing and resolution via DNS

To improve the accessibility of metadata documents and provide additional indirection between an entity's unique identifier and the location of metadata, entities may publish their metadata document locations in a zone of their corresponding DNS as defined in IETF RFC 1034. The entity's unique identifier (a URI) is used as the input to the process. Since URIs are flexible identifiers, location publication methods and the resolution process are determined by the URI's scheme and fully-qualified name. URI locations for metadata can subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in IETF RFC 2914 and IETF RFC 3403.

It is recommended that entities publish their resource records in signed zone files using IETF RFC 2535 such that relying parties may establish the validity of the published location and authority of the zone, and integrity of the DNS response. If DNS zone signatures are present, relying parties must properly validate the signature.

9.3.2.1 Publication

This Recommendation makes use of the NAPTR resource record described in IETF RFC 2915 and IETF RFC 3403. Familiarity with these documents is encouraged.

Dynamic Delegation Discovery System (DDDS) is a general purpose system for the retrieval of information based on an application-specific input string and the application of well-known rules to transform that string until a terminal condition is reached requiring a look-up into an application-specific defined database or resolution of a URL based on the rules defined by the application. DDDS defines a specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS necessary to apply DDDS rules.

Entities may publish separate URLs when multiple metadata documents need to be distributed, or when different metadata documents are required due to multiple trust relationships that require separate keying material, or when service interfaces require separate metadata declarations. This may be accomplished through the use of the optional `<AdditionalMetadataLocation>` element, or through the regexp facility and multiple service definition fields in the NAPTR resource record itself.

If the publishing protocol permits MIME-based identification of content types, the content type of the metadata instance must be `application/samlmetadata+xml`.

If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as specified in IETF RFC 3404. Otherwise, the resolution of the metadata location proceeds as specified below.

The following is the application-specific profile of DDDS for SAML metadata resolution:

1) **First well-known rule**

   The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique identifier and extract the fully-qualified domain name (subexpression 3).

2) **The order field**

   The order field indicates the order for processing each NAPTR resource record returned. Publishers may provide multiple NAPTR resource records which must be processed by the resolver application in the order indicated by this field.

3) **The preference field**

   For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving application. The resolving application may ignore this order, in cases where the service field value does not meet the resolver's requirements (e.g., the resource record returns a protocol the application does not support).

4) **The flag field**

   SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying additional resource records are to be processed). The "U" flag indicates that the output of the rule is a URI.

5) **The service field**

   The SAML-specific service field, as described in the following BNF, declares the modes by which instance document(s) shall be made available:

```
servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":"
servicetype)]
proto = 1("https" / "uddi")
class = 1[ "entity" / "entitygroup" )
servicetype = 1(si / "spsso" / "idpsso" / "authn" / "authnauth" / "pdp" /
"attrauth" / alphanum )
si = "si" [":" alphanum] [":endpoint"]
alphanum = 1*32(ALPHA / DIGIT)
```

where:

– `servicefield` PID2U resolves an entity's unique identifier to metadata URL.

– `servicefield` NID2U resolves a principal's `<NameID>` into a metadata URL.

– `proto` describes the retrieval protocol (`https` or `uddi`). In the case of UDDI, the URL will be an http(s) URL referencing a WSDL document.

– `class` identifies whether the referenced metadata document describes a single entity, or multiple. In the latter case, the referenced document must contain the entity defined by the original unique identifier as a member of a group of entities within the document itself such as an `<AffiliationDescriptor>` or `<EntitiesDescriptor>`.

– `servicetype` allows an entity to publish metadata for distinct roles and services as separate documents. Resolvers who encounter multiple `servicetype` declarations will dereference the appropriate URI, depending on which service is required for an operation (e.g., an entity operating both as an identity provider and a service provider can publish metadata for each role at different locations). The `authn` service type represents a `<SingleSignOnService>` endpoint.

- `si` (with optional endpoint component) allows the publisher to either directly publish the metadata for a service instance, or by articulating a SOAP endpoint (using `endpoint`).

For example:

- `PID2U+https:entity` – represents the entity's complete metadata document available via the https protocol.
- `PID2U+uddi:entity:si:foo` – represents the WSDL document location that describes a service instance "foo".
- `PID2U+https:entitygroup:idpsso` – represents the metadata for a group of entities acting as SSO identity providers, of which the original entity is a member.
- `NID2U+https:idp` – represents the metadata for the SSO identity provider of a principal.

6) **The regex and replacement fields**

The expected output after processing the input string through the regex must be a valid `https` URL or UDDI node (WSDL document) address.

9.3.2.2 NAPTR examples

This clause lists some examples of URL and emails that can be used by entities that support NAPTR (see IETF RFC 2915).

a) Entity metadata NAPTR examples

Entities publish metadata URLs in the following manner:

```
$ORIGIN provider.biz

;; order pref f service regexp or replacement

IN NAPTR 100 10 "U" PID2U+https:entity
 "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
IN NAPTR 110 10 "U" PID2U+https: entity:trust
 "!^.*!https://foo.provider.biz:1443/mdtrust.xml!" ""
IN NAPTR 125 10 "U" PID2U+https:"
IN NAPTR 110 10 "U" PID2U+uddi:entity
 "!^.*$!https://this.uddi.node.provider.biz/libmd.wsdl" ""
```

b) Name identifier examples

A principal's employer `example.int` operates an identity provider which may be used by an office supply company to authenticate authorized buyers. The supplier takes a users' email address `buyer@example.int` as input to the resolution process, and parses the email address to extract the FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```
$ORIGIN example.int

IN NAPTR 100 10 "U" NID2U+https:authn
 "!^([^@]+)@(.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
IN NAPTR 100 10 "U" NID2U+https:idp
 "!^([^@]+)@(.*)$!https://auth.example.int/app/auth?\1" ""
```

9.3.2.3 Resolution

When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial input into the resolution process, rather than as an actual location Proceed as follows:

- If the unique identifier is a URN, proceed with the resolution steps as defined in IETF RFC 3403.
- Otherwise, parse the identifier to obtain the fully-qualified domain name.
- Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource record is returned.
- Identify which resource record to use based on the service fields, then order fields, then preference fields of the result set.
- Obtain the document(s) at the provided location(s) as required by the application.

To initiate the resolution of the location of the metadata information, it will be necessary in some cases to decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

The following regular expression should be used when initiating the decomposition process:

```
^([^:/?#]+:)?/*([^:/?#]*@)?(([^/?:#]*\.)*(([^/?#:\.]+)\.([^/?#:\.]+)))(:\d+)?([^?
#]*)(\?[^#]*)?(#.*)?$
     1            2           34            56             7             8      9
10         11
```

Subexpression 3 must result in a Fully-Qualified Domain Name (FQDN), which will be the basis for retrieving metadata locations from this zone.

Upon completion of the parsing of the identifier, the application then performs a DNS query for the resulting domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses. Applications may exclude from the result set any service definitions that do not concern the present request operations.

Resolving applications must subsequently order the result set according to the order field, and may order the result set based on the preference set. Resolvers are not required to follow the ordering of the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the order flag) until a terminal NAPTR resource record is reached.

The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

9.3.2.4 Metadata location caching

Location caching must not exceed the TTL of the DNS zone from which the location was derived. Resolvers must obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of the zone.

Publishers of metadata documents should carefully consider the TTL of the zone when making changes to metadata document locations. Should such a location change occur, a publisher must either keep the document at both the old and new location until all conforming resolvers are certain to have the updated location (e.g., time of zone change + TTL), or provide an HTTP Redirect response at the old location specifying the new location.

9.3.3 Post-processing of metadata

The following subclauses describe the post-processing of metadata.

9.3.3.1 Metadata instance caching

Document caching must not exceed the `validUntil` or `cacheDuration` attribute of the subject element(s). If metadata elements have parent elements which contain caching policies, the parent element takes precedence.

To properly process the `cacheDuration` attribute, consumers must retain the date and time when the document was retrieved.

When a document or element has expired, the consumer must retrieve a fresh copy, which may require a refresh of the document location(s). Consumers should process document cache processing according to IETF RFC 2616, 13, and may request the Last-Modified date and time from the HTTP server. Publishers should ensure acceptable cache processing as described in IETF RFC 2616, 10.3.5 (304 Not Modified).

9.3.3.2 Handling of HTTPS redirects

Publishers may issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect) as defined in IETF RFC 2616, and user agents must follow the specified URL in the Redirect response. Redirects should be of the same protocol as the initial request.

9.3.3.3 Processing of XML signatures and general trust processing

Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and for the trust ascribed to the entity described by such metadata:

- Trust derived from the signature of the DNS zone from which the metadata location URL was resolved, ensuring accuracy of the metadata document location(s);
- Trust derived from signature processing of the metadata document itself, ensuring the integrity of the XML document;
- Trust derived from the TLS server authentication of the metadata location URL, ensuring the identity of the publisher of the metadata.

Post-processing of the metadata document must include signature processing at the XML-document level and may include one of the other two processes. Specifically, the relying party may choose to trust any of the cited authorities in the resolution and parsing process. Publishers of metadata must employ a document-integrity mechanism and may employ any of the other two processing profiles to establish trust in the metadata document, governed by implementation policies. The following considerations must be followed:

1) **Processing signed DNS zones**

   Verification of DNS zone signature should be processed, if present, as described in IETF RFC 2535.

2) **Processing signed documents and fragments**

   Published metadata documents should be signed, as described in this Recommendation, either by a certificate issued to the subject of the document, or by another trusted party. Publishers may consider signatures of other parties as a means of trust conveyance.

   Metadata consumers must validate signatures, when present, on the metadata document as described by in this Recommendation.

3) **Processing server authentication during metadata retrieval via TLS**

   It is strongly recommended that publishers implement TLS URLs; therefore, consumers should consider the trust inherited from the issuer of the TLS certificate. Publication URLs may not always be located in the domain of the subject of the metadata document; therefore, consumers should not presume certificates whose subject is the entity in question, as it may be hosted by another trusted party.

   As the basis of this trust may not be available against a cached document, other mechanisms should be used under such circumstances.

10 Bindings for SAML

This clause specifies SAML protocol bindings for the use of SAML assertions and request-response messages in communications protocols and frameworks.

Mappings of SAML request-response message exchanges onto standard messaging or communication protocols are called SAML *protocol bindings* (or just *bindings*). An instance of mapping SAML request-response message exchanges into a specific communication protocol `<FOO>` is termed a *<FOO> binding for SAML* or a *SAML <FOO> binding*.

For example, a SAML SOAP binding describes how SAML request and response message exchanges are mapped into SOAP message exchanges.

The intent of this Recommendation is to specify a selected set of bindings in sufficient detail to ensure that independently implemented SAML-conforming software can interoperate when using standard messaging or communication protocols.

Unless otherwise specified, a binding should be understood to support the transmission of any SAML protocol message derived from the **samlp:RequestAbstractType** and **samlp:StatusResponseType** types. Further, when a binding refers to "SAML requests and responses", it should be understood to mean any protocol messages derived from those types.

This Recommendation uses the following typographical conventions in text: `<ns:Element>`, XMLAttribute, **Datatype**, OtherKeyword. In some cases, angle brackets are used to indicate non-terminals, rather than XML elements; the intent will be clear from the context.

10.1 Guidelines for specifying additional protocol bindings

This Recommendation defines a selected set of protocol bindings, but others will possibly be developed in the future. This clause offers guidelines for third parties who wish to specify additional bindings. Following is a checklist of issues that must be addressed by each protocol binding:

– Specify three pieces of identifying information: a URI that uniquely identifies the protocol binding, postal or electronic contact information for the author, and a reference to previously defined bindings or profiles that the new binding updates or obsoletes.

– Describe the set of interactions between parties involved in the binding. Any restrictions on applications used by each party and the protocols involved in each interaction must be explicitly called out.

– Identify the parties involved in each interaction, including how many parties are involved and whether intermediaries may be involved.

– Specify the method of authentication of parties involved in each interaction, including whether authentication is required and the acceptable authentication types.

– Identify the level of support for message integrity, including the mechanisms used to ensure message integrity.

– Identify the level of support for confidentiality, including whether a third party may view the contents of SAML messages and assertions, whether the binding requires confidentiality, and the mechanisms recommended for achieving confidentiality.

– Identify the error states, including the error states at each participant, especially those that receive and process SAML assertions or messages.

– Identify security considerations, including analysis of threats and description of countermeasures.

– Identify metadata considerations, such that support for a binding involving a particular communications protocol or used in a particular profile can be advertised in an efficient and interoperable way.

10.2 Protocol bindings

The following subclauses define the protocol bindings that are specified as part of the SAML standard.

10.2.1 General considerations

The following clauses describe characteristics of all protocol bindings defined for SAML.

10.2.1.1 Use of RelayState

Some bindings define a "RelayState" mechanism for preserving and conveying state information. When such a mechanism is used in conveying a request message as the initial step of a SAML protocol, it places requirements on the selection and use of the binding subsequently used to convey the response. Namely, if a SAML request message is accompanied by RelayState data, then the SAML responder must return its SAML protocol response using a binding that also supports a RelayState mechanism, and it must place the exact RelayState data it received with the request into the corresponding RelayState parameter in the response.

10.2.1.2 Security

Unless stated otherwise, these security statements apply to all bindings. Bindings may also make additional statements about these security features.

1) **Use of TLS 1.0**

   Unless otherwise specified, in any SAML binding's use of TLS 1.0 (IETF RFC 2246), servers must authenticate to clients using an X.509 v3 certificate. The client must establish server identity based on contents of the certificate (typically through examination of the certificate's subject DN field, `subjectAltName` attribute, etc.).

2) **Data origin authentication**

   Authentication of both the SAML requester and the SAML responder associated with a message is optional and depends on the environment of use. Authentication mechanisms available at the SOAP message exchange layer or from the underlying substrate protocol (for example in many bindings the TLS or HTTP protocol) may be utilized to provide data origin authentication.

   Transport authentication will not meet end-end origin-authentication requirements in bindings where the SAML protocol message passes through an intermediary – in this case message authentication is recommended.

   SAML itself offers mechanisms for parties to authenticate to one another, but in addition SAML may use other authentication mechanisms to provide security for SAML itself.

3) **Message integrity**

   Message integrity of both SAML requests and SAML responses is optional and depends on the environment of use. The security layer in the underlying substrate protocol or a mechanism at the SOAP message exchange layer may be used to ensure message integrity.

   Transport integrity will not meet end-end integrity requirements in bindings where the SAML protocol message passes through an intermediary – in this case message integrity is recommended.

4) **Message confidentiality**

   Message confidentiality of both SAML requests and SAML responses is optional and depends on the environment of use. The security layer in the underlying substrate protocol or a mechanism at the SOAP message exchange layer may be used to ensure message confidentiality.

   Transport confidentiality will not meet end-end confidentiality requirements in bindings where the SAML protocol message passes through an intermediary.

5) **Other security considerations**

Before deployment, each combination of authentication, message integrity, and confidentiality mechanisms should be analysed for vulnerability in the context of the specific protocol exchange and the deployment environment (see Appendix I for more details). IETF RFC 2617 describes possible attacks in the HTTP environment when basic or message-digest authentication schemes are used. Special care should be given to the impact of possible caching on security.

10.2.2 SAML SOAP binding

SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics. Two major design goals for SOAP are simplicity and extensibility. SOAP attempts to meet these goals by omitting, from the messaging framework, features that are often found in distributed systems. Such features include but are not limited to "reliability", "security", "correlation", "routing", and "message exchange patterns" (MEPs).

A SOAP message is fundamentally a one-way transmission between SOAP nodes from a SOAP sender to a SOAP receiver, possibly routed through one or more SOAP intermediaries. SOAP messages are expected to be combined by applications to implement more complex interaction patterns ranging from request/response to multiple, back-and-forth "conversational" exchanges.

SOAP defines an XML message envelope that includes header and body sections, allowing data and control information to be transmitted. SOAP also defines processing rules associated with this envelope and an HTTP binding for SOAP message transmission.

The SAML SOAP binding defines how to use SOAP to send and receive SAML requests and responses.

Like SAML, SOAP can be used over multiple underlying transports. This binding has protocol-independent aspects, but also calls out the use of SOAP over HTTP as required (mandatory to implement).

10.2.2.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:bindings:SOAP`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: `urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding`

10.2.2.2 Protocol-independent aspects of the SAML SOAP binding

The following subclauses define aspects of the SAML SOAP binding that are independent of the underlying protocol, such as HTTP, on which the SOAP messages are transported. This binding only supports the use of SOAP 1.1.

10.2.2.2.1 Basic operation

SOAP 1.1 messages consist of three elements: an envelope, header data, and a message body. SAML request-response protocol elements must be enclosed within the SOAP message body.

SOAP 1.1 also defines an optional data encoding system. This system is not used within the SAML SOAP binding. This means that SAML messages can be transported using SOAP without re-encoding from the "standard" SAML schema to one based on the SOAP encoding.

The system model used for SAML conversations over SOAP is a simple request-response model.

- A system entity acting as a SAML requester transmits a SAML request element within the body of a SOAP message to a system entity acting as a SAML responder. The SAML requester must not include more than one SAML request per SOAP message or include any additional XML elements in the SOAP body.
- The SAML responder must return either a SAML response element within the body of another SOAP message or generate a SOAP fault. The SAML responder must not include more than one SAML response per SOAP message or include any additional XML elements in the SOAP body. If a SAML responder cannot, for some reason, process a SAML request, it must generate a SOAP fault. SOAP fault codes must not be sent for errors within the SAML problem domain, for example, inability to find an extension schema or as a signal that the subject is not authorized to access a resource in an authorization query.

NOTE (informative) – PE19 (see OASIS PE:2006) suggests to replace the above paragraph with:

The SAML responder should return a SOAP message containing either a SAML response element in the body or a SOAP fault. The SAML responder must not include more than one SAML response per SOAP message or include any additional XML elements in the SOAP body. SOAP fault codes should not be sent for errors within the SAML problem domain, for example, inability to find an extension schema or as a signal that the subject is not authorized to access a resource in an authorization query.

On receiving a SAML response in a SOAP message, the SAML requester must not send a fault code or other error messages to the SAML responder. Since the format for the message interchange is a simple request-response pattern, adding additional items such as error conditions would needlessly complicate the protocol.

W3C SOAP references an early draft of the XML Schema specification including an obsolete namespace. SAML requesters should generate SOAP documents referencing only the final XML schema namespace. SAML responders must be able to process both the XML schema namespace used in SOAP 1.1 (see W3C SOAP) as well as the final XML schema namespace.

10.2.2.2.2 SOAP headers

A SAML requester in a SAML conversation over SOAP may add arbitrary headers to the SOAP message. This binding does not define any additional SOAP headers.

NOTE 1 – The reason other headers need to be allowed is that some SOAP software and libraries might add headers to a SOAP message that are out of the control of the SAML-aware process. Also, some headers might be needed for underlying protocols that require routing of messages or by message security mechanisms.

A SAML responder must not require any headers in the SOAP message in order to process the SAML message correctly itself, but may require additional headers that address underlying routing or message security requirements.

NOTE 2 – The rationale is that requiring extra headers will cause fragmentation of the SAML standard and will hurt interoperability.

10.2.2.3 Use of SOAP over HTTP

A SAML processor that claims conformance to the SAML SOAP binding must implement SAML over SOAP over HTTP. This clause describes certain specifics of using SOAP over HTTP, including HTTP headers, caching, and error reporting.

The HTTP binding for SOAP is described in W3C SOAP, 6.0. It requires the use of a `SOAPAction` header as part of a SOAP HTTP request. A SAML responder must not depend on the value of this header. A SAML requester may set the value of the `SOAPAction` header as follows:

```
http://www.oasis-open.org/committees/security
```

10.2.2.3.1 HTTP headers

A SAML requester in a SAML conversation over SOAP over HTTP may add arbitrary headers to the HTTP request. This binding does not define any additional HTTP headers.

NOTE 1 – The reason other headers need to be allowed is that some HTTP software and libraries might add headers to an HTTP message that are out of the control of the SAML-aware process. Also, some headers might be needed for underlying protocols that require routing of messages or by message security mechanisms.

A SAML responder must not require any headers in the HTTP request to correctly process the SAML message itself, but may require additional headers that address underlying routing or message security requirements.

NOTE 2 – The rationale is that requiring extra headers will cause fragmentation of the SAML standard and will hurt interoperability.

10.2.2.3.2 Caching

HTTP proxies should not cache SAML protocol messages. To ensure this, the following rules should be followed.

When using HTTP 1.1, requesters should:

- Include a `Cache-Control` header field set to "`no-cache, no-store`".
- Include a `Pragma` header field set to "`no-cache`".

When using HTTP 1.1, responders should:

- Include a `Cache-Control` header field set to "`no-cache, no-store, must-revalidate, private`".
- Include a `Pragma` header field set to "`no-cache`".
- not include a Validator, such as a `Last-Modified` or ETag header.

10.2.2.3.3 Error reporting

A SAML responder that refuses to perform a message exchange with the SAML requester should return a "`403 Forbidden`" response. In this case, the content of the HTTP body is not significant.

As described in W3C SOAP, 6.2, in the case of a SOAP error while processing a SOAP request, the SOAP HTTP server must return a "`500 Internal Server Error`" response and include a SOAP message in the response with a SOAP `<SOAP-ENV:fault>` element. This type of error should be returned for SOAP-related errors detected before control is passed to the SAML processor, or when the SOAP processor reports an internal error (for example, the SOAP XML namespace is incorrect, the SAML schema cannot be located, the SAML processor throws an exception, and so on).

NOTE (informative) – PE19 (see [OASIS Document Errata]) suggests to replace the first sentence in the above paragraph with:

> As described in W3C SOAP, 6.2, in the case of a SOAP error while processing a SOAP request, the SOAP HTTP server should return a `"500 Internal Server Error"` response and include a SOAP message in the response with a SOAP `<SOAP-ENV:fault>` element.

In the case of a SAML processing error, the SOAP HTTP server must respond with "`200 OK`" and include a SAML-specified `<samlp:Status>` element in the SAML response within the SOAP body. The `<samlp:Status>` element does not appear by itself in the SOAP body, but only within a SAML response of some sort.

For more information about the use of SAML status codes, see the SAML assertions and protocols clause in this Recommendation.

10.2.2.3.4 Metadata considerations

Support for the SOAP binding should be reflected by indicating either a URL endpoint at which requests contained in SOAP messages for a particular protocol or profile are to be sent, or alternatively with a WSDL port/endpoint definition.

10.2.2.3.5 Example SAML message exchange using SOAP over HTTP

Following is an example of a query that asks for an assertion containing an attribute statement from a SAML attribute authority.

```
POST /SamlService HTTP/1.1
Host: www.example.com
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <samlp:AttributeQuery xmlns:samlp:="…"
xmlns:saml="…" xmlns:ds="…" ID="_6c3a4f8b9c2d" Version="2.0"
IssueInstant="2004-03-27T08:41:00Z"
            <ds:Signature> … </ds:Signature>
            <saml:Subject>
            …
            </saml:Subject>
        </samlp:AttributeQuery>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
Following is an example of the corresponding response, which supplies an
assertion containing the attribute statement as requested.
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <samlp:Response xmlns:samlp="…" xmlns:saml="…" xmlns:ds="…"
ID="_6c3a4f8b9c2d" Version="2.0" IssueInstant="2004-03-27T08:42:00Z">
            <saml:Issuer>https://www.example.com/SAML</saml:Issuer>
            <ds:Signature> … </ds:Signature>
            <Status>
              <StatusCode Value="…"/>
            </Status>
            <saml:Assertion>
                <saml:Subject>
                …
                </saml:Subject>
                <saml:AttributeStatement>
            …
                </saml:AttributeStatement>
             </saml:Assertion>
        </samlp:Response>
    </SOAP-Env:Body>
</SOAP-ENV:Envelope>
```

10.2.3 Reverse SOAP (PAOS) binding

This binding leverages the Reverse HTTP Binding for SOAP specification (see PAOS:2003). Implementers must comply with the general processing rules specified in PAOS in addition to those specified in this Recommendation. In case of conflict, Liberty Alliance POAS:2003 is normative.

10.2.3.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:bindings:PAOS`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

10.2.3.2 Overview

The reverse SOAP binding is a mechanism by which an HTTP requester can advertise the ability to act as a SOAP responder or a SOAP intermediary to a SAML requester. The HTTP requester is able to support a pattern where a SAML request is sent to it in a SOAP envelope in an HTTP response from the SAML requester, and the HTTP requester responds with a SAML response in a SOAP envelope in a subsequent HTTP request. This message exchange pattern supports the use case defined in the ECP SSO profile, in which the HTTP requester is an intermediary in an authentication exchange.

10.2.3.3 Message exchange

The PAOS binding includes two component message exchange patterns:

1) The HTTP requester sends an HTTP request to a SAML requester. The SAML requester responds with an HTTP response containing a SOAP envelope containing a SAML request message.

2) Subsequently, the HTTP requester sends an HTTP request to the original SAML requester containing a SOAP envelope containing a SAML response message. The SAML requester responds with an HTTP response, possibly in response to the original service request in step 1.

The ECP profile uses the PAOS binding to provide authentication of the client to the service provider before the service is provided. This occurs in the following steps, illustrated in Figure 10-1.

1) The client requests a service using an HTTP request.

2) The service provider responds with a SAML authentication request. This is sent using a SOAP request, carried in the HTTP response.

3) The client returns a SOAP response carrying a SAML authentication response. This is sent using a new HTTP request.

4) Assuming the service provider authentication and authorization is successful, the service provider may respond to the original service request in the HTTP response.

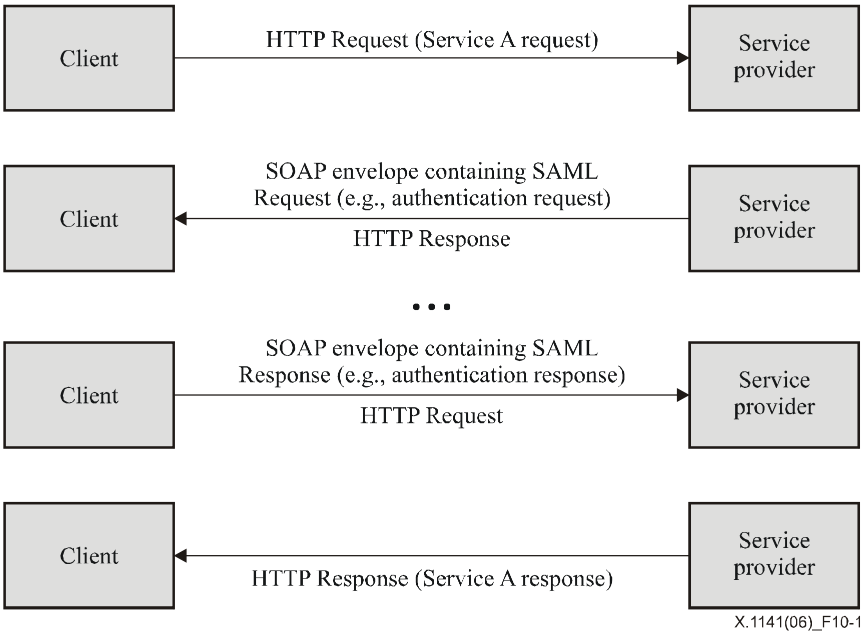
Client
HTTP Request (Service A request)
Service provider
Client
SOAP envelope containing SAML Request (e.g., authentication request)
HTTP Response
Service provider
• • •
Client
SOAP envelope containing SAML Response (e.g., authentication response)
HTTP Request
Service provider
Client
HTTP Response (Service A response)
Service provider
X.1141(06)_F10-1

Figure 10-1/X.1141 – PAOS binding message exchanges

The HTTP requester advertises the ability to handle this reverse SOAP binding in its HTTP requests using the HTTP headers defined by the PAOS:2003 specification. Specifically:

- The HTTP `Accept` Header field must indicate an ability to accept the "`application/vnd.paos+xml`" content type.
- The HTTP `PAOS` Header field must be present and specify the PAOS version with "`urn:liberty:paos:2003-08`" at a minimum.

NOTE 1 (informative) – PE21 (see OASIS PE:2006) suggests to delete "at a minimum" from the above text.

Additional PAOS headers such as the service value may be specified by profiles that use the PAOS binding. The HTTP requester may add arbitrary headers to the HTTP request.

NOTE 2 – This binding does not define a RelayState mechanism. Specific profiles that make use of this binding must therefore define such a mechanism, if needed. The use of a SOAP header is suggested for this purpose.

The following subclauses provide more detail on the two steps of the message exchange.

10.2.3.3.1 HTTP request, SAML request in SOAP response

In response to an arbitrary HTTP request, the HTTP responder may return a SAML request message using this binding by returning a SOAP 1.1 envelope in the HTTP response containing a single SAML request message in the SOAP body, with no additional body content. The SOAP envelope may contain arbitrary SOAP headers defined by PAOS, SAML profiles, or additional Recommendations.

While the SAML request message is delivered to the HTTP requester, the actual intended recipient may be another system entity, with the HTTP requester acting as an intermediary, as defined by specific profiles.

10.2.3.3.2 SAML response in SOAP request, HTTP response

When the HTTP requester delivers a SAML response message to the intended recipient using the PAOS binding, it places it as the only element in the SOAP body in a SOAP envelope in an HTTP request. The HTTP requester may or may not be the originator of the SAML response. The SOAP envelope may contain arbitrary SOAP headers defined by PAOS, SAML profiles, or additional Recommendations. The SAML exchange is considered complete and the HTTP response is unspecified by this binding.

Profiles may define additional constraints on the HTTP content of non-SOAP responses during the exchanges covered by this binding.

10.2.3.4 Caching

HTTP proxies should not cache SAML protocol messages. To ensure this, the following rules should be followed.

When using HTTP 1.1, requesters sending SAML protocol messages should:

- include a `Cache-Control` header field set to "`no-cache, no-store`";
- include a `Pragma` header field set to "`no-cache`".

When using HTTP 1.1, responders returning SAML protocol messages should:

- include a `Cache-Control` header field set to "`no-cache`, `no-store`, `must-revalidate`, `private`";
- include a `Pragma` header field set to "`no-cache`";
- not include a Validator, such as a `Last-Modified` or ETag header.

10.2.3.5 Security considerations

The HTTP requester in the PAOS binding may act as a SOAP intermediary and when it does, transport layer security for origin authentication, integrity and confidentiality may not meet end-end security requirements. In this case, security at the SOAP message layer is recommended.

NOTE (informative) – PE31 (see OASIS PE:2006) suggests to change recommended to RECOMMENDED.

10.2.3.5.1 Error reporting

Standard HTTP and SOAP error conventions must be observed. Errors that occur during SAML processing must not be signalled at the HTTP or SOAP layer and must be handled using SAML response messages with an error `<samlp:Status>` element.

10.2.3.5.2 Metadata considerations

Support for the PAOS binding should be reflected by indicating a URL endpoint at which HTTP requests and/or SAML protocol messages contained in SOAP envelopes for a particular protocol or profile are to be sent. Either a single endpoint or distinct request and response endpoints may be supplied.

10.2.4 HTTP redirect binding

The HTTP Redirect binding defines a mechanism by which SAML protocol messages can be transmitted within URL parameters. Permissible URL length is theoretically infinite, but unpredictably limited in practice. Therefore, specialized encodings are needed to carry XML messages on a URL, and larger or more complex message content can be sent using the HTTP POST or Artifact bindings.

This binding may be composed with the HTTP POST binding (see 10.2.5) and the HTTP Artifact binding (see 10.2.6) to transmit request and response messages in a single protocol exchange using two different bindings.

This binding involves the use of a message encoding. While the definition of this binding includes the definition of one particular message encoding, others may be defined and used.

10.2.4.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

10.2.4.2 Overview

The HTTP Redirect binding is intended for cases in which the SAML requester and responder need to communicate using an HTTP user agent (as defined in IETF RFC 2616) as an intermediary. This may be necessary, for example, if the communicating parties do not share a direct path of communication. It may also be needed if the responder requires an interaction with the user agent in order to fulfil the request, such as when the user agent must authenticate to it.

Some HTTP user agents may have the capacity to play a more active role in the protocol exchange and may support other bindings that use HTTP, such as the SOAP and Reverse SOAP bindings. This binding assumes nothing apart from the capabilities of a common web browser.

10.2.4.3 RelayState

RelayState data may be included with a SAML protocol message transmitted with this binding. The value must not exceed 80 bytes in length and should be integrity protected by the entity creating the message independent of any other protections that may or may not exist during message transmission. Signing is not realistic given the space limitation, but because the value is exposed to third-party tampering, the entity should ensure that the value has not been tampered with by using a checksum, a pseudo-random value, or similar means.

NOTE (informative) – PE1 (see OASIS PE:2006) states that last sentence in the above paragraph should read as below:

> RelayState data may be included with a SAML protocol message transmitted with this binding. The value must not exceed 80 bytes in length and should be integrity protected by the entity creating the message, either via a digital signature (see clause 10) or by some independent means.

If a SAML request message is accompanied by RelayState data, then the SAML responder must return its SAML protocol response using a binding that also supports a RelayState mechanism, and it must place the exact data it received with the request into the corresponding RelayState parameter in the response.

If no such value is included with a SAML request message, or if the SAML response message is being generated without a corresponding request, then the SAML responder may include RelayState data to be interpreted by the recipient based on the use of a profile or prior agreement between the parties.

10.2.4.4 Message encoding

Messages are encoded for use with this binding using a URL encoding technique, and transmitted using the HTTP GET method. There are many possible ways to encode XML into a URL, depending on the constraints in effect. This Recommendation defines one such method without precluding others. Binding endpoints should indicate which encodings they support using metadata, when appropriate. Particular encodings must be uniquely identified with a URI when defined. It is not a requirement that all possible SAML messages be encodable with a particular set of rules, but the rules must clearly indicate which messages or content can or cannot be so encoded.

A URL encoding must place the message entirely within the URL query string, and must reserve the rest of the URL for the endpoint of the message recipient.

A query string parameter named `SAMLEncoding` is reserved to identify the encoding mechanism used. If this parameter is omitted, then the value is assumed to be `urn:oasis:names:tc:SAML:2.0:bindings:URL-Encoding:DEFLATE`.

All endpoints that support this binding must support the DEFLATE encoding described next.

i) DEFLATE encoding

Identification: `urn:oasis:names:tc:SAML:2.0:bindings:URL-Encoding:DEFLATE`

SAML protocol messages can be encoded into a URL via the DEFLATE (IETF RFC 1951) compression method. In such an encoding, the following procedure should be applied to the original SAML protocol message's XML serialization:

1) Any signature on the SAML protocol message, including the `<ds:Signature>` XML element itself, must be removed. If the content of the message includes another signature, such as a signed SAML assertion, this embedded signature is not removed. However, the length of such a message after encoding essentially precludes using this mechanism. Thus SAML protocol messages that contain signed content should not be encoded using this mechanism.

2) The DEFLATE compression mechanism, as specified in IETF RFC 1951 is then applied to the entire remaining XML content of the original SAML protocol message.

3) The compressed data is subsequently base64-encoded according to the rules specified in IETF RFC 2045. Linefeeds or other whitespace must be removed from the result.

4) The base-64 encoded data is then URL-encoded, and added to the URL as a query string parameter which must be named `SAMLRequest` (if the message is a SAML request) or `SAMLResponse` (if the message is a SAML response).

5) If RelayState data is to accompany the SAML protocol message, it must be URL-encoded and placed in an additional query string parameter named `RelayState`.

6) If the original SAML protocol message was signed using an XML digital signature, a new signature covering the encoded data as specified above must be attached using the rules stated below.

XML digital signatures are not directly URL-encoded according to the above rules, due to space concerns. If the underlying SAML protocol message is signed with an XML signature, the URL-encoded form of the message must be signed as follows:

1) The signature algorithm identifier must be included as an additional query string parameter, named `SigAlg`. The value of this parameter must be a URI that identifies the algorithm used to sign the URL-encoded SAML protocol message, specified according to XML signature or whatever Recommendation governs the algorithm.

2) To construct the signature, a string consisting of the concatenation of the `RelayState` (if present), `SigAlg`, and `SAMLRequest` (or `SAMLResponse`) query string parameters (each one URL-encoded) is constructed in one of the following ways (ordered as below):

   a) `SAMLRequest=value&RelayState=value&SigAlg=value`
      `SAMLResponse=value&RelayState=value&SigAlg=value`

   b) The resulting string of bytes is the octet string to be fed into the signature algorithm. Any other content in the original query string is not included and not signed.

   c) The signature value must be encoded using the base64 encoding (see IETF RFC 2045) with any whitespace removed, and included as a query string parameter named `Signature`. Some characters in the base64-encoded signature value may themselves require URL-encoding before being added.

   d) The following signature algorithms (see W3C Signature) and their URI representations must be supported with this encoding mechanism:

      - DSAwithSHA1 – http://www.w3.org/2000/09/xmldsig#dsa-sha1
      - RSAwithSHA1 – http://www.w3.org/2000/09/xmldsig#rsa-sha1

   NOTE – NIST (National Institute of Standards and Technology) now encourages the use of SHA-256 (Secure Hash Algorithm with 256-bit encoded keys) instead of SHA-1.

When verifying signatures, the order of the query string parameters on the resulting URL to be verified is not prescribed by this binding. The parameters may appear in any order. Before verifying a signature, if any, the relying party must ensure that the parameter values to be verified are ordered as required by the signing rules above.

URL-encoding is not canonical; that is, there are multiple legal encodings for a given value. The relying party must therefore perform the verification step using the original URL-encoded values it received on the query string. It is not sufficient to re-encode the parameters after they have been processed by software because the resulting encoding may not match the signer's encoding.

If there is no `RelayState` value, the entire parameter should be omitted from the signature computation (and not included as an empty parameter name).

10.2.4.5 Message exchange

The system model used for SAML conversations via this binding is a request-response model, but these messages are sent to the user agent in an HTTP response and delivered to the message recipient in an HTTP request. The HTTP interactions before, between, and after these exchanges take place is unspecified. Both the SAML requester and the SAML responder are assumed to be HTTP responders. See the following sequence diagram (Figure 10-2) illustrating the messages exchanged.

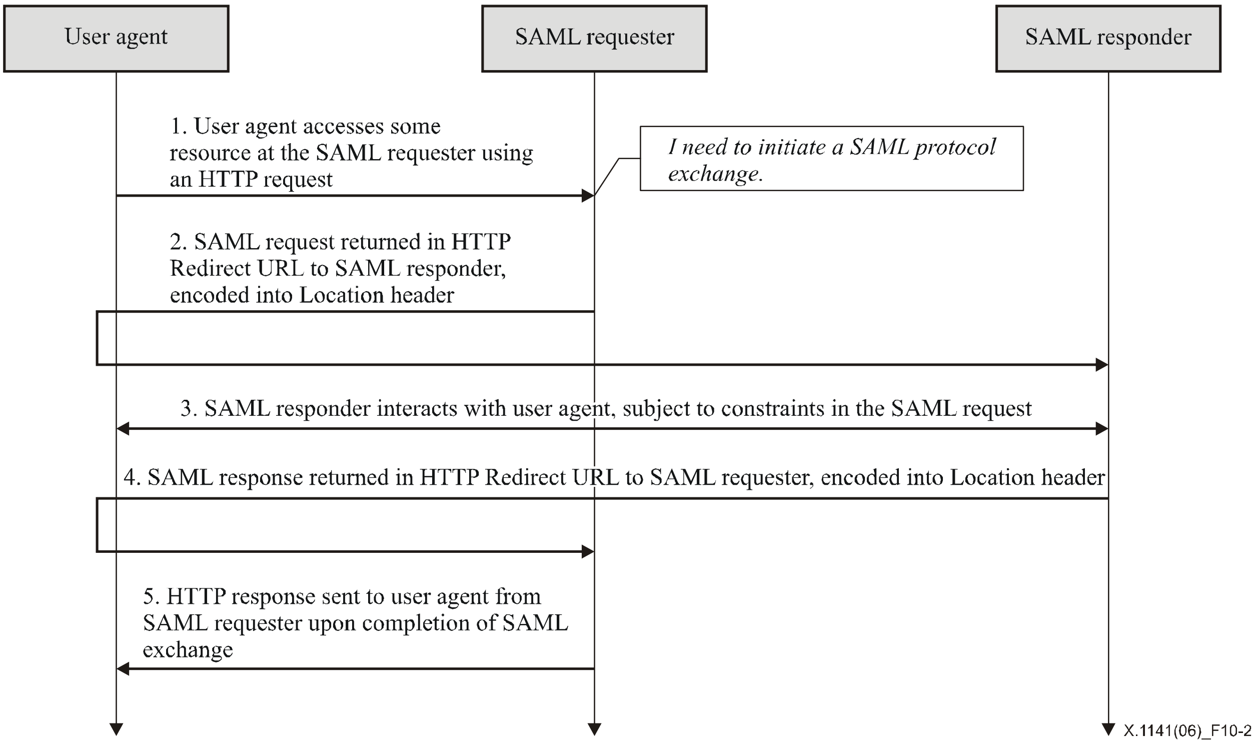
User agent
SAML requester
SAML responder
1. User agent accesses some resource at the SAML requester using an HTTP request
I need to initiate a SAML protocol exchange.
2. SAML request returned in HTTP Redirect URL to SAML responder, encoded into Location header
3. SAML responder interacts with user agent, subject to constraints in the SAML request
4. SAML response returned in HTTP Redirect URL to SAML requester, encoded into Location header
5. HTTP response sent to user agent from SAML requester upon completion of SAML exchange
X.1141(06)_F10-2

Figure 10-2/X.1141 – HTTP redirect message exchange

1) Initially, the user agent makes an arbitrary HTTP request to a system entity. In the course of processing the request, the system entity decides to initiate a SAML protocol exchange.

2) The system entity acting as a SAML requester responds to the HTTP request from the user agent in step 1 by returning a SAML request. The SAML request is returned encoded into the HTTP response's Location header, and the HTTP status must be either 303 or 302. The SAML requester may include additional presentation and content in the HTTP response to facilitate the user agent's transmission of the message, as defined in IETF RFC 2616. The user agent delivers the SAML request by issuing an HTTP GET request to the SAML responder.

3) In general, the SAML responder may respond to the SAML request by immediately returning a SAML response or may return arbitrary content to facilitate subsequent interaction with the user agent necessary to fulfil the request. Specific protocols and profiles may include mechanisms to indicate the requester's level of willingness to permit this kind of interaction (for example, the `IsPassive` attribute in `<samlp:AuthnRequest>`).

4) Eventually the responder should return a SAML response to the user agent to be returned to the SAML requester. The SAML response is returned in the same fashion as described for the SAML request in step 2.

5) Upon receiving the SAML response, the SAML requester returns an arbitrary HTTP response to the user agent.

10.2.4.5.1 HTTP and caching considerations

HTTP proxies and the user agent intermediary should not cache SAML protocol messages. To ensure this, the following rules should be followed.

When returning SAML protocol messages using HTTP 1.1, HTTP responders should:

- Include a `Cache-Control` header field set to "`no-cache, no-store`".
- Include a `Pragma` header field set to "`no-cache`".

There are no other restrictions on the use of HTTP headers.

10.2.4.5.2 Security considerations

The presence of the user agent intermediary means that the requester and responder cannot rely on the transport layer for end-end authentication, integrity and confidentiality. URL-encoded messages may be signed to provide origin authentication and integrity if the encoding method specifies a means for signing.

If the message is signed, the `Destination` XML attribute in the root SAML element of the protocol message must contain the URL to which the sender has instructed the user agent to deliver the message. The recipient must then verify that the value matches the location at which the message has been received.

This binding should not be used if the content of the request or response should not be exposed to the user agent intermediary. Otherwise, confidentiality of both SAML requests and SAML responses is optional and depends on the environment of use. If confidentiality is necessary, or TLS 1.0 should be used to protect the message in transit between the user agent and the SAML requester and responder.

URL-encoded messages may be exposed in a variety of HTTP logs as well as the HTTP "Referrer" header.

Before deployment, each combination of authentication, message integrity, and confidentiality mechanisms should be analysed for vulnerability in the context of the specific protocol exchange, and the deployment environment (see Appendix I).

In general, this binding relies on message-level authentication and integrity protection via signing and does not support confidentiality of messages from the user agent intermediary.

10.2.4.6 Error reporting

A SAML responder that refuses to perform a message exchange with the SAML requester should return a SAML response message with a second-level `<samlp:StatusCode>` value of `urn:oasis:names:tc:SAML:2.0:status:RequestDenied`.

HTTP interactions during the message exchange must not use HTTP error status codes to indicate failures in SAML processing, since the user agent is not a full party to the SAML protocol exchange. See also clause 9.

10.2.4.7 Metadata considerations

Support for the HTTP Redirect binding should be reflected by indicating URL endpoints at which requests and responses for a particular protocol or profile should be sent. Either a single endpoint or distinct request and response endpoints may be supplied.

NOTE (informative) – PE2 (see OASIS PE:2006) states to replace the above paragraph with:

Support for receiving messages using the HTTP Artifact binding should be reflected by indicating URL endpoints at which requests and responses for a particular protocol or profile should be sent. Either a single endpoint or distinct request and response endpoints may be supplied. Support for sending messages using this binding should be accompanied by one or more indexed `<md:ArtifactResolutionService>` endpoints for processing `<samlp:ArtifactResolve>` messages.

10.2.4.8 Example SAML message exchange using HTTP Redirect

In this example, a `<LogoutRequest>` and `<LogoutResponse>` message pair is exchanged using the HTTP Redirect binding.

First, here are the actual SAML protocol messages being exchanged:

```
<samlp:LogoutRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="d2b7c388cec36fa7c39c28fd298644a8" IssueInstant="2004-01-
21T19:00:49Z" Version="2.0">
    <Issuer>https://IdentityProvider.com/SAML</Issuer>
    <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">005a06e0-ad82-110d-a556-004005b13a2b</NameID>
    <samlp:SessionIndex>1</samlp:SessionIndex>
</samlp:LogoutRequest>

<samlp:LogoutResponse xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="b0730d21b628110d8b7e004005b13a2b"
InResponseTo="d2b7c388cec36fa7c39c28fd298644a8"
    IssueInstant="2004-01-21T19:00:49Z" Version="2.0">
    <Issuer>https://ServiceProvider.com/SAML</Issuer>
    <samlp:Status>
        <samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
```

```
    </samlp:Status>
</samlp:LogoutResponse>
```

The initial HTTP request from the user agent in the above is not defined by this binding. To initiate the logout protocol exchange, the SAML requester returns the following HTTP response, containing a signed SAML request message. The `SAMLRequest` parameter value is actually derived from the request message above. The signature portion is only illustrative and not the result of an actual computation. The line feeds in the HTTP `Location` header below are an artifact of the document, and there are no line feeds in the actual header value.

```
HTTP/1.1 302 Object Moved
Date: 21 Jan 2004 07:00:49 GMT
Location:
https://ServiceProvider.com/SAML/SLO/Browser?SAMLRequest=fVFdS8MwFH0f7D%2BU
vGdNsq62oSsIQyhMESc%2B%2BJYlmRbWpObeyvz3puv2IMjyFM7HPedyK1DdsZdb%2F%2BEHfLF
fgwVMTt3RgTwzazIEJ72CFqRTnQWJWu7uH7dSLJjsg0ev%2FZFMlttiBWADtt6R%2BSyJr9msiR
H7O70sCm31Mj%2Bo%2BC%2B1KA5GlEWeZaogSQMw2MYBKodrIhjLKONU8FdeSsZkVr6T5M0GiHM
jvWCknqZXZ2OoPxF7kGnaGOuwxZ%2Fn4L9bY8NC%2By4du1XpRXnxPcXizSZ58KFTeHujEWkNPZ
ylsh9bAMYYUjO2Uiy3jCpTCMo5M1StVjmN9SO150sl9lU6RV2Dp0vsLIy7NM7YU82r9B90PrvCf
85W%2FwL8zSVQzAEAAA%3D%3D&RelayState=0043bfc1bc45110dae17004005b13a2b&SigAl
g=http%3A%2F%2Fwww.w3.org%2F200%2F09%2Fxmldsig%23rsa-
sha1&Signature=NOTAREALSIGNATUREBUTTHEREALONEWOULDGOHERE
Content-Type: text/html; charset=iso-8859-1
```

After any unspecified interactions may have taken place, the SAML responder returns the HTTP response below containing the signed SAML response message. Again, the `SAMLResponse` parameter value is actually derived from the response message above. The signature portion is only illustrative and not the result of an actual computation.

```
HTTP/1.1 302 Object Moved
Date: 21 Jan 2004 07:00:49 GMT
Location:
https://IdentityProvider.com/SAML/SLO/Response?SAMLResponse=fVFNa4QwEL0X%2B
h8k912TaDUGFUp7EbZQ6rKH3mKcbQVNJBOX%2FvxaXQ9tYec0vHlv3nzkqIZ%2BlAf7YSf%2FBj
hagxB8Db1BuZQKMjkjrcIOpVEDoPRa1o8vB8n3VI7OeqttT1bJbbJCBOc7a8j9XTBH9VyQhqYRb
TlrEi4Yo61oUqA0pvShYZHiDQkqs411tAVpeZPqSAgNOkrOas4zzcW55ZlI4liJrTXiBJVBr4wv
CJ877ijbcXZkmaRUxtk7CU7gcB5mLu8pKVddvghd%2Ben9iDIMa3CXTsOrs5euBbfXdgh%2F9sn
DK%2FEqW69Ye%2BUnvGL%2F8CfbQnBS%2FQS3z4QLW9aT1oBIws0j%2FGOyAb9%2FV34Dw5k779
IBAAA%3D&RelayState=0043bfc1bc45110dae17004005b13a2b&SigAlg=http%3A%2F%2Fww
w.w3.org%2F200%2F09%2Fxmldsig%23rsa-
sha1&Signature=NOTAREALSIGNATUREBUTTHEREALONEWOULDGOHERE
Content-Type: text/html; charset=iso-8859-1
```

10.2.5 HTTP POST binding

The HTTP POST binding defines a mechanism by which SAML protocol messages may be transmitted within the base64-encoded content of an HTML form control.

This binding may be composed with the HTTP Redirect binding (see 10.2.4) and the HTTP Artifact binding (see 10.2.6) to transmit request and response messages in a single protocol exchange using two different bindings.

10.2.5.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

10.2.5.2 Overview

The HTTP POST binding is intended for cases in which the SAML requester and responder need to communicate using an HTTP user agent (as defined in IETF RFC 2616) as an intermediary. This may be necessary, for example, if the communicating parties do not share a direct path of communication. It may also be needed if the responder requires an interaction with the user agent in order to fulfil the request, such as when the user agent must authenticate to it.

Some HTTP user agents may have the capacity to play a more active role in the protocol exchange and may support other bindings that use HTTP, such as the SOAP and Reverse SOAP bindings. This binding assumes nothing apart from the capabilities of a common web browser.

10.2.5.3 RelayState

RelayState data may be included with a SAML protocol message transmitted with this binding. The value must not exceed 80 bytes in length and should be integrity protected by the entity creating the message independent of any other protections that may or may not exist during message transmission. Signing is not realistic given the space limitation, but because the value is exposed to third-party tampering, the entity should ensure that the value has not been tampered with by using a checksum, a pseudo-random value, or similar means.

If a SAML request message is accompanied by RelayState data, then the SAML responder must return its SAML protocol response using a binding that also supports a RelayState mechanism, and it must place the exact data it received with the request into the corresponding RelayState parameter in the response.

If no such value is included with a SAML request message, or if the SAML response message is being generated without a corresponding request, then the SAML responder may include RelayState data to be interpreted by the recipient based on the use of a profile or prior agreement between the parties.

NOTE (informative) – PE31 (see OASIS PE:2006) suggests to clarify the above paragraph as below:

> If no RelayState parameter is included with a SAML request message, or if the SAML response message is being generated without a corresponding request, then the SAML responder may include RelayState data to be interpreted by the recipient based on the use of a profile or prior agreement between the parties.

10.2.5.4 Message encoding

Messages are encoded for use with this binding by encoding the XML into an HTML form control and are transmitted using the HTTP POST method. A SAML protocol message is form-encoded by applying the base-64 encoding rules to the XML representation of the message and placing the result in a hidden form control within a form as defined by W3C HTML, clause 17. The HTML document must adhere to W3C XHTML in accordance with common practice.

If the message is a SAML request, then the form control must be named `SAMLRequest`. If the message is a SAML response, then the form control must be named `SAMLResponse`. Any additional form controls or presentation may be included but must not be required in order for the recipient to process the message.

If a "RelayState" value is to accompany the SAML protocol message, it must be placed in an additional hidden form control named `RelayState` within the same form with the SAML message.

The action attribute of the form must be the recipient's HTTP endpoint for the protocol or profile using this binding to which the SAML message is to be delivered. The method attribute must be "POST".

Any technique supported by the user agent may be used to cause the submission of the form, and any form content necessary to support this may be included, such as submit controls and client-side scripting commands. However, the recipient must be able to process the message without regard for the mechanism by which the form submission is initiated.

Any form control values included must be transformed so as to be safe to include in the XHTML document. This includes transforming characters such as quotes into HTML entities, etc.

10.2.5.5 Message exchange

The system model used for SAML conversations via this binding is a request-response model, but these messages are sent to the user agent in an HTTP response and delivered to the message recipient in an HTTP request. The HTTP interactions before, between, and after these exchanges take place is unspecified. Both the SAML requester and responder are assumed to be HTTP responders. See Figure 10-3 illustrating the messages exchanged.

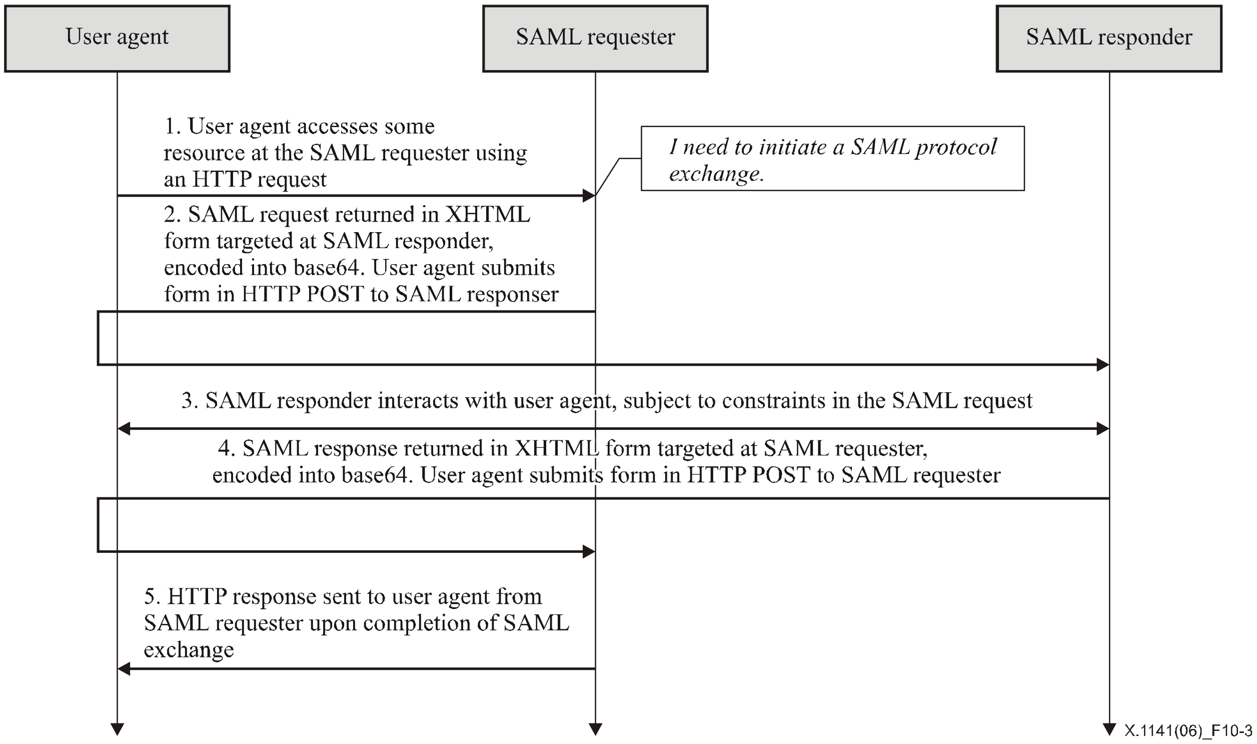
User agent
SAML requester
SAML responder
1. User agent accesses some resource at the SAML requester using an HTTP request
I need to initiate a SAML protocol exchange.
2. SAML request returned in XHTML form targeted at SAML responder, encoded into base64. User agent submits form in HTTP POST to SAML responser
3. SAML responder interacts with user agent, subject to constraints in the SAML request
4. SAML response returned in XHTML form targeted at SAML requester, encoded into base64. User agent submits form in HTTP POST to SAML requester
5. HTTP response sent to user agent from SAML requester upon completion of SAML exchange
X.1141(06)_F10-3

Figure 10-3/X.1141 – HTTP POST message exchange

1) Initially, the user agent makes an arbitrary HTTP request to a system entity. In the course of processing the request, the system entity decides to initiate a SAML protocol exchange.

2) The system entity acting as a SAML requester responds to an HTTP request from the user agent by returning a SAML request. The request is returned in an XHTML document containing the form and content defined in 10.2.5.4. The user agent delivers the SAML request by issuing an HTTP POST request to the SAML responder.

3) In general, the SAML responder may respond to the SAML request by immediately returning a SAML response or it may return arbitrary content to facilitate subsequent interaction with the user agent necessary to fulfil the request. Specific protocols and profiles may include mechanisms to indicate the requester's level of willingness to permit this kind of interaction (for example, the `IsPassive` attribute in `<samlp:AuthnRequest>`).

4) Eventually the responder should return a SAML response to the user agent to be returned to the SAML requester. The SAML response is returned in the same fashion as described for the SAML request in step 2.

5) Upon receiving the SAML response, the SAML requester returns an arbitrary HTTP response to the user agent.

10.2.5.5.1 HTTP and caching considerations

HTTP proxies and the user agent intermediary should not cache SAML protocol messages. To ensure this, the following rules should be followed.

When returning SAML protocol messages using HTTP 1.1, HTTP responders should:

- Include a `Cache-Control` header field set to "`no-cache`, `no-store`".
- Include a `Pragma` header field set to "`no-cache`".

There are no other restrictions on the use of HTTP headers.

10.2.5.5.2 Security considerations

The presence of the user agent intermediary means that the requester and responder cannot rely on the transport layer for end-end authentication, integrity or confidentiality protection and must authenticate the messages received instead. SAML provides for a signature on protocol messages for authentication and integrity for such cases. Form-encoded messages may be signed before the base64 encoding is applied.

If the message is signed, the `Destination` XML attribute in the root SAML element of the protocol message must contain the URL to which the sender has instructed the user agent to deliver the message. The recipient must then verify that the value matches the location at which the message has been received.

This binding should not be used if the content of the request or response should not be exposed to the user agent intermediary. Otherwise, confidentiality of both SAML requests and SAML responses is optional and depends on the environment of use. If confidentiality is necessary, TLS 1.0 should be used to protect the message in transit between the user agent and the SAML requester and responder.

In general, this binding relies on message-level authentication and integrity protection via signing and does not support confidentiality of messages from the user agent intermediary.

There is no mechanism defined to protect the integrity of the relationship between the SAML protocol message and the "RelayState" value, if any. That is, an attacker can potentially recombine a pair of valid HTTP responses by switching the "RelayState" values associated with each SAML protocol message. The individual "RelayState" and SAML message values can be integrity protected, but not the combination. As a result, the producer and consumer of "RelayState" information must take care not to associate sensitive state information with the "RelayState" value without taking additional precautions (such as based on the information in the SAML message).

10.2.5.6 Error reporting

A SAML responder that refuses to perform a message exchange with the SAML requester should return a response message with a second-level `<samlp:StatusCode>` value of `urn:oasis:names:tc:SAML:2.0:status:RequestDenied`.

HTTP interactions during the message exchange must not use HTTP error status codes to indicate failures in SAML processing, since the user agent is not a full party to the SAML protocol exchange.

For more information about SAML status codes, see clause 8.2.

10.2.5.7 Metadata considerations

Support for the HTTP POST binding should be reflected by indicating URL endpoints at which requests and responses for a particular protocol or profile should be sent. Either a single endpoint or distinct request and response endpoints may be supplied.

10.2.5.8 Example SAML message exchange using HTTP POST

In this example, a `<LogoutRequest>` and `<LogoutResponse>` message pair is exchanged using the HTTP POST binding.

First, here are the actual SAML protocol messages being exchanged:

```
<samlp:LogoutRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="d2b7c388cec36fa7c39c28fd298644a8" IssueInstant="2004-01-
21T19:00:49Z" Version="2.0">
    <Issuer>https://IdentityProvider.com/SAML</Issuer>
    <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">005a06e0-ad82-110d-a556-004005b13a2b</NameID>
    <samlp:SessionIndex>1</samlp:SessionIndex>
</samlp:LogoutRequest>

<samlp:LogoutResponse xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="b0730d21b628110d8b7e004005b13a2b"
InResponseTo="d2b7c388cec36fa7c39c28fd298644a8"
    IssueInstant="2004-01-21T19:00:49Z" Version="2.0">
    <Issuer>https://ServiceProvider.com/SAML</Issuer>
    <samlp:Status>
        <samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
</samlp:LogoutResponse>
```

The initial HTTP request from the user agent in the above is not defined by this binding. To initiate the logout protocol exchange, the SAML requester returns the following HTTP response, containing a SAML request message. The `SAMLRequest` parameter value is actually derived from the request message above.

```
HTTP/1.1 200 OK
Date: 21 Jan 2004 07:00:49 GMT
Content-Type: text/html; charset=iso-8859-1

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<body onload="document.forms[0].submit()">

<noscript>
<p>
<strong>Note:</strong> Since your browser does not support JavaScript, you
must press the Continue button once to proceed.
</p>
</noscript>

<form action="https://ServiceProvider.com/SAML/SLO/Browser" method="post">
<div>
<input type="hidden" name="RelayState"
value="0043bfc1bc45110dae17004005b13a2b"/>
<input type="hidden" name="SAMLRequest"
value="PHNhbWxwOkxvZ291dFJlcXVlc3QgeG1sbnM6c2FtbHA9InVybjpvYXNpczpuYW1l
czp0YzpTQU1MOjIuMDpwcm90b2NvbCIgeG1sbnM9InVybjpvYXNpczpuYW1lczp0
YzpTQU1MOjIuMDphc3NlcnRpb24iDQogICAgSUQ9ImQyYjdjMzg4Y2VjMzZmYTdj
MzljMjhmZDI5ODY0NGE4IiBJc3N1ZUluc3RhbnQ9IjIwMDQtMDEtMjFUMTk6MDA6
NDlaIiBWZXJzaW9uPSIyLjAiPg0KICAgIDxJc3N1ZXI+aHR0cHM6Ly9JZGVudGl0
eVByb3ZpZGVyLmNvbS9TQU1MPC9Jc3N1ZXI+DQogICAgPE5hbWVJRCBGb3JtYXQ9
InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDpuYW1laWQtZm9ybWF0OnBlcnNp
c3RlbnQiPjAwNWEwNmUwLWFkODItMTEwZC1hNTU2LTAwNDAwNWIxM2EyYjwvTmFt
ZUlEPg0KICAgIDxzYW1scDpTZXNzaW9uSW5kZXg+MTwvc2FtbHA6U2Vzc2lvbklu
ZGV4Pg0KPC9zYW1scDpMb2dvdXRSZXF1ZXN0Pg=="/>
</div>
<noscript>
<div>
<input type="submit" value="Continue"/>
</div>
</noscript>
</form>
</body>
</html>
```

After any unspecified interactions may have taken place, the SAML responder returns the HTTP response below containing the SAML response message. Again, the `SAMLResponse` parameter value is actually derived from the response message above.

```
HTTP/1.1 200 OK
Date: 21 Jan 2004 07:00:49 GMT
Content-Type: text/html; charset=iso-8859-1

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<body onload="document.forms[0].submit()">

<noscript>
<p>
<strong>Note:</strong> Since your browser does not support JavaScript, you
must press the Continue button once to proceed.
</p>
</noscript>

<form action="https://IdentityProvider.com/SAML/SLO/Response"
method="post">
<div>
<input type="hidden" name="RelayState"
value="0043bfc1bc45110dae17004005b13a2b"/>
<input type="hidden" name="SAMLResponse"
value="PHNhbWxwOkxvZ291dFJlc3BvbnNlIHhtbG5zOnNhbWxwPSJ1cm46b2FzaXM6bmFt
```

```
ZXM6dGM6U0FNTDoyLjA6cHJvdG9jb2wiIHhtbG5zPSJ1cm46b2FzaXM6bmFtZXM6
dGM6U0FNTDoyLjA6YXNzZXJ0aW9uIg0KICAgIElEPSJiMDczMGQyMWI2MjgxMTBk
OGI3ZTAwNDAwNWIxM2EyYiIgSW5SZXNwb25zZVRvPSJkMmI3YzM4OGNlYzM2ZmE3
YzM5YzI4ZmQyOTg2NDRhOCINCiAgICBJc3N1ZUluc3RhbnQ9IjIwMDQtMDEtMjFU
MTk6MDA6NDlaIiBWZXJzaW9uPSIyLjAiPg0KICAgIDxJc3N1ZXI+aHR0cHM6Ly9T
ZXJ2aWNlUHJvdmlkZXIuY29tL1NBTUw8L0lzc3Vlcj4NCiAgICA8c2FtbHA6U3Rh
dHVzPg0KICAgICAgICA8c2FtbHA6U3RhdHVzQ29kZSBWYWx1ZT0idXJuOm9hc2lz
Om5hbWVzOnRjOlNBTUw6Mi4wOnN0YXR1czpTdWNjZXNzIi8+DQogICAgPC9zYW1s
cDpTdGF0dXM+DQo8L3NhbWxwOkxvZ291dFJlc3BvbnNlPg=="/>
</div>
<noscript>
<div>
<input type="submit" value="Continue"/>
</div>
</noscript>
</form>
</body>
</html>
```

10.2.6 HTTP Artifact binding

In the HTTP Artifact binding, the SAML request, the SAML response, or both are transmitted by reference using a small stand-in called an artifact. A separate, synchronous binding, such as the SAML SOAP binding, is used to exchange the artifact for the actual protocol message using the artifact resolution protocol defined in clause 8.

This binding may be composed with the HTTP Redirect binding (see 10.2.4) and the HTTP POST binding (see 10.2.5) to transmit request and response messages in a single protocol exchange using two different bindings.

10.2.6.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

10.2.6.2 Overview

The HTTP Artifact binding is intended for cases in which the SAML requester and responder need to communicate using an HTTP user agent as an intermediary, but the intermediary's limitations preclude or discourage the transmission of an entire message (or message exchange) through it. This may be for technical reasons or because of a reluctance to expose the message content to the intermediary (and if the use of encryption is not practical).

Because of the need to subsequently resolve the artifact using another synchronous binding, such as SOAP, a direct communication path must exist between the SAML message sender and recipient in the reverse direction of the artifact's transmission (the receiver of the message and artifact must be able to send a `<samlp:ArtifactResolve>` request back to the artifact issuer). The artifact issuer must also maintain state while the artifact is pending, which has implications for load-balanced environments.

10.2.6.3 Message encoding

There are two methods of encoding an artifact for use with this binding. One is to encode the artifact into a URL parameter and the other is to place the artifact in an HTML form control. When URL encoding is used, the HTTP GET method is used to deliver the message, while POST is used with form encoding. All endpoints that support this binding must support both techniques.

10.2.6.3.1 RelayState

RelayState data may be included with a SAML artifact transmitted with this binding. The value must not exceed 80 bytes in length and should be integrity protected by the entity creating the message independent of any other protections that may or may not exist during message transmission. Signing is not realistic given the space limitation, but because the value is exposed to third-party tampering, the entity should ensure that the value has not been tampered with by using a checksum, a pseudo-random value, or similar means.

If an artifact that represents a SAML request is accompanied by RelayState data, then the SAML responder must return its SAML protocol response using a binding that also supports a RelayState mechanism, and it must place the exact data it received with the artifact into the corresponding RelayState parameter in the response.

If no such value is included with an artifact representing a SAML request, or if the SAML response message is being generated without a corresponding request, then the SAML responder may include RelayState data to be interpreted by the recipient based on the use of a profile or prior agreement between the parties.

10.2.6.3.2 URL encoding

To encode an artifact into a URL, the artifact value is URL-encoded and placed in a query string parameter named `SAMLart`.

If a "RelayState" value is to accompany the SAML artifact, it must be URL-encoded and placed in an additional query string parameter named `RelayState`.

10.2.6.3.3 Form encoding

A SAML artifact is form-encoded by placing it in a hidden form control within a form as defined by W3C HTML. The HTML document must adhere to W3C XHTML. The form control must be named `SAMLart`. Any additional form controls or presentation may be included but must not be required in order for the recipient to process the artifact.

If a "RelayState" value is to accompany the SAML artifact, it must be placed in an additional hidden form control named `RelayState`, within the same form with the SAML message.

The `action` attribute of the form must be the recipient's HTTP endpoint for the protocol or profile using this binding to which the artifact is to be delivered. The `method` attribute must be set to "POST".

Any technique supported by the user agent may be used to cause the submission of the form, and any form content necessary to support this may be included, such as submit controls and client-side scripting commands. However, the recipient must be able to process the artifact without regard for the mechanism by which the form submission is initiated.

Any form control values included must be transformed so as to be safe to include in the XHTML document. This includes transforming characters such as quotes into HTML entities, etc.

10.2.6.4 Artifact format

With respect to this binding, an artifact is a short, opaque string. Different types can be defined and used without affecting the binding. The important characteristics are the ability of an artifact receiver to identify the issuer of the artifact, resistance to tampering and forgery, uniqueness and compactness.

The general format of any artifact includes a mandatory two-byte artifact type code and a two-byte index value identifying a specific endpoint of the artifact resolution service of the issuer, as follows:

```
SAML_artifact      := B64(TypeCode EndpointIndex RemainingArtifact)
TypeCode           := Byte1Byte2
EndpointIndex      := Byte1Byte2
```

The notation `B64(TypeCode EndpointIndex RemainingArtifact)` stands for the application of the base64 (see IETF RFC 2045) transformation to the catenation of the `TypeCode`, `EndpointIndex`, and `RemainingArtifact`.

The following practices are recommended for the creation of SAML artifacts:

- Each issuer is assigned an identifying URI, also known as the issuer's entity (or provider) ID. See clause 8 for a discussion of this kind of identifier.
- The issuer constructs the `SourceID` component of the artifact by taking the SHA-1 hash of the identification URL. The hash value is not encoded into hexadecimal.

  NOTE 1 – NIST (National Institute of Standards and Technology) now encourages the use of SHA-256 (Secure Hash Algorithm with 256-bit encoded keys) instead of SHA-1.
- The `MessageHandle` value is constructed from a cryptographically strong random or pseudo-random number sequence (see IETF RFC 1750) generated by the issuer. The sequence consists of values of at least 16 bytes in size. These values should be padded as needed to a total length of 20 bytes.

  NOTE 2 (informative) – PE4 (see [OASIS Errata Document]) suggest to add the following text to the end of the above paragraph:

  Although the general artifact structure resembles that used in prior versions of SAML and the type code of the single format described below does not conflict with previously defined formats, there is explicitly no correspondence between SAML 2.0 artifacts and those found in any previous specifications, and artifact formats not defined specifically for use with SAML 2.0 must not be used with this binding.

The following describes the single artifact type defined by SAML V2.0.

10.2.6.4.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:artifact-04`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

10.2.6.4.2 Format details

SAML V2.0 defines an artifact type of type code 0x0004. This artifact type is defined as follows:

```
TypeCode            := 0x0004
RemainingArtifact := SourceID MessageHandle
SourceID            := 20-byte_sequence
MessageHandle       := 20-byte_sequence
```

`SourceID` is a 20-byte sequence used by the artifact receiver to determine artifact issuer identity and the set of possible resolution endpoints.

It is assumed that the destination site will maintain a table of `SourceID` values as well as one or more indexed URL endpoints (or addresses) for the corresponding SAML responder. Clause 9 may be used for this purpose. On receiving the SAML artifact, the receiver determines if the `SourceID` belongs to a known artifact issuer and obtains the location of the SAML responder using the `EndpointIndex` before sending a SAML `<samlp:ArtifactResolve>` message to it.

Any two artifact issuers with a common receiver must use distinct `SourceID` values. Construction of `MessageHandle` values is governed by the principle that they should have no predictable relationship to the contents of the referenced message at the issuing site and it must be infeasible to construct or guess the value of a valid, outstanding message handle.

10.2.6.5 Message exchange

The system model used for SAML conversations by means of this binding is a request-response model in which an artifact reference takes the place of the actual message content, and the artifact reference is sent to the user agent in an HTTP response and delivered to the message recipient in an HTTP request. The HTTP interactions before, between, and after these exchanges take place is unspecified. Both the SAML requester and responder are assumed to be HTTP responders.

Additionally, it is assumed that on receipt of an artifact by way of the user agent, the recipient invokes a separate, direct exchange with the artifact issuer using the Artifact Resolution Protocol defined in this Recommendation. This exchange must use a binding that does not use the HTTP user agent as an intermediary, such as the SOAP binding. On the successful acquisition of a SAML protocol message, the artifact is discarded and the processing of the primary SAML protocol exchange resumes (or ends, if the message is a response).

Issuing and delivering an artifact, along with the subsequent resolution step, constitutes half of the overall SAML protocol exchange. This binding can be used to deliver either or both halves of a SAML protocol exchange. A binding composable with it, such as the HTTP Redirect (see 10.2.4) or POST (see 10.2.5) binding, may be used to carry the other half of the exchange. The following sequence assumes that the artifact binding is used for both halves. See Figure 10-4 below illustrating the messages exchanged.

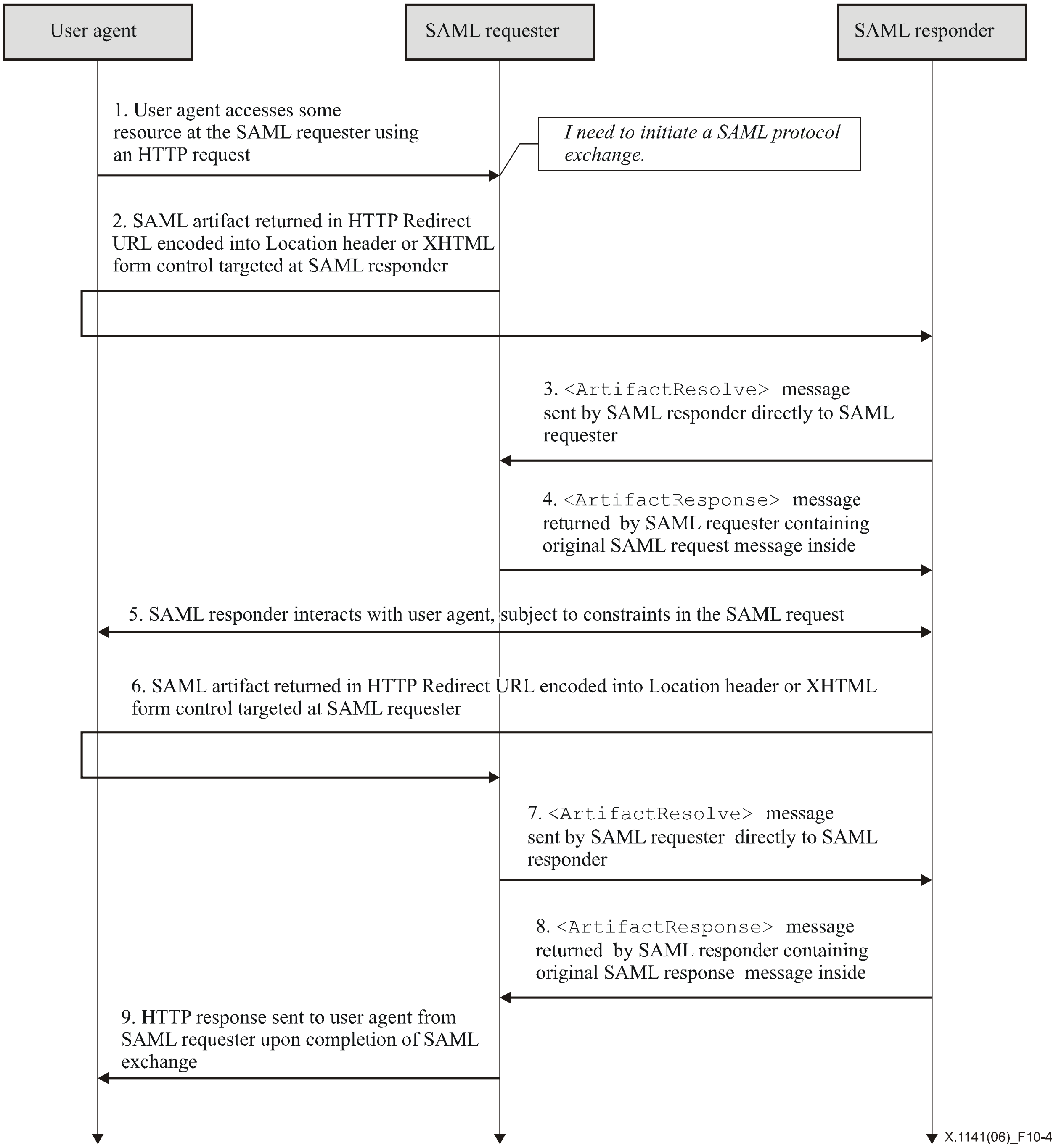
User agent
SAML requester
SAML responder
1. User agent accesses some resource at the SAML requester using an HTTP request
I need to initiate a SAML protocol exchange.
2. SAML artifact returned in HTTP Redirect URL encoded into Location header or XHTML form control targeted at SAML responder
3. <ArtifactResolve> message sent by SAML responder directly to SAML requester
4. <ArtifactResponse> message returned by SAML requester containing original SAML request message inside
5. SAML responder interacts with user agent, subject to constraints in the SAML request
6. SAML artifact returned in HTTP Redirect URL encoded into Location header or XHTML form control targeted at SAML requester
7. <ArtifactResolve> message sent by SAML requester directly to SAML responder
8. <ArtifactResponse> message returned by SAML responder containing original SAML response message inside
9. HTTP response sent to user agent from SAML requester upon completion of SAML exchange
X.1141(06)_F10-4

Figure 10-4/X.1141 – HTTP artifact message exchange

1) Initially, the user agent makes an arbitrary HTTP request to a system entity. In the course of processing the request, the system entity decides to initiate a SAML protocol exchange.

2) The system entity acting as a SAML requester responds to an HTTP request from the user agent by returning an artifact representing a SAML request.

   - If URL-encoded, the artifact is returned encoded into the HTTP response's Location header, and the HTTP status must be either 303 or 302. The SAML requester may include additional presentation and content in the HTTP response to facilitate the user agent's transmission of the message, as defined in IETF RFC 2616. The user agent delivers the artifact by issuing an HTTP GET request to the SAML responder.
   - If form-encoded, then the artifact is returned in an XHTML document containing the form and content defined in 10.2.6.3.3. The user agent delivers the artifact by issuing an HTTP POST request to the SAML responder.

3) The SAML responder determines the SAML requester by examining the artifact (the exact process depends on the type of artifact), and issues a `<samlp:ArtifactResolve>` request containing the artifact to the SAML requester using a direct SAML binding, temporarily reversing roles.

4) Assuming the necessary conditions are met, the SAML requester returns a `<samlp:ArtifactResponse>` containing the original SAML request message it wishes the SAML responder to process.

5) In general, the SAML responder may respond to the SAML request by immediately returning a SAML artifact or may return arbitrary content to facilitate subsequent interaction with the user agent necessary to fulfil the request. Specific protocols and profiles may include mechanisms to indicate the requester's level of willingness to permit this kind of interaction (for example, the `IsPassive` attribute in `<samlp:AuthnRequest>`).

6) Eventually the responder should return a SAML artifact to the user agent to be returned to the SAML requester. The SAML response artifact is returned in the same fashion as described for the SAML request artifact in step 2.

7) The SAML requester determines the SAML responder by examining the artifact, and issues a `<samlp:ArtifactResolve>` request containing the artifact to the SAML responder using a direct SAML binding, as in step 3.

   NOTE (informative) – PE31 (see OASIS PE:2006) suggests to replace the last sentence of step 6 with:

   The SAML requester determines the SAML responder by examining the artifact, and issues a `<samlp:ArtifactResolve>` request containing the artifact to the SAML responder using a synchronous SAML binding, as in step 3.

8) Assuming the necessary conditions are met, the SAML responder returns a `<samlp:ArtifactResponse>` containing the SAML response message it wishes the requester to process, as in step 4.

9) Upon receiving the SAML response, the SAML requester returns an arbitrary HTTP response to the user agent.

10.2.6.5.1 HTTP and caching considerations

HTTP proxies and the user agent intermediary should not cache SAML artifacts. To ensure this, the following rules should be followed.

When returning SAML artifacts using HTTP 1.1, HTTP responders should:

- include a `Cache-Control` header field set to "`no-cache`, `no-store`".
- include a `Pragma` header field set to "`no-cache`".

There are no other restrictions on the use of HTTP headers.

10.2.6.5.2 Security considerations

This binding uses a combination of indirect transmission of a message reference followed by a direct exchange to return the actual message. As a result, the message reference (artifact) need not itself be authenticated or integrity protected, but the callback request/response exchange that returns the actual message may be mutually authenticated and integrity protected, depending on the environment of use.

If the actual SAML protocol message is intended for a specific recipient, then the artifact's issuer must authenticate the sender of the subsequent `<samlp:ArtifactResolve>` message before returning the actual message.

The transmission of an artifact to and from the user agent should be protected with confidentiality; or TLS 1.0 should be used. The callback request/response exchange that returns the actual message may be protected, depending on the environment of use.

In general, this binding relies on the artifact as a hard-to-forge short-term reference and applies other security measures to the callback request/response that returns the actual message. All artifacts must have a single-use semantic enforced by the artifact issuer.

Furthermore, it is recommended that artifact receivers also enforce a single-use semantic on the artifact values they receive, to prevent an attacker from interfering with the resolution of an artifact by a user agent and then resubmitting it to the artifact receiver. If an attempt to resolve an artifact does not complete successfully, the artifact should be placed into a blocked artifact list for a period of time that exceeds a reasonable acceptance period during which the artifact issuer would resolve the artifact.

There is no mechanism defined to protect the integrity of the relationship between the artifact and the "RelayState" value, if any. That is, an attacker can potentially recombine a pair of valid HTTP responses by switching the "RelayState" values associated with each artifact. As a result, the producer/consumer of "RelayState" information must take care not to associate sensitive state information with the "RelayState" value without taking additional precautions (such as based on the information in the SAML protocol message retrieved via artifact).

10.2.6.6 Error reporting

A SAML responder that refuses to perform a message exchange with the SAML requester should return a response message with a second-level `<samlp:StatusCode>` value of `urn:oasis:names:tc:SAML:2.0:status:RequestDenied`.

HTTP interactions during the message exchange must not use HTTP error status codes to indicate failures in SAML processing, since the user agent is not a full party to the SAML protocol exchange.

If the issuer of an artifact receives a `<samlp:ArtifactResolve>` message that it can understand, it must return a `<samlp:ArtifactResponse>` with a `<samlp:StatusCode>` value of `urn:oasis:names:tc:SAML:2.0:status:Success`, even if it does not return the corresponding message (for example because the artifact requester is not authorized to receive the message or the artifact is no longer valid).

10.2.6.7 Metadata considerations

Support for the HTTP Artifact binding should be reflected by indicating URL endpoints at which requests and responses for a particular protocol or profile should be sent. Either a single endpoint or distinct request and response endpoints may be supplied. One or more indexed endpoints for processing `<samlp:ArtifactResolve>` messages should also be described.

10.2.6.8 Example SAML message exchange using HTTP Artifact

In this example, a `<LogoutRequest>` and `<LogoutResponse>` message pair is exchanged using the HTTP Artifact binding, with the artifact resolution taking place using the SOAP binding bound to HTTP.

First, here are the actual SAML protocol messages being exchanged:

```
<samlp:LogoutRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="d2b7c388cec36fa7c39c28fd298644a8" IssueInstant="2004-01-
21T19:00:49Z" Version="2.0">
    <Issuer>https://IdentityProvider.com/SAML</Issuer>
    <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">005a06e0-ad82-110d-a556-004005b13a2b</NameID>
    <samlp:SessionIndex>1</samlp:SessionIndex>
</samlp:LogoutRequest>

<samlp:LogoutResponse xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="b0730d21b628110d8b7e004005b13a2b"
InResponseTo="d2b7c388cec36fa7c39c28fd298644a8"
    IssueInstant="2004-01-21T19:00:49Z" Version="2.0">
    <Issuer>https://ServiceProvider.com/SAML</Issuer>
    <samlp:Status>
        <samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
</samlp:LogoutResponse>
```

The initial HTTP request from the user agent in the above is not defined by this binding. To initiate the logout protocol exchange, the SAML requester returns the following HTTP response, containing a SAML artifact. Line feeds in the HTTP Location header below are a result of document formatting, and there are no line feeds in the actual header value.

```
HTTP/1.1 302 Object Moved
Date: 21 Jan 2004 07:00:49 GMT
Location:
https://ServiceProvider.com/SAML/SLO/Browser?SAMLart=AAQAADWNEw5VT47wcO4zX%
2FiEzMmFQvGknDfws2ZtqSGdkNSbsW1cmVR0bzU%3D&RelayState=0043bfc1bc45110dae170
04005b13a2b
Content-Type: text/html; charset=iso-8859-1
```

The SAML responder then resolves the artifact it received into the actual SAML request using the Artifact Resolution protocol and the SOAP binding in steps 3 and 4, as follows:

Step 3:

```
POST /SAML/Artifact/Resolve HTTP/1.1
Host: IdentityProvider.com
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <samlp:ArtifactResolve
             xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
             xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
             ID="_6c3a4f8b9c2d" Version="2.0"
             IssueInstant="2004-01-21T19:00:49Z">
             <Issuer>https://ServiceProvider.com/SAML</Issuer>
             <Artifact>
             AAQAADWNEw5VT47wcO4zX/iEzMmFQvGknDfws2ZtqSGdkNSbsW1cmVR0bzU=
             </Artifact>
        </samlp:ArtifactResolve>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Step 4:

```
HTTP/1.1 200 OK
Date: 21 Jan 2004 07:00:49 GMT
Content-Type: text/xml
Content-Length: nnnn

<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <samlp:ArtifactResponse
             xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
             xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
             ID="_FQvGknDfws2Z" Version="2.0"
             InResponseTo="_6c3a4f8b9c2d"
             IssueInstant="2004-01-21T19:00:49Z">
             <Issuer>https://IdentityProvider.com/SAML</Issuer>
             <samlp:Status>
                    <samlp:StatusCode
             Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
             </samlp:Status>
             <samlp:LogoutRequest ID="d2b7c388cec36fa7c39c28fd298644a8"
                    IssueInstant="2004-01-21T19:00:49Z"
                    Version="2.0">
                    <Issuer>https://IdentityProvider.com/SAML</Issuer>
                    <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">005a06e0-ad82-110d-a556-004005b13a2b</NameID>
                    <samlp:SessionIndex>1</samlp:SessionIndex>
             </samlp:LogoutRequest>
        </samlp:ArtifactResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

After any unspecified interactions may have taken place, the SAML responder returns a second SAML artifact in its HTTP response in step 6:

```
HTTP/1.1 302 Object Moved
Date: 21 Jan 2004 07:05:49 GMT
Location:
https://IdentityProvider.com/SAML/SLO/Response?SAMLart=AAQAAFGIZXv5%2BQaBaE
5qYurHWJO1nAgLAsqfnyiDHIggbFU0mlSGFTyQiPc%3D&RelayState=0043bfc1bc45110dae1
7004005b13a2b
Content-Type: text/html; charset=iso-8859-1
```

The SAML responder then resolves the artifact it received into the actual SAML request using the Artifact Resolution protocol and the SOAP binding in steps 7 and 8, as follows:

Step 7:

```
POST /SAML/Artifact/Resolve HTTP/1.1
Host: ServiceProvider.com
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <samlp:ArtifactResolve
             xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
             xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
             ID="_ec36fa7c39" Version="2.0"
             IssueInstant="2004-01-21T19:05:49Z">
             <Issuer>https://IdentityProvider.com/SAML</Issuer>
             <Artifact>
             AAQAAFGIZXv5+QaBaE5qYurHWJO1nAgLAsqfnyiDHIggbFU0mlSGFTyQiPc=
             </Artifact>
        </samlp:ArtifactResolve>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Step 8:

```
HTTP/1.1 200 OK
Date: 21 Jan 2004 07:05:49 GMT
Content-Type: text/xml
Content-Length: nnnn

<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <samlp:ArtifactResponse
             xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
             xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
             ID="_FQvGknDfws2Z" Version="2.0"
             InResponseTo="_ec36fa7c39"
             IssueInstant="2004-01-21T19:05:49Z">
             <Issuer>https://ServiceProvider.com/SAML</Issuer>
             <samlp:Status>
                    <samlp:StatusCode
             Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
             </samlp:Status>
             <samlp:LogoutResponse ID="_b0730d21b628110d8b7e004005b13a2b"
                    InResponseTo="_d2b7c388cec36fa7c39c28fd298644a8"
                    IssueInstant="2004-01-21T19:05:49Z"
                    Version="2.0">
                    <Issuer>https://ServiceProvider.com/SAML</Issuer>
                    <samlp:Status>
                           <samlp:StatusCode
             Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
                    </samlp:Status>
             </samlp:LogoutResponse>
        </samlp:ArtifactResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.2.7 SAML URI binding

URIs are a protocol-independent means of referring to a resource. This binding is not a general SAML request/response binding, but rather supports the encapsulation of a `<samlp:AssertionIDRequest>` message with a single `<saml:AssertionIDRef>` into the resolution of a URI. The result of a successful request is a SAML `<saml:Assertion>` element (but not a complete SAML response).

Like SOAP, URI resolution can occur over multiple underlying transports. This binding has transport-independent aspects, but also calls out the use of HTTP with TLS 1.0 as required (mandatory to implement).

NOTE (informative) – PE24 (see OASIS PE:2006) suggests to replace the above paragraph with the following:

Like SOAP, URI resolution can occur over multiple underlying transports. This binding has protocol-independent aspects, but also calls out as mandatory the implementation of HTTP URIs.

10.2.7.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:bindings:URI`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None

10.2.7.2 Protocol-independent aspects of the SAML URI binding

The following clauses define aspects of the SAML URI binding that are independent of the underlying transport protocol of the URI resolution process.

A SAML URI reference identifies a specific SAML assertion. The result of resolving the URI must be a message containing the assertion, or a transport-specific error. The specific format of the message depends on the underlying transport protocol. If the transport protocol permits the returned content to be described, such as HTTP 1.1, then the assertion may be encoded in whatever format is permitted. If not, the assertion must be returned in a form which can be unambiguously interpreted as or transformed into an XML serialization of the assertion.

It must be the case that if the same URI reference is resolved in the future, then either the same SAML assertion, or an error, is returned. That is, the reference may be persistent but must consistently reference the same assertion, if any.

10.2.7.3 Security considerations

Indirect use of a SAML assertion presents dangers if the binding of the reference to the result is not secure. The particular threats and their severity depend on the use to which the assertion is being put. In general, the result of resolving a URI reference to a SAML assertion should only be trusted if the requester can be certain of the identity of the responder and that the contents have not been modified in transit.

It is often not sufficient that the assertion itself be signed, because URI references are by their nature somewhat opaque to the requester. The requester should have independent means to ensure that the assertion returned is actually the one that is represented by the URI; this is accomplished by both authenticating the responder and relying on the integrity of the response.

10.2.7.4 MIME encapsulation

For resolution protocols that support MIME as a content description and packaging mechanism, the resulting assertion should be returned as a MIME entity of type `application/samlassertion+xml`, as defined in Appendix II.

10.2.7.5 Use of HTTP URIs

A SAML authority that claims conformance to the SAML URI binding must implement support for HTTP. This clause describes certain specifics of using HTTP URIs, including URI syntax, HTTP headers, and error reporting.

10.2.7.5.1 URI syntax

In general, there are no restrictions on the permissible syntax of a SAML URI reference as long as the SAML authority responsible for the reference creates the message containing it. However, authorities must support a URL endpoint at which an HTTP request can be sent with a single query string parameter named `ID`. There must be no query string in the endpoint URL itself independent of this parameter.

For example, if the documented endpoint at an authority is "https://saml.example.edu/assertions", a request for an assertion with an `ID` of `abcde` can be sent to:

```
https://saml.example.edu/assertions?ID=abcde
```

The use of wildcards is not allowed for such ID queries.

NOTE (informative) – PE31 (see OASIS PE:2006) suggests to replace the above text with:

Note that the URI syntax does not support the use of wildcards in such queries.

10.2.7.5.2 HTTP and caching considerations

HTTP proxies must not cache SAML assertions. To ensure this, the following rules should be followed.

When returning SAML assertions using HTTP 1.1, HTTP responders should:

- include a `Cache-Control` header field set to "`no-cache`, `no-store`".
- include a `Pragma` header field set to "`no-cache`".

10.2.7.5.3 Security considerations

IETF RFC 2617 describes possible attacks in the HTTP environment when basic or message-digest authentication schemes are used.

Use of TLS 1.0 is strongly recommended as a means of authentication, integrity protection and confidentiality.

10.2.7.5.4 Error reporting

As an HTTP protocol exchange, the appropriate HTTP status code should be used to indicate the result of a request. For example, a SAML responder that refuses to perform a message exchange with the SAML requester should return a "`403 Forbidden`" response. If the assertion specified is unknown to the responder, then a "`404 Not Found`" response should be returned. In these cases, the content of the HTTP body is not significant.

10.2.7.5.5 Metadata considerations

Support for the URI binding over HTTP should be reflected by indicating a URL endpoint at which requests for arbitrary assertions are to be sent.

10.2.7.5.6 Example SAML message exchange using an HTTP URI

Following is an example of a request for an assertion.

```
GET /SamlService?ID=abcde HTTP/1.1
Host: www.example.com
```

Following is an example of the corresponding response, which supplies the requested assertion.

```
HTTP/1.1 200 OK
Content-Type: application/samlassertion+xml
Cache-Control: no-cache, no-store
Pragma: no-cache
Content-Length: nnnn

<saml:Assertion ID="abcde" ...>
...
</saml:Assertion>
```

11 Profiles for SAML

This clause specifies profiles that define the use of SAML assertions and request-response messages in communications protocols and frameworks, as well as profiles that define SAML attribute value syntax and naming conventions.

11.1 Profile concepts

One type of SAML profile outlines a set of rules describing how to embed SAML assertions into and extract them from a framework or protocol. Such a profile describes how SAML assertions are embedded in or combined with other objects (for example, files of various types, or protocol data units of communication protocols) by an originating party, communicated from the originating party to a receiving party, and subsequently processed at the destination. A particular set of rules for embedding SAML assertions into and extracting them from a specific class of `<FOO>` objects is termed a *<FOO> profile of SAML*.

For example, a SOAP profile of SAML describes how SAML assertions can be added to SOAP messages, how SOAP headers are affected by SAML assertions, and how SAML-related error states should be reflected in SOAP messages.

Another type of SAML profile defines a set of constraints on the use of a general SAML protocol or assertion capability for a particular environment or context of use. Profiles of this nature may constrain optionality, require the use of specific SAML functionality (for example, attributes, conditions, or bindings), and in other respects define the processing rules to be followed by profile actors.

A particular example of the latter are those that address SAML attributes. The SAML `<Attribute>` element provides a great deal of flexibility in attribute naming, value syntax, and including in-band metadata through the use of XML attributes. Interoperability is achieved by constraining this flexibility when warranted by adhering to profiles that define how to use these elements with greater specificity than the generic rules defined in clause 8.

Attribute profiles provide the definitions necessary to constrain SAML attribute expression when dealing with particular types of attribute information or when interacting with external systems or other open standards that require greater strictness.

The intent of this Recommendation is to specify a selected set of profiles of various kinds in sufficient detail to ensure that independently implemented products will interoperate.

11.2 Specification of additional profiles

This Recommendation defines a selected set of profiles, but others will possibly be developed in the future. The following subclauses offer guidelines for specifying profiles.

11.2.1 Guidelines for specifying profiles

This clause provides a checklist of issues that must be addressed by each profile.

1) Specify a URI that uniquely identifies the profile, postal or electronic contact information for the author, and provide reference to previously defined profiles that the new profile updates or obsoletes.
2) Describe the set of interactions between parties involved in the profile. Any restrictions on applications used by each party and the protocols involved in each interaction must be explicitly called out.
3) Identify the parties involved in each interaction, including how many parties are involved and whether intermediaries may be involved.
4) Specify the method of authentication of parties involved in each interaction, including whether authentication is required and acceptable authentication types.
5) Identify the level of support for message integrity, including the mechanisms used to ensure message integrity.
6) Identify the level of support for confidentiality, including whether a third party may view the contents of SAML messages and assertions, whether the profile requires confidentiality, and the mechanisms recommended for achieving confidentiality.
7) Identify the error states, including the error states at each participant, especially those that receive and process SAML assertions or messages.
8) Identify security considerations, including analysis of threats and description of countermeasures.
9) Identify SAML confirmation method identifiers defined and/or utilized by the profile.
10) Identify relevant SAML metadata defined and/or utilized by the profile.

11.2.2 Guidelines for specifying attribute profiles

This clause provides a checklist of items that must in particular be addressed by attribute profiles.

1) Specify a URI that uniquely identifies the profile, postal or electronic contact information for the author, and provide reference to previously defined profiles that the new profile updates or obsoletes.
2) Syntax and restrictions on the acceptable values of the `NameFormat` and `Name` attributes of SAML `<Attribute>` elements.
3) Any additional namespace-qualified XML attributes defined by the profile that may be used in SAML `<Attribute>` elements.
4) Rules for determining the equality of SAML `<Attribute>` elements as defined by the profile, for use when processing attributes, queries, etc.
5) Syntax and restrictions on values acceptable in the SAML `<AttributeValue>` element, including whether the `xsi:type` XML attribute can or should be used.

11.3 Confirmation method identifiers

Clause 8 defines the <SubjectConfirmation> element as a Method plus optional <SubjectConfirmationData>. The <SubjectConfirmation> element should be used by the relying party to confirm that the request or message came from a system entity that is associated with the subject of the assertion, within the context of a particular profile.

The Method attribute indicates the specific method that the relying party should use to make this determination. This may or may not have any relationship to an authentication that was performed previously. Unlike the authentication context, the subject confirmation method will often be accompanied by additional information, such as a certificate or key, in the <SubjectConfirmationData> element that will allow the relying party to perform the necessary verification. A common set of attributes is also defined and may be used to constrain the conditions under which the verification can take place.

It is anticipated that profiles will define and use several different values for <ConfirmationMethod>, each corresponding to a different SAML usage scenario. The following methods are defined for use by profiles defined within this Recommendation and other profiles that find them useful.

11.3.1 Holder of key

```
URI: urn:oasis:names:tc:SAML:2.0:cm:holder-of-key
```

One or more <ds:KeyInfo> elements must be present within the <SubjectConfirmationData> element. An xsi:type attribute may be present in the <SubjectConfirmationData> element and, if present, must be set to **saml:KeyInfoConfirmationDataType** (the namespace prefix is arbitrary but must reference the SAML assertion namespace).

As described in W3C Signature, each <ds:KeyInfo> element holds a key or information that enables an application to obtain a key. The holder of a specified key is considered to be the subject of the assertion by the asserting party.

In accordance with W3C Signature, each <ds:KeyInfo> element must identify a single cryptographic key. Multiple keys may be identified with separate <ds:KeyInfo> elements, such as when different confirmation keys are needed for different relying parties.

Example: The holder of the key named "By-Tor" or the holder of the key named "Snow Dog" can confirm itself as the subject.

```
<SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
      <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
            <ds:KeyInfo>
                  <ds:KeyName>By-Tor</ds:KeyName>
            </ds:KeyInfo>
            <ds:KeyInfo>
                  <ds:KeyName>Snow Dog</ds:KeyName>
            </ds:KeyInfo>
      </SubjectConfirmationData>
</SubjectConfirmation>
```

11.3.2 Sender vouches

```
URI: urn:oasis:names:tc:SAML:2.0:cm:sender-vouches
```

Indicates that no other information is available about the context of use of the assertion. The relying party should utilize other means to determine if it should process the assertion further, subject to optional constraints on confirmation using the attributes that may be present in the <SubjectConfirmationData> element.

11.3.3 Bearer

```
URI: urn:oasis:names:tc:SAML:2.0:cm:bearer
```

The subject of the assertion is the bearer of the assertion, subject to optional constraints on confirmation using the attributes that may be present in the <SubjectConfirmationData> element, as defined in clause 8.

Example: The bearer of the assertion can confirm itself as the subject, provided the assertion is delivered in a message sent to "https://www.serviceprovider.com/saml/consumer" before 1:37 PM GMT on March 19th, 2004, in response to a request with ID "_1234567890".

```
<SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
      <SubjectConfirmationData InResponseTo="_1234567890"
            Recipient="https://www.serviceprovider.com/saml/consumer"
            NotOnOrAfter="2004-03-19T13:27:00Z"
      </SubjectConfirmationData>
</SubjectConfirmation>
```

11.4 SSO Profiles of SAML

A set of profiles is defined to support single sign-on (SSO) of browsers and other client devices.

- A web browser-based profile of the Authentication Request protocol in clause 8 is defined to support web single sign-on.
- An additional web SSO profile is defined to support enhanced clients.
- A profile of the Single Logout and Name Identifier Management protocols in clause 8 is defined over both front-channel (browser) and back-channel bindings.
- An additional profile is defined for identity provider discovery using cookies.

11.4.1 Web browser SSO profile

In the scenario supported by the web browser SSO profile, a web user either accesses a resource at a service provider, or accesses an identity provider such that the service provider and desired resource are understood or implicit. The web user authenticates (or has already authenticated) to the identity provider, which then produces an authentication assertion (possibly with input from the service provider) and the service provider consumes the assertion to establish a security context for the web user. During this process, a name identifier might also be established between the providers for the principal, subject to the parameters of the interaction and the consent of the parties.

To implement this scenario, a profile of the SAML Authentication Request protocol is used, in conjunction with the HTTP Redirect, HTTP POST and HTTP Artifact bindings.

It is assumed that the user is using a standard commercial browser and can authenticate to the identity provider by some means outside the scope of SAML.

11.4.1.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:SSO:browser`

Contact information: security-services-comment@lists.oasis-open.org

SAML confirmation method identifiers: The SAML V2.0 "bearer" confirmation method identifier, `urn:oasis:names:tc:SAML:2.0:cm:bearer`, is used by this profile.

Description: Given below.

Updates: None.

11.4.1.2 Profile overview

Figure 11-1 illustrates the basic template for achieving SSO. The following steps are described by the profile. Within an individual step, there may be one or more actual message exchanges depending on the binding used for that step and other implementation-dependent behaviour.

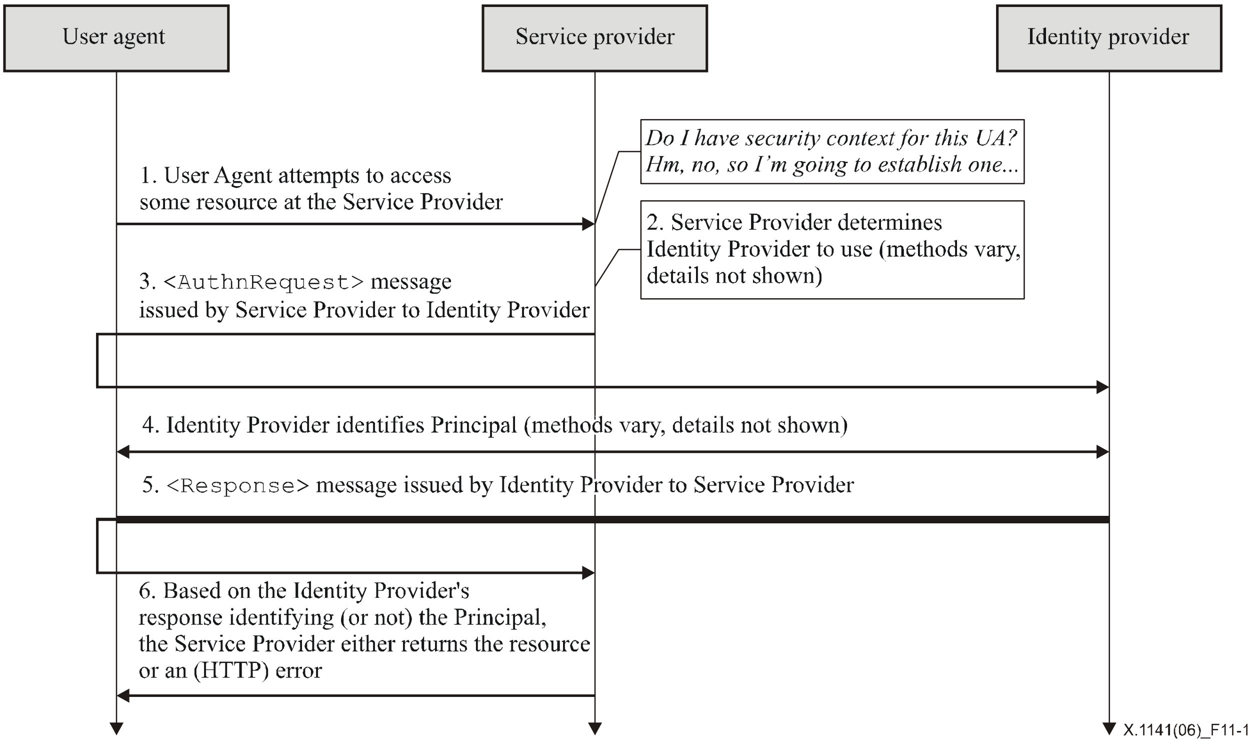
User agent
Service provider
Identity provider
Do I have security context for this UA? Hm, no, so I'm going to establish one...
1. User Agent attempts to access some resource at the Service Provider
2. Service Provider determines Identity Provider to use (methods vary, details not shown)
3. <AuthnRequest> message issued by Service Provider to Identity Provider
4. Identity Provider identifies Principal (methods vary, details not shown)
5. <Response> message issued by Identity Provider to Service Provider
6. Based on the Identity Provider's response identifying (or not) the Principal, the Service Provider either returns the resource or an (HTTP) error
X.1141(06)_F11-1

Figure 11-1/X.1141 – Basic template for achieving SSO

1) **HTTP request to service provider**

In step 1, the principal, via an HTTP User Agent, makes an HTTP request for a secured resource at the service provider without a security context.

2) **Service provider determines identity provider**

In step 2, the service provider obtains the location of an endpoint at an identity provider for the authentication request protocol that supports its preferred binding. The means by which this is accomplished is implementation-dependent. The service provider may use the SAML identity provider discovery profile described in 8.7.4.

3) **<AuthnRequest> issued by service provider to identity provider**

In step 3, the service provider issues an `<AuthnRequest>` message to be delivered by the user agent to the identity provider. Either the HTTP Redirect, HTTP POST, or HTTP Artifact binding can be used to transfer the message to the identity provider through the user agent.

4) **Identity provider identifies principal**

In step 4, the principal is identified by the identity provider by some means outside the scope of this profile. This may require a new act of authentication, or it may reuse an existing authenticated session.

5) **Identity provider issues <Response> to Service Provider**

In step 5, the identity provider issues a `<Response>` message to be delivered by the user agent to the service provider. Either the HTTP POST, or HTTP Artifact binding can be used to transfer the message to the service provider through the user agent. The message may indicate an error, or will include (at least) an authentication assertion. The HTTP Redirect binding must not be used, as the response will typically exceed the URL length permitted by most user agents.

6) **Service provider grants or denies access to principal**

In step 6, having received the response from the identity provider, the service provider can respond to the principal's user agent with its own error, or can establish its own security context for the principal and return the requested resource.

An identity provider can initiate this profile at step 5 and issue a `<Response>` message to a service provider without the preceding steps.

11.4.1.3 Profile description

If the profile is initiated by the service provider, start with clause 11.4.1.3.1. If initiated by the identity provider, start with clause 11.4.1.3.5. In the descriptions below, the following are referred to:

Single sign-on service

This is the authentication request protocol endpoint at the identity provider to which the <AuthnRequest> message (or artifact representing it) is delivered by the user agent.

Assertion consumer service

This is the authentication request protocol endpoint at the service provider to which the <Response> message (or artifact representing it) is delivered by the user agent.

11.4.1.3.1 HTTP request to service provider

If the first access is to the service provider, an arbitrary request for a resource can initiate the profile. There are no restrictions on the form of the request. The service provider is free to use any means it wishes to associate the subsequent interactions with the original request. Each of the bindings provide a RelayState mechanism that the service provider may use to associate the profile exchange with the original request. The service provider should reveal as little of the request as possible in the RelayState value unless the use of the profile does not require such privacy measures.

11.4.1.3.2 Service provider determines identity provider

This step is implementation-dependent. The service provider may use the SAML identity provider discovery profile, described in 11.4.3. The service provider may also choose to redirect the user agent to another service that is able to determine an appropriate identity provider. In such a case, the service provider may issue an <AuthnRequest> (as in the next step) to this service to be relayed to the identity provider, or it may rely on the intermediary service to issue an <AuthnRequest> message on its behalf.

11.4.1.3.3 <AuthnRequest> is issued by service provider to identity provider

Once an identity provider is selected, the location of its single sign-on service is determined, based on the SAML binding chosen by the service provider for sending the <AuthnRequest>. Metadata may be used for this purpose. In response to an HTTP request by the user agent, an HTTP response is returned containing an <AuthnRequest> message or an artifact, depending on the SAML binding used, to be delivered to the identity provider's single sign-on service.

The exact format of this HTTP response and the subsequent HTTP request to the single sign-on service is defined by the SAML binding used. Profile-specific rules for the contents of the <AuthnRequest> message are included in 11.4.1.4.1. If the HTTP Redirect or POST binding is used, the <AuthnRequest> message is delivered directly to the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile defined in 11.4.6 is used by the identity provider, which makes a callback to the service provider to retrieve the <AuthnRequest> message, using, for example, the SOAP binding.

It is recommended that the HTTP exchanges in this step be made over TLS 1.0 to maintain confidentiality and message integrity. The <AuthnRequest> message may be signed, if authentication of the request issuer is required. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.

The identity provider must process the <AuthnRequest> message as described in this Recommendation. This may constrain the subsequent interactions with the user agent, for example if the IsPassive attribute is included.

11.4.1.3.4 Identity provider identifies principal

At any time during the previous step or subsequent to it, the identity provider must establish the identity of the principal (unless it returns an error to the service provider). The ForceAuthn <AuthnRequest> attribute, if present with a value of true, obligates the identity provider to freshly establish this identity, rather than relying on an existing session it may have with the principal. Otherwise, and in all other respects, the identity provider may use any means to authenticate the user agent, subject to any requirements included in the <AuthnRequest> in the form of the <RequestedAuthnContext> element.

11.4.1.3.5 Identity provider issues <Response> to service provider

Regardless of the success or failure of the <AuthnRequest>, the identity provider should produce an HTTP response to the user agent containing a <Response> message or an artifact, depending on the SAML binding used, to be delivered to the service provider's assertion consumer service.

The exact format of this HTTP response and the subsequent HTTP request to the assertion consumer service is defined by the SAML binding used. Profile-specific rules on the contents of the `<Response>` are included in 11.4.1.4.2. If the HTTP POST binding is used, the `<Response>` message is delivered directly to the service provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile defined in 11.4.6 is used by the service provider, which makes a callback to the identity provider to retrieve the `<Response>` message, using for example the SOAP binding.

The location of the assertion consumer service may be determined using metadata. The identity provider must have some means to establish that this location is in fact controlled by the service provider. A service provider may indicate the SAML binding and the specific assertion consumer service to use in its `<AuthnRequest>` and the identity provider must honour them if it can.

It is recommended that the HTTP requests in this step be made over TLS 1.0 to maintain confidentiality and message integrity. The `<Assertion>` element(s) in the `<Response>` must be signed, if the HTTP POST binding is used, and may be signed if the HTTP-Artifact binding is used.

The service provider must process the `<Response>` message and any enclosed <Assertion> elements as described in this Recommendation.

11.4.1.3.6 Service provider grants or denies access to user agent

To complete the profile, the service provider processes the `<Response>` and `<Assertion>(`s) and grants or denies access to the resource. The service provider may establish a security context with the user agent using any session mechanism it chooses. Any subsequent use of the `<Assertion>`(s) provided are at the discretion of the service provider and other relying parties, subject to any restrictions on use contained within them.

11.4.1.4 Use of authentication request protocol

This profile is based on the Authentication Request protocol defined in this Recommendation. Here, the service provider is the request issuer and the relying party, and the principal is the presenter, requested subject, and confirming entity. There may be additional relying parties or confirming entities at the discretion of the identity provider.

11.4.1.4.1 <AuthnRequest> usage

A service provider may include any message content as described in this Recommendation. All processing rules are as defined in this Recommendation. The `<Issuer>` element must be present and must contain the unique identifier of the requesting service provider; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

If the identity provider cannot or will not satisfy the request, it must respond with a `<Response>` message containing an appropriate error status code or codes.

If the service provider wishes to permit the identity provider to establish a new identifier for the principal if none exists, it must include a `<NameIDPolicy>` element with the `AllowCreate` attribute set to "true". Otherwise, only a principal for whom the identity provider has previously established an identifier usable by the service provider can be authenticated successfully.

The service provider may include a `<Subject>` element in the request that names the actual identity about which it wishes to receive an assertion. This element must not contain any `<SubjectConfirmation>` elements. If the identity provider does not recognize the principal as that identity, then it must respond with a `<Response>` message containing an error status and no assertions.

The `<AuthnRequest>` message may be signed (as directed by the SAML binding used). If the HTTP Artifact binding is used, authentication of the parties is optional and any mechanism permitted by the binding may be used.

If the `<AuthnRequest>` is not authenticated and/or integrity protected, the information in it must not be trusted except as advisory. Whether the request is signed or not, the identity provider must ensure that any `<AssertionConsumerServiceURL>` or `<AssertionConsumerServiceIndex>` elements in the request are verified as belonging to the service provider to whom the response will be sent. Failure to do so can result in a man-in-the-middle attack.

11.4.1.4.2 <Response> usage

NOTE 1 (informative) – PE26 (see OASIS PE:2006) offers clarification for the intent of this clause, see Appendix VIII for more details.

If the identity provider wishes to return an error, it must not include any assertions in the `<Response>` message. Otherwise, if the request is successful (or if the response is not associated with a request), the `<Response>` element must conform to the following:

- The `<Issuer>` element may be omitted, but if present it must contain the unique identifier of the issuing identity provider; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

  NOTE 2 (informative) – PE17 (see OASIS PE:2006) suggests to replace the above paragraph with:

  If the `<Response>` message is signed or if an enclosed assertion is encrypted, then the `<Issuer>` element must be present. Otherwise, it may be omitted. If present it must contain the unique identifier of the issuing identity provider; the Format attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

- It must contain at least one `<Assertion>`. Each assertion's `<Issuer>` element must contain the unique identifier of the issuing identity provider; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.
- The set of one or more assertions must contain at least one `<AuthnStatement>` that reflects the authentication of the principal to the identity provider.
- At least one assertion containing an `<AuthnStatement>` must contain a `<Subject>` element with at least one `<SubjectConfirmation>` element containing a `Method` of `urn:oasis:names:tc:SAML:2.0:cm:bearer`. If the identity provider supports the Single Logout profile, defined in 11.4.4, any such authentication statements must include a `SessionIndex` attribute to enable per-session logout requests by the service provider.
- The bearer `<SubjectConfirmation>` element described above must contain a `<SubjectConfirmationData>` element that contains a `Recipient` attribute containing the service provider's assertion consumer service URL and a `NotOnOrAfter` attribute that limits the window during which the assertion can be delivered. It may contain an `Address` attribute limiting the client address from which the assertion can be delivered. It must not contain a `NotBefore` attribute. If the containing message is in response to an `<AuthnRequest>`, then the `InResponseTo` attribute must match the request's `ID`.
- Other statements and confirmation methods may be included in the assertion(s) at the discretion of the identity provider. In particular, `<AttributeStatement>` elements may be included. The `<AuthnRequest>` may contain an `AttributeConsumingServiceIndex` XML attribute referencing information about desired or required attributes as in clause 9. The identity provider may ignore this, or send other attributes at its discretion.
- The assertion(s) containing a bearer subject confirmation must contain an `<AudienceRestriction>` including the service provider's unique identifier as an `<Audience>`.
- Other conditions (and other `<Audience>` elements) may be included as requested by the service provider or at the discretion of the identity provider. (Of course, all such conditions must be understood by and accepted by the service provider in order for the assertion to be considered valid.) The identity provider is not obligated to honour the requested set of `<Conditions>` in the `<AuthnRequest>`, if any.

11.4.1.4.3 <Response> message processing rules

NOTE (informative) – PE26 (see OASIS PE:2006) offers clarification for the intent of this subclause, see Appendix VIII for more details.

Regardless of the SAML binding used, the service provider must do the following:

- Verify any signatures present on the assertion(s) or the response.
- Verify that the `Recipient` attribute in any bearer `<SubjectConfirmationData>` matches the assertion consumer service URL to which the `<Response>` or artifact was delivered.
- Verify that the `NotOnOrAfter` attribute in any bearer `<SubjectConfirmationData>` has not passed, subject to allowable clock skew between the providers.
- Verify that the `InResponseTo` attribute in the bearer `<SubjectConfirmationData>` equals the `ID` of its original `<AuthnRequest>` message, unless the response is unsolicited in which case the attribute must not be present.
- Verify that any assertions relied upon are valid in other respects.
- If any bearer `<SubjectConfirmationData>` includes an `Address` attribute, the service provider may check the user agent's client address against it.

- Any assertion which is not valid, or whose subject confirmation requirements cannot be met, should be discarded and should not be used to establish a security context for the principal.
- If an `<AuthnStatement>` used to establish a security context for the principal contains a `SessionNotOnOrAfter` attribute, the security context should be discarded once this time is reached, unless the service provider re-establishes the principal's identity by repeating the use of this profile.

11.4.1.4.4 Artifact-specific <Response> message processing rules

If the HTTP Artifact binding is used to deliver the `<Response>,` the dereferencing of the artifact using the Artifact Resolution profile must be mutually authenticated, integrity protected and confidential.

The identity provider must ensure that only the service provider to whom the `<Response>` message has been issued is given the message as the result of an `<ArtifactResolve>` request.

Either the SAML binding used to dereference the artifact or message signatures can be used to authenticate the parties and protect the messages.

11.4.1.4.5 POST-specific processing rules

NOTE (informative) – PE26 (see OASIS PE:2006) offers clarification for the intent of this subclause, see Appendix VIII for more details.

If the HTTP POST binding is used to deliver the `<Response>`, the enclosed assertion(s) must be signed.

The service provider must ensure that bearer assertions are not replayed, by maintaining the set of used `ID` values for the length of time for which the assertion would be considered valid based on the `NotOnOrAfter` attribute in the `<SubjectConfirmationData>`.

11.4.1.5 Unsolicited responses

An identity provider may initiate this profile by delivering an unsolicited `<Response>` message to a service provider.

An unsolicited `<Response>` must not contain an `InResponseTo` attribute, nor should any bearer `<SubjectConfirmationData>` elements contain one. If metadata is used, the `<Response>` or artifact should be delivered to the `<md:AssertionConsumerService>` endpoint of the service provider designated as the default.

Of special mention is that the identity provider may include a binding-specific "RelayState" parameter that indicates, based on mutual agreement with the service provider, how to handle subsequent interactions with the user agent. This may be the URL of a resource at the service provider. The service provider should be prepared to handle unsolicited responses by designating a default location to send the user agent subsequent to processing a response successfully.

11.4.1.6 Use of metadata

Clause 11.4.2.5 defines an endpoint element, `<md:SingleSignOnService>`, to describe supported bindings and location(s) to which a service provider may send requests to an identity provider using this profile.

The `<md:IDPSSODescriptor>` element's `WantAuthnRequestsSigned` attribute may be used by an identity provider to document a requirement that requests be signed. The `<md:SPSSODescriptor>` element's `AuthnRequestsSigned` attribute may be used by a service provider to document the intention to sign all of its requests.

The providers may document the key(s) used to sign requests, responses, and assertions with `<md:KeyDescriptor>` elements with a `use` attribute of `sign`. When encrypting SAML elements, `<md:KeyDescriptor>` elements with a `use` attribute of `encrypt` may be used to document supported encryption algorithms and settings, and public keys used to receive bulk encryption keys.

The indexed endpoint element `<md:AssertionConsumerService>` is used to describe supported bindings and location(s) to which an identity provider may send responses to a service provider using this profile. The `index` attribute is used to distinguish the possible endpoints that may be specified by reference in the `<AuthnRequest>` message. The `isDefault` attribute is used to specify the endpoint to use if not specified in a request.

The `<md:SPSSODescriptor>` element's `WantAssertionsSigned` attribute may be used by a service provider to document a requirement that assertions delivered with this profile be signed. This is in addition to any requirements for signing imposed by the use of a particular binding. The identity provider is not obligated by this, but is being made aware of the likelihood that an unsigned assertion will be insufficient.

If the request or response message is delivered using the HTTP Artifact binding, the artifact issuer must provide at least one `<md:ArtifactResolutionService>` endpoint element in its metadata.

The <md:IDPSSODescriptor> may contain <md:NameIDFormat>, <md:AttributeProfile>, and <saml:Attribute> elements to indicate the general ability to support particular name identifier formats, attribute profiles, or specific attributes and values. The ability to support any such features during a given authentication exchange is dependent on policy and the discretion of the identity provider.

The <md:SPSSODescriptor> element may also be used to document the service provider's need or desire for SAML attributes to be delivered along with authentication information. The actual inclusion of attributes is always at the discretion of the identity provider. One or more <md:AttributeConsumingService> elements may be included in its metadata, each with an index attribute to distinguish different services that may be specified by reference in the <AuthnRequest> message. The isDefault attribute is used to specify a default set of attribute requirements.

11.4.2 Enhanced client or proxy (ECP) profile

An *enhanced client or proxy* (ECP) is a system entity that knows how to contact an appropriate identity provider, possibly in a context-dependent fashion, and also supports the Reverse SOAP (PAOS) binding (see clause 10).

An example scenario enabled by this profile is as follows: A principal, wielding an ECP, uses it to either access a resource at a service provider, or access an identity provider such that the service provider and desired resource are understood or implicit. The principal authenticates (or has already authenticated) with the identity provider, which then produces an authentication assertion (possibly with input from the service provider). The service provider then consumes the assertion and subsequently establishes a security context for the principal. During this process, a name identifier might also be established between the providers for the principal, subject to the parameters of the interaction and the consent of the principal.

This profile is based on the SAML Authentication Request protocol in conjunction with the PAOS binding.

NOTE – The means by which a principal authenticates with an identity provider is outside of the scope of SAML.

11.4.2.1 Required information

Identification: urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp (this is also the target namespace assigned in the corresponding ECP profile schema in Annex A).

Contact information: security-services-comment@lists.oasis-open.org

SAML confirmation method identifiers: The SAML V2.0 "bearer" confirmation method identifier, urn:oasis:names:tc:SAML:2.0:cm:bearer, is used by this profile.

Description: Given below.

Updates: None.

11.4.2.2 Profile overview

As introduced above, the ECP profile specifies interactions between enhanced clients or proxies and service providers and identity providers. It is a specific application of the SSO profile described in 11.4.1. If not otherwise specified by this profile, and if not specific to the use of browser-based bindings, the rules specified in 11.4.1 must be observed.

An ECP is a client or proxy that satisfies the following two conditions:

- It has, or knows how to obtain, information about the identity provider that the principal associated with the ECP wishes to use, in the context of an interaction with a service provider.

This allows a service provider to make an authentication request to the ECP without the need to know or discover the appropriate identity provider (effectively bypassing step 2 of the SSO profile in 11.4.1).

- It is able to use a reverse SOAP (PAOS) binding as profiled here for an authentication request and response.

This enables a service provider to obtain an authentication assertion via an ECP that is not otherwise (i.e., outside of the context of the immediate interaction) necessarily directly addressable nor continuously available. It also leverages the benefits of SOAP while using a well-defined exchange pattern and profile to enable interoperability. The ECP may be viewed as a SOAP intermediary between the service provider and the identity provider.

An *enhanced client* may be a browser or some other user agent that supports the functionality described in this profile. An *enhanced proxy* is an HTTP proxy that emulates an enhanced client. Unless stated otherwise, all statements referring to enhanced clients are to be understood as statements about both enhanced clients as well as enhanced client proxies.

Since the enhanced client sends and receives messages in the body of HTTP requests and responses, it has no arbitrary restrictions on the size of the protocol messages.

This profile leverages the Reverse SOAP (PAOS) binding (see clause 10). Implementers of this profile must follow the rules for HTTP indications of PAOS support specified in that binding, in addition to those specified in this profile. This profile utilizes a PAOS SOAP header block conveyed between the HTTP responder and the ECP but does not define PAOS itself. This profile defines SOAP header blocks that accompany the SAML requests and responses. These header blocks may be composed with other SOAP header blocks as necessary, for example with the SOAP Message Security header block to add security features if needed, for example a digital signature applied to the authentication request.

Two sets of request/response SOAP header blocks are used: PAOS header blocks for generic PAOS information and ECP profile-specific header blocks to convey information specific to ECP profile functionality.

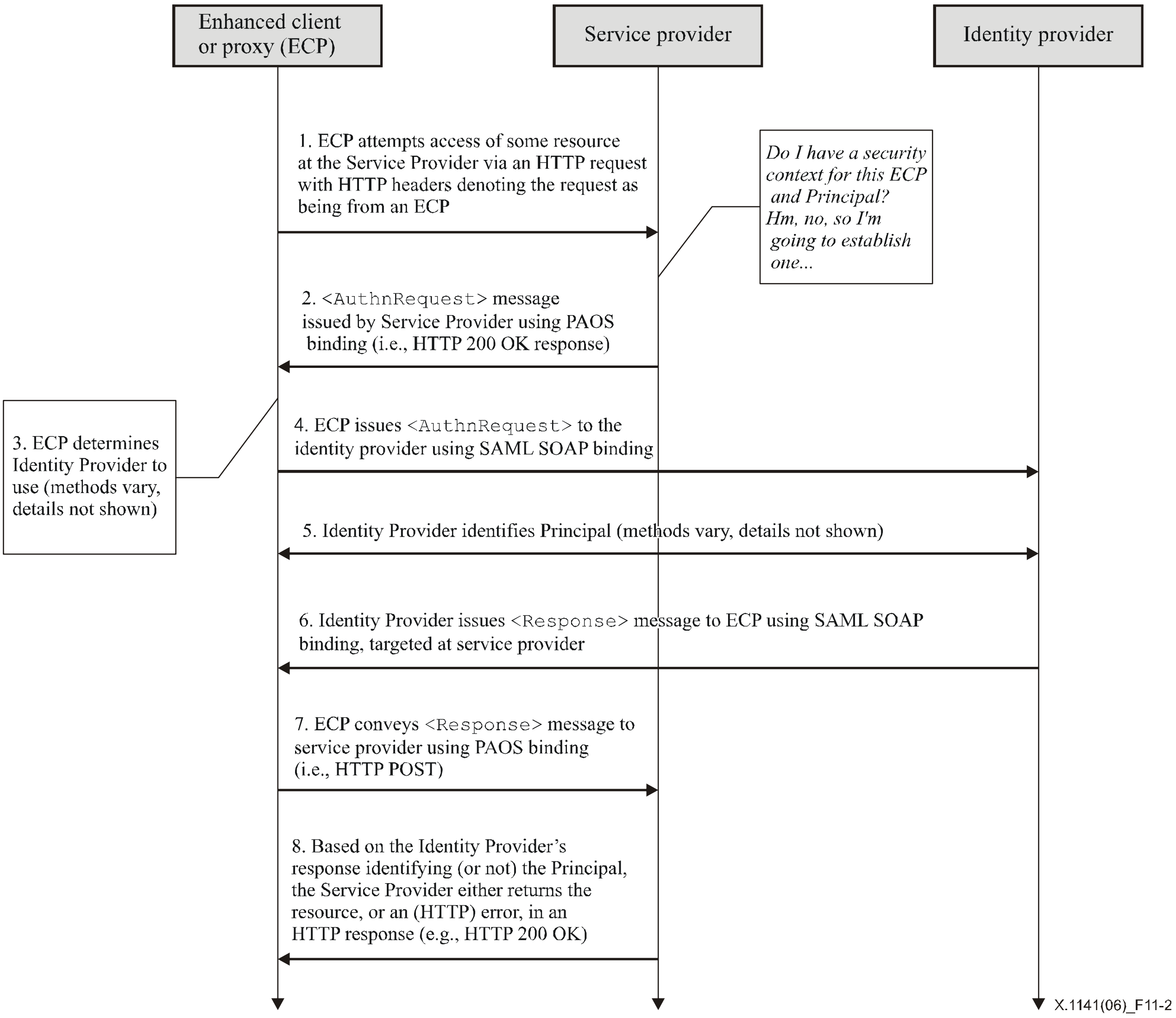
Enhanced client or proxy (ECP)
Service provider
Identity provider
1. ECP attempts access of some resource at the Service Provider via an HTTP request with HTTP headers denoting the request as being from an ECP
Do I have a security context for this ECP and Principal? Hm, no, so I'm going to establish one...
2. <AuthnRequest> message issued by Service Provider using PAOS binding (i.e., HTTP 200 OK response)
3. ECP determines Identity Provider to use (methods vary, details not shown)
4. ECP issues <AuthnRequest> to the identity provider using SAML SOAP binding
5. Identity Provider identifies Principal (methods vary, details not shown)
6. Identity Provider issues <Response> message to ECP using SAML SOAP binding, targeted at service provider
7. ECP conveys <Response> message to service provider using PAOS binding (i.e., HTTP POST)
8. Based on the Identity Provider's response identifying (or not) the Principal, the Service Provider either returns the resource, or an (HTTP) error, in an HTTP response (e.g., HTTP 200 OK)
X.1141(06)_F11-2

Figure 11-2/X.1141 – Processing flow in the ECP profile

Figure 11-2 illustrates the basic template for SSO using an ECP. The following steps are described by the profile. Within an individual step, there may be one or more actual message exchanges depending on the binding used for that step and other implementation-dependent behaviour.

1) **ECP issues HTTP request to service provider**

In step 1, the Principal, via an ECP, makes an HTTP request for a secured resource at a service provider, where the service provider does not have an established security context for the ECP and Principal.

2) **Service provider issues `<AuthnRequest>` to ECP**

In step 2, the service provider issues an `<AuthnRequest>` message to the ECP, which is to be delivered by the ECP to the appropriate identity provider. The Reverse SOAP (PAOS) binding (see clause 10) is used here.

3) **ECP determines identity provider**

In step 3, the ECP obtains the location of an endpoint at an identity provider for the authentication request protocol that supports its preferred binding. The means by which this is accomplished is implementation-dependent. The ECP may use the SAML identity provider discovery profile described in 11.4.3.

NOTE (informative) – PE18 (see OASIS PE:2006) suggests to delete the last line from the above paragraph.

4) **ECP conveys `<AuthnRequest>` to identity provider**

In step 4, the ECP conveys the `<AuthnRequest>` to the identity provider identified in step 3 using a modified form of the SAML SOAP binding (see clause 10) with the additional allowance that the identity provider may exchange arbitrary HTTP messages with the ECP before responding to the SAML request.

5) **Identity provider identifies principal**

In step 5, the Principal is identified by the identity provider by some means outside the scope of this profile. This may require a new act of authentication, or it may reuse an existing authenticated session.

6) **Identity provider issues `<Response>` to ECP, targeted at service provider**

In step 6, the identity provider issues a `<Response>` message, using the SAML SOAP binding, to be delivered by the ECP to the service provider. The message may indicate an error, or will include (at least) an authentication assertion.

7) **ECP conveys `<Response>` message to service provider**

In step 7, the ECP conveys the `<Response>` message to the service provider using the PAOS binding.

8) **Service provider grants or denies access to principal**

In step 8, having received the `<Response>` message from the identity provider, the service provider either establishes its own security context for the principal and returns the requested resource, or responds to the principal's ECP with an error.

11.4.2.3 Profile description

The following subclauses provide detailed definitions of the individual steps.

11.4.2.3.1 ECP issues HTTP request to service provider

The ECP sends an HTTP request to a service provider, specifying some resource. This HTTP request must conform to the PAOS binding, which means it must include the following HTTP header fields:

1) The HTTP `Accept` Header field indicating the ability to accept the MIME type "`application/vnd.paos+xml`".
2) The HTTP `PAOS` Header field specifying the `PAOS` version with `urn:liberty:paos:2003-08` at minimum.
3) Furthermore, support for this profile must be specified in the HTTP `PAOS` Header field as a service value, with the value `urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp`. This value should correspond to the service attribute in the PAOS Request SOAP header block.

For example, a user agent may request a page from a service provider as follows:

```
GET /index HTTP/1.1
Host: identity-service.example.com
Accept: text/html; application/vnd.paos+xml
PAOS: ver='urn:liberty:paos:2003-08' ;
'urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp'
```

11.4.2.3.2 Service provider issues <AuthnRequest> to ECP

When the service provider requires a security context for the principal before allowing access to the specified resource, that is, before providing a service or data, it can respond to the HTTP request using the PAOS binding with an `<AuthnRequest>` message in the HTTP response. The service provider will issue an HTTP 200 OK response to the ECP containing a single SOAP envelope.

The SOAP envelope must contain:

1) An `<AuthnRequest>` element in the SOAP body, intended for the ultimate SOAP recipient, the identity provider.
2) A PAOS SOAP header block targeted at the ECP using the SOAP actor value of `http://schemas.xmlsoap.org/soap/actor/next`. This header block provides control information such as the URL to which to send the response in this solicit-response message exchange pattern.
3) An ECP profile-specific Request SOAP header block targeted at the ECP using the SOAP actor `http://schemas.xmlsoap.org/soap/actor/next`. The ECP Request header block defines information related to the authentication request that the ECP may need to process it, such as a list of identity providers acceptable to the service provider, whether the ECP may interact with the principal through the client, and the service provider's human-readable name that may be displayed to the principal.

The SOAP envelope may contain an ECP RelayState SOAP header block targeted at the ECP using the SOAP actor value of `http://schemas.xmlsoap.org/soap/actor/next`. The header contains state information to be returned by the ECP along with the SAML response.

11.4.2.3.3 ECP determines identity provider

The ECP will determine which identity provider is appropriate and route the SOAP message appropriately.

11.4.2.3.4 ECP issues <AuthnRequest> to identity provider

The ECP must remove the PAOS, ECP RelayState, and ECP Request header blocks before passing the `<AuthnRequest>` message on to the identity provider, using a modified form of the SAML SOAP binding. The SAML request is submitted via SOAP in the usual fashion, but the identity provider may respond to the ECP's HTTP request with an HTTP response containing, for example, an HTML login form or some other presentation-oriented response. A sequence of HTTP exchanges may take place, but ultimately the identity provider must complete the SAML SOAP exchange and return a SAML response via the SOAP binding.

The `<AuthnRequest>` element may itself be signed by the service provider. In this and other respects, the message rules specified in the browser SSO profile in 11.4.1.4.1 must be followed.

Prior to or subsequent to this step, the identity provider must establish the identity of the principal by some means, or it must return an error `<Response>`, as described in 11.4.2.3.6 below.

11.4.2.3.5 Identity provider identifies principal

At any time during the previous step or subsequent to it, the identity provider must establish the identity of the principal (unless it returns an error to the service provider). The `ForceAuthn` `<AuthnRequest>` attribute, if present with a value of true, obligates the identity provider to freshly establish this identity, rather than relying on an existing session it may have with the principal. Otherwise, and in all other respects, the identity provider may use any means to authenticate the user agent, subject to any requirements included in the `<AuthnRequest>` in the form of the `<RequestedAuthnContext>` element.

11.4.2.3.6 Identity provider issues <Response> to ECP, targeted at service provider

The identity provider returns a SAML `<Response>` message (or SOAP fault) when presented with an authentication request, after having established the identity of the principal. The SAML response is conveyed using the SAML SOAP binding in a SOAP message with a `<Response>` element in the SOAP body, intended for the service provider as the ultimate SOAP receiver. The rules for the response specified in the browser SSO profile in 11.4.1.4.2 must be followed.

The identity provider's response message must contain a profile-specific ECP Response SOAP header block, and may contain an ECP RelayState header block, both targeted at the ECP.

11.4.2.3.7 ECP conveys <Response> message to service provider

The ECP removes the header block(s), and may add a PAOS Response SOAP header block and an ECP RelayState header block before forwarding the SOAP response to the service provider using the PAOS binding.

The `<paos:Response>` SOAP header block in the response to the service provider is generally used to correlate this response to an earlier request from the service provider. In this profile, the correlation `refToMessageID` attribute is not required since the SAML `<Response>` element's `InResponseTo` attribute may be used for this purpose, but if the `<paos:Request>` SOAP Header block had a `messageID` then the `<paos:Response>` SOAP header block must be used.

The `<ecp:RelayState>` header block value is typically provided by the service provider to the ECP with its request, but if the identity provider is producing an unsolicited response (without having received a corresponding SAML request), then it may include a RelayState header block that indicates, based on mutual agreement with the service provider, how to handle subsequent interactions with the ECP. This may be the URL of a resource at the service provider.

If the service provider included an `<ecp:RelayState>` SOAP header block in its request to the ECP, or if the identity provider included an `<ecp:RelayState>` SOAP header block with its response, then the ECP must include an identical header block with the SAML response sent to the service provider. The service provider's value for this header block (if any) must take precedence.

11.4.2.3.8 Service provider grants or denies access to principal

Once the service provider has received the SAML response in an HTTP request (in a SOAP envelope using PAOS), it may respond with the service data in the HTTP response. In consuming the response, the rules specified in the browser SSO profile in 11.4.1.4.3 and 11.4.1.4.5 must be followed. That is, the same processing rules used when receiving the `<Response>` with the HTTP POST binding apply to the use of PAOS.

11.4.2.4 ECP Profile schema usage

The ECP Profile XML schema defines the SOAP Request/Response header blocks used by this profile. Following is a complete listing of this schema document.

```
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
        schemaLocation="saml-schema-protocol-2.0.xsd"/>
    <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="saml-schema-assertion-2.0.xsd"/>
    <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
        schemaLocation="http://schemas.xmlsoap.org/soap/envelope/"/>
    <annotation>
        <documentation>
            Document identifier: saml-schema-ecp-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
              V2.0 (March, 2005):
                Custom schema for ECP profile, first published in SAML 2.0.
        </documentation>
    </annotation>

    <element name="Request" type="ecp:RequestType"/>
    <complexType name="RequestType">
        <sequence>
            <element ref="saml:Issuer"/>
            <element ref="samlp:IDPList" minOccurs="0"/>
        </sequence>
        <attribute ref="S:mustUnderstand" use="required"/>
        <attribute ref="S:actor" use="required"/>
        <attribute name="ProviderName" type="string" use="optional"/>
        <attribute name="IsPassive" type="boolean" use="optional"/>
    </complexType>

    <element name="Response" type="ecp:ResponseType"/>
    <complexType name="ResponseType">
        <attribute ref="S:mustUnderstand" use="required"/>
        <attribute ref="S:actor" use="required"/>
        <attribute name="AssertionConsumerServiceURL" type="anyURI"
use="required"/>
    </complexType>
    <element name="RelayState" type="ecp:RelayStateType"/>
    <complexType name="RelayStateType">
        <simpleContent>
            <extension base="string">
                <attribute ref="S:mustUnderstand" use="required"/>
                <attribute ref="S:actor" use="required"/>
            </extension>
        </simpleContent>
    </complexType>
</schema>
```

The following subclauses describe how these XML constructs are to be used.

11.4.2.4.1 PAOS Request header block: SP to ECP

The PAOS Request header block signals the use of PAOS processing and includes the following attributes:

– responseConsumerURL [Required]

Specifies where the ECP is to send an error response. Also used to verify the correctness of the identity provider's response, by cross-checking this location against the AssertionServiceConsumerURL in the ECP response header block. This value must be the same as the AssertionServiceConsumerURL (or the URL referenced in metadata) conveyed in the <AuthnRequest>.

NOTE (informative) – PE22 (see OASIS PE:2006) suggests to change AssertionServiceConsumerURL in the last sentence to AssertionConsumerServiceURL.

– service [Required]

Indicates that the PAOS service being used is this SAML authentication profile. The value must be urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp.

– SOAP-ENV:mustUnderstand [Required]

The value must be 1 (true). A SOAP fault must be generated if the PAOS header block is not understood.

– SOAP-ENV:actor [Required]

The value must be http://schemas.xmlsoap.org/soap/actor/next.

– messageID [Optional]

Allows optional response correlation. It may be used in this profile, but is not required, since this functionality is provided by the SAML protocol layer, via the ID attribute in the <AuthnRequest> and the InResponseTo attribute in the <Response>.

The PAOS Request SOAP header block has no element content.

11.4.2.4.2 ECP Request header block: SP to ECP

The ECP Request SOAP header block is used to convey information needed by the ECP to process the authentication request. It is mandatory and its presence signals the use of this profile. It contains the following elements and attributes:

– SOAP-ENV:mustUnderstand [Required]

The value must be 1 (true). A SOAP fault must be generated if the ECP header block is not understood.

– SOAP-ENV:actor [Required]

The value must be http://schemas.xmlsoap.org/soap/actor/next.

– ProviderName [Optional]

A human-readable name for the requesting service provider.

– IsPassive [Optional]

A boolean value. If true, the identity provider and the client itself must not take control of the user interface from the request issuer and interact with the principal in a noticeable fashion. If a value is not provided, the default is true.

– `<saml:Issuer>` [Required]

This element must contain the unique identifier of the requesting service provider; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

– `<samlp:IDPList>` [Optional]

Optional list of identity providers that the service provider recognizes and from which the ECP may choose to service the request.

11.4.2.4.3 ECP RelayState header block: SP to ECP

The ECP RelayState SOAP header block is used to convey state information from the service provider that it will need later when processing the response from the ECP. It is optional, but if used, the ECP must include an identical header block in the response in step 5 in Figure 11-2. It contains the following attributes:

NOTE (informative) – PE27 (see OASIS PE:2006) suggests to replace step 5 with step 7 in the text above.

– `SOAP-ENV:mustUnderstand` [Required]

The value must be `1` (true). A SOAP fault must be generated if the header block is not understood.

– `SOAP-ENV:actor` [Required]

The value must be `http://schemas.xmlsoap.org/soap/actor/next`.

The content of the header block element is a string containing state information created by the requester. If provided, the ECP must include the same value in a RelayState header block when responding to the service provider in step 5. The string value must not exceed 80 bytes in length and should be integrity protected by the requester independent of any other protections that may or may not exist during message transmission.

The following is an example of the SOAP authentication request from the service provider to the ECP:

```
<SOAP-ENV:Envelope
       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
       xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
       xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
       responseConsumerURL="http://identity-service.example.com/abc"
       messageID="6c3a4f8b9c2d" SOAP-
ENV:actor="http://schemas.xmlsoap.org/soap/actor/next" SOAP-
ENV:mustUnderstand="1"
       service="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp">
    </paos:Request>
    <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
       SOAP-ENV:mustUnderstand="1" SOAP-
ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
       ProviderName="Service Provider X" IsPassive="0">
      <saml:Issuer>https://ServiceProvider.example.com</saml:Issuer>
      <samlp:IDPList>
        <samlp:IDPEntry ProviderID="https://IdentityProvider.example.com"
            Name="Identity Provider X"
            Loc="https://IdentityProvider.example.com/saml2/sso"
        </samlp:IDPEntry>
        <samlp:GetComplete>
        https://ServiceProvider.example.com/idplist?id=604be136-fe91-441e-
afb8
        </samlp:GetComplete>
      </samlp:IDPList>
    </ecp:Request>
     <ecp:RelayState
xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
       SOAP-ENV:mustUnderstand="1" SOAP-
ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
      ...
    </ecp:RelayState>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

As noted above, the PAOS and ECP header blocks are removed from the SOAP message by the ECP before the authentication request is forwarded to the identity provider. An example authentication request from the ECP to the identity provider is as follows:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  <SOAP-ENV:Body>
    <samlp:AuthnRequest> ... </samlp:AuthnRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

11.4.2.4.4 ECP Response header block: IdP to ECP

The ECP Response SOAP header block must be used on the response from the identity provider to the ECP. It contains the following attributes:

- `SOAP-ENV:mustUnderstand` [Required]

  The value must be `1` (true). A SOAP fault must be generated if the ECP header block is not understood.

- `SOAP-ENV:actor` [Required]

  The value must be `http://schemas.xmlsoap.org/soap/actor/next`.

- `AssertionConsumerServiceURL` [Required]

  Set by the identity provider based on the `<AuthnRequest>` message or the service provider's metadata obtained by the identity provider.

The ECP must confirm that this value corresponds to the value the ECP obtained in the `responseConsumerURL` in the PAOS Request SOAP header block it received from the service provider. Since the `responseConsumerURL` may be relative and the `AssertionConsumerServiceURL` is absolute, some processing/normalization may be required.

This mechanism is used for security purposes to confirm the correct response destination. If the values do not match, then the ECP must generate a SOAP fault response to the service provider and must not return the SAML response.

The ECP Response SOAP header has no element content.

Following is an example of an IdP-to-ECP response.

```
<SOAP-ENV:Envelope
      xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
     <ecp:Response SOAP-ENV:mustUnderstand="1" SOAP-
ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assert
ion_consumer"/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <samlp:Response> ... </samlp:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

11.4.2.4.5 PAOS Response header block: ECP to SP

The PAOS Response header block includes the following attributes:

- `SOAP-ENV:mustUnderstand` [Required]

  The value must be `1` (true). A SOAP fault must be generated if the PAOS header block is not understood.

- `SOAP-ENV:actor` [Required]

  The value must be `http://schemas.xmlsoap.org/soap/actor/next`.

– `refToMessageID` [Optional]

Allows correlation with the PAOS request. This optional attribute (and the header block as a whole) must be added by the ECP if the corresponding PAOS request specified the `messageID` attribute. The equivalent functionality is provided in SAML using `<AuthnRequest>` and `<Response>` correlation.

The PAOS Response SOAP header has no element content.

Following is an example of an ECP-to-SP response.

```
<SOAP-ENV:Envelope
       xmlns:paos="urn:liberty:paos:2003-08"
       xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
       xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <paos:Response refToMessageID="6c3a4f8b9c2d" SOAP-
ENV:actor="http://schemas.xmlsoap.org/soap/actor/next/" SOAP-
ENV:mustUnderstand="1"/>
    <ecp:RelayState
xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
       SOAP-ENV:mustUnderstand="1" SOAP-
ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
       ...
    </ecp:RelayState>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <samlp:Response> ... </samlp:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

11.4.2.5 Security considerations

The `<AuthnRequest>` message should be signed. Per the rules specified by the browser SSO profile, the assertions enclosed in the `<Response>` must be signed. The delivery of the response in the SOAP envelope via PAOS is essentially analogous to the use of the HTTP POST binding and security countermeasures appropriate to that binding are used.

The SOAP headers should be integrity protected, such as with SOAP Message Security or through the use of TLS over every HTTP exchange with the client.

The service provider should be authenticated to the ECP, for example with server-side TLS authentication.

The ECP should be authenticated to the identity provider, such as by maintaining an authenticated session. Any HTTP exchanges subsequent to the delivery of the `<AuthnRequest>` message and before the identity provider returns a `<Response>` must be securely associated with the original request.

NOTE (informative) – PE20 (see OASIS PE:2006) suggests the addition of a subclause to discuss ECP Metadata considerations as given below:

The rules specified in the browser SSO profile in clause 11 apply here as well. Specifically, the indexed endpoint element `<md:AssertionConsumerService>` with a binding of `urn:oasis:names:tc:SAML:2.0:bindings:PAOS` may be used to describe the supported binding and location(s) to which an identity provider may send responses to a service provider using this profile. Moreover, the endpoint `<md:SingleSignOnService>` with a binding of `urn:oasis:names:tc:SAML:2.0:bindings:SOAP` may be used to describe the supported binding and location(s) to which a service provider may send requests to a identity provider using this profile.

11.4.3 Identity provider discovery profile

This clause defines a profile by which a service provider can discover which identity providers a principal is using with the Web Browser SSO profile. In deployments having more than one identity provider, service providers need a means to discover which identity provider(s) a principal uses. The discovery profile relies on a cookie that is written in a domain that is common between identity providers and service providers in a deployment. The domain that the deployment predetermines is known as the common domain in this profile, and the cookie containing the list of identity providers is known as the common domain cookie.

Which entities host web servers in the common domain is a deployment issue and is outside the scope of this profile.

NOTE (informative) – PE32 (see OASIS PE:2006) suggests to add the following to describe the required information:

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:SSO:idp-discovery`

Contact information: security-services-comment@lists.oasis-open.org

11.4.3.1 Common domain cookie

The name of the cookie must be "_saml_idp". The format of the cookie value must be a set of one or more base64-encoded URI values separated by a single space character. Each URI is the unique identifier of an identity provider, as defined in clause 7. The final set of values is then URL encoded.

The common domain cookie writing service should append the identity provider's unique identifier to the list. If the identifier is already present in the list, it may remove and append it. The intent is that the most recently established identity provider session is the last one in the list.

The cookie must be set with a Path prefix of "/". The Domain must be set to ".[common-domain]" where [common-domain] is the common domain established within the deployment for use with this profile. There must be a leading period. The cookie must be marked as secure.

Cookie syntax should be in accordance with IETF RFC 2965. The cookie may be either session-only or persistent. This choice may be made within a deployment, but should apply uniformly to all identity providers in the deployment.

11.4.3.2 Setting the common domain cookie

After the identity provider authenticates a principal, it may set the common domain cookie. The means by which the identity provider sets the cookie is implementation-specific so long as the cookie is successfully set with the parameters given above. One possible implementation strategy follows and should be considered non-normative. The identity provider may:

- Have previously established a DNS and IP alias for itself in the common domain.
- Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL scheme. The structure of the URL is private to the implementation and may include session information needed to identify the user agent.
- Set the cookie on the redirected user agent using the parameters specified above.
- Redirect the user agent back to itself, or, if appropriate, to the service provider.

11.4.3.3 Obtaining the common domain cookie

When a service provider needs to discover which identity providers a principal uses, it invokes an exchange designed to present the common domain cookie to the service provider after it is read by an HTTP server in the common domain.

If the HTTP server in the common domain is operated by the service provider or if other arrangements are in place, the service provider may utilize the HTTP server in the common domain to relay its <AuthnRequest> to the identity provider for an optimized single sign-on process.

The specific means by which the service provider reads the cookie is implementation-specific so long as it is able to cause the user agent to present cookies that have been set with the parameters given in 11.4.3.1. One possible implementation strategy is described as follows and should be considered non-normative. Additionally, it may be sub-optimal for some applications.

- Have previously established a DNS and IP alias for itself in the common domain.
- Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL scheme. The structure of the URL is private to the implementation and may include session information needed to identify the user agent.
- Redirect the user agent back to itself, or, if appropriate, to the identity provider.

11.4.4 Single logout profile

Once a principal has authenticated to an identity provider, the authenticating entity may establish a session with the principal (typically by means of a cookie, URL re-writing, or some other implementation-specific means). The identity provider may subsequently issue assertions to service providers or other relying parties, based on this authentication event; a relying party may use this to establish *its own* session with the principal.

In such a situation, the identity provider can act as a session authority and the relying parties as session participants. At some later time, the principal may wish to terminate his or her session either with an individual session participant, or with all session participants in a given session managed by the session authority. The former case is considered out of scope of this Recommendation. The latter case, however, may be satisfied using this profile of the SAML Single Logout protocol (see 11.4).

A principal (or an administrator terminating a principal's session) may choose to terminate this "global" session either by contacting the session authority, or an individual session participant. Also an identity provider acting as a session authority may *itself* act as a session participant in situations in which it is the relying party for another identity provider's assertions regarding that principal.

The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A front-channel binding may be required, for example, in cases in which a principal's session state exists solely in a user agent in the form of a cookie and a direct interaction between the user agent and the session participant or session authority is required. As will be discussed below, session participants should, if possible, use a "front-channel" binding when initiating this profile to maximize the likelihood that the session authority can propagate the logout successfully to all participants.

11.4.4.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None

11.4.4.2 Profile overview

Figure 11-3 illustrates the basic template for achieving single logout:

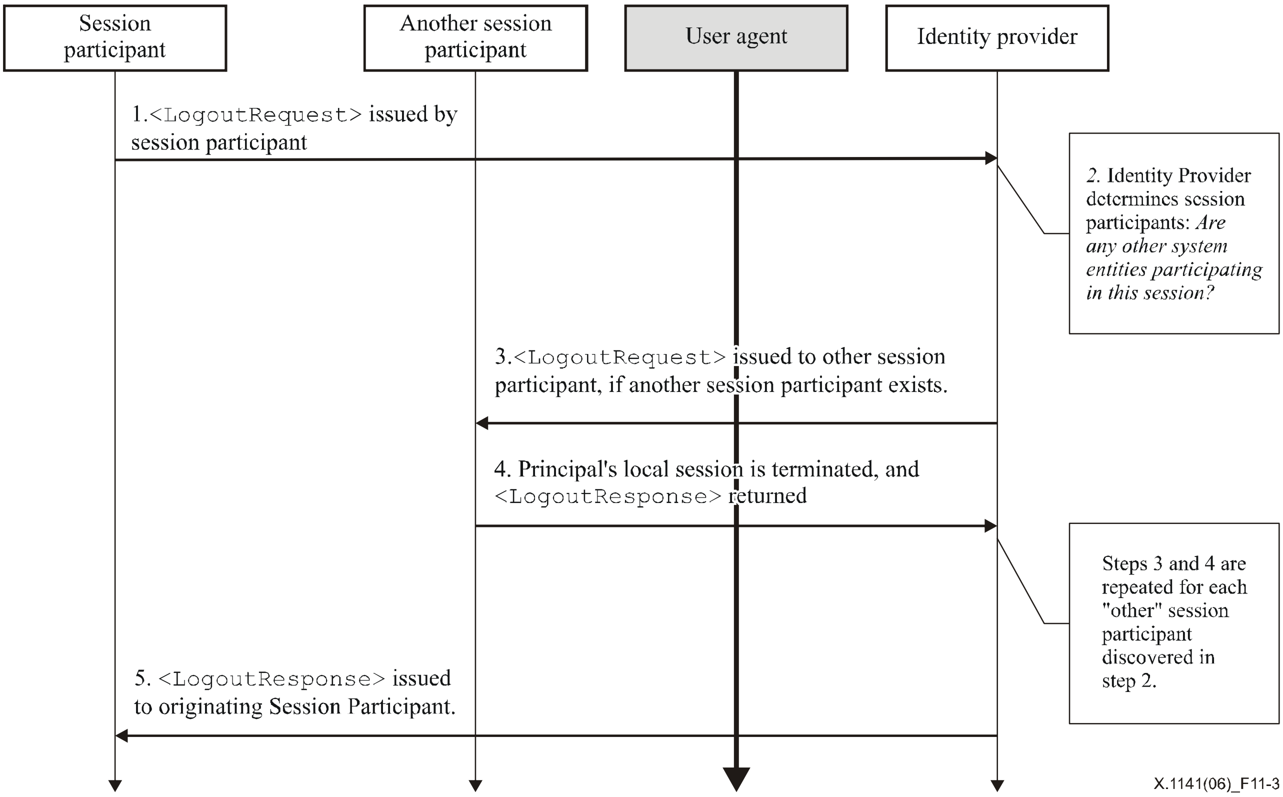
Session participant
Another session participant
User agent
Identity provider
1.<LogoutRequest> issued by session participant
2. Identity Provider determines session participants: *Are any other system entities participating in this session?*
3.<LogoutRequest> issued to other session participant, if another session participant exists.
4. Principal's local session is terminated, and <LogoutResponse> returned
Steps 3 and 4 are repeated for each "other" session participant discovered in step 2.
5. <LogoutResponse> issued to originating Session Participant.
X.1141(06)_F11-3

Figure 11-3/X.1141 – Template for achieving single logout

The grayed-out user agent illustrates that the message exchange may pass through the user agent or may be a direct exchange between system entities, depending on the SAML binding used to implement the profile.

The following steps are described by the profile. Within an individual step, there may be one or more actual message exchanges depending on the binding used for that step and other implementation-dependent behaviour.

1) **<LogoutRequest> issued by session participant to identity provider**

   In step 1, the session participant initiates single logout and terminates a principal's session(s) by sending a `<LogoutRequest>` message to the identity provider from whom it received the corresponding authentication assertion. The request may be sent directly to the identity provider or sent indirectly through the user agent.

2) **Identity provider determines session participants**

In step 2, the identity provider uses the contents of the <LogoutRequest> message (or if initiating logout itself, some other mechanism) to determine the session(s) being terminated. If there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4 are repeated for each session participant identified.

3) **<LogoutRequest> issued by identity provider to session participant/authority**

In step 3, the identity provider issues a <LogoutRequest> message to a session participant or session authority related to one or more of the session(s) being terminated. The request may be sent directly to the entity or sent indirectly through the user agent (if consistent with the form of the request in step 1).

4) **Session participant/authority issues <LogoutResponse> to identity provider**

In step 4, a session participant or session authority terminates the principal's session(s) as directed by the request (if possible) and returns a <LogoutResponse> to the identity provider. The response may be returned directly to the identity provider or indirectly through the user agent (if consistent with the form of the request in step 3).

5) **Identity provider issues <LogoutResponse> to session participant**

In step 5, the identity provider issues a <LogoutResponse> message to the original requesting session participant. The response may be returned directly to the session participant or indirectly through the user agent (if consistent with the form of the request in step 1).

An identity provider (acting as session authority) can initiate this profile at step 2 and issue a <LogoutRequest> to all session participants, also skipping step 5.

11.4.4.3 Profile description

If the profile is initiated by a session participant, start with 11.4.4.3.1. If initiated by the identity provider, start with 11.4.4.3.2. In the descriptions below, the following is referred to:

– **Single logout service**

This is the single logout protocol endpoint at an identity provider or session participant to which the <LogoutRequest> or <LogoutResponse> messages (or an artifact representing them) are delivered. The same or different endpoints may be used for requests and responses.

11.4.4.3.1 <LogoutRequest> issued by session participant to identity provider

If the logout profile is initiated by a session participant, it examines the authentication assertion(s) it received pertaining to the session(s) being terminated, and collects the SessionIndex value(s) it received from the identity provider. If multiple identity providers are involved, then the profile must be repeated independently for each one.

To initiate the profile, the session participant issues a <LogoutRequest> message to the identity provider's single logout service request endpoint containing one or more applicable <SessionIndex> elements. At least one element must be included. Metadata may be used to determine the location of this endpoint and the bindings supported by the identity provider.

Asynchronous bindings (Front-channel)

The session participant should (if the principal's user agent is present) use an asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings (see clause 10), to send the request to the identity provider through the user agent. The identity provider should then propagate any required logout messages to additional session participants as required using either a synchronous or asynchronous binding. The use of an asynchronous binding for the original request is preferred because it gives the identity provider the best chance of successfully propagating the logout to the other session participants during step 3 in 11.4.4.2.

If the HTTP Redirect or POST binding is used, then the <LogoutRequest> message is delivered to the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile defined in 11.4.6 is used by the identity provider, which makes a callback to the session participant to retrieve the <LogoutRequest> message, using for example the SOAP binding.

It is recommended that the HTTP exchanges in this step be made over TLS 1.0 to maintain confidentiality and message integrity. The <LogoutRequest> message must be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.

Each of these bindings provide a RelayState mechanism that the session participant may use to associate the profile exchange with the original request. The session participant should reveal as little information as possible in the `RelayState` value unless the use of the profile does not require such privacy measures.

Synchronous bindings (Back-channel)

Alternatively, the session participant may use a synchronous binding, such as the SOAP binding (see clause 10), to send the request directly to the identity provider. The identity provider should then propagate any required logout messages to additional session participants as required using a synchronous binding. The requester must authenticate itself to the identity provider, either by signing the `<LogoutRequest>` or using any other binding-supported mechanism.

Profile-specific rules for the contents of the `<LogoutRequest>` message are included in 11.4.4.4.1.

11.4.4.3.2 Identity provider determines session participants

If the logout profile is initiated by an identity provider, or upon receiving a valid `<LogoutRequest>` message, the identity provider processes the request, it must examine the identifier and `<SessionIndex>` elements and determine the set of sessions to be terminated.

The identity provider then follows steps 3 and 4 in Figure 11-3 for each entity participating in the session(s) being terminated, other than the original requesting session participant (if any), as described in 8.2.7.

11.4.4.3.3 <LogoutRequest> issued by identity provider to session participant/authority

To propagate the logout, the identity provider issues its own `<LogoutRequest>` to a session authority or participant in a session being terminated. The request is sent using a SAML binding consistent with the capability of the responder and the availability of the user agent at the identity provider.

In general, the binding with which the original request was received in step 1 in Figure 11-3 does not dictate the binding that may be used in this step except that as noted in step 1, using a synchronous binding that bypasses the user agent constrains the identity provider to use a similar binding to propagate additional requests.

Profile-specific rules for the contents of the `<LogoutRequest>` message are included in 11.4.4.4.1.

11.4.4.3.4 Session participant/authority issues <LogoutResponse> to identity provider

The session participant/authority must process the `<LogoutRequest>` message as defined in 8.2.7. After processing the message or upon encountering an error, the entity must issue a `<LogoutResponse>` message containing an appropriate status code to the requesting identity provider to complete the SAML protocol exchange.

Synchronous bindings (Back-channel)

If the identity provider used a synchronous binding, such as the SOAP binding (see clause 10), the response is returned directly to complete the synchronous communication. The responder must authenticate itself to the requesting identity provider, either by signing the `<LogoutResponse>` or using any other binding-supported mechanism.

Asynchronous bindings (Front-channel)

If the identity provider used an asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings (see clause 10), then the `<LogoutResponse>` (or artifact) is returned through the user agent to the identity provider's single logout service response endpoint. Metadata may be used to determine the location of this endpoint and the bindings supported by the identity provider. Any asynchronous binding supported by both entities may be used.

If the HTTP Redirect or POST binding is used, then the `<LogoutResponse>` message is delivered to the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile defined in 11.4.6 is used by the identity provider, which makes a callback to the responding entity to retrieve the `<LogoutResponse>` message, using for example the SOAP binding.

It is recommended that the HTTP exchanges in this step be made over TLS 1.0 to maintain confidentiality and message integrity. The `<LogoutResponse>` message must be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the response issuer when the artifact is dereferenced.

Profile-specific rules for the contents of the `<LogoutResponse>` message are included in 11.4.4.4.2.

11.4.4.3.5 Identity provider issues <LogoutResponse> to session participant

After processing the original session participant's `<LogoutRequest>` as described in the previous steps, the identity provider must respond to the original request with a `<LogoutResponse>` containing an appropriate status code to complete the SAML protocol exchange.

The response is sent to the original session participant, using a SAML binding consistent with the binding used in the original request, the capability of the responder, and the availability of the user agent at the identity provider. Assuming an asynchronous binding was used in step 1 in Figure 11-3, then any binding supported by both entities may be used.

Profile-specific rules for the contents of the `<LogoutResponse>` message are included in 11.4.4.4.2.

11.4.4.4 Use of single logout protocol

This clause describes `<LogoutRequest>` and `<LogoutResponse>` usage.

11.4.4.4.1 <LogoutRequest> usage

The `<Issuer>` element must be present and must contain the unique identifier of the requesting entity; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

The requester must authenticate itself to the responder and ensure message integrity, either by signing the message or using a binding-specific mechanism.

The principal must be identified in the request using an identifier that **strongly matches** the identifier in the authentication assertion the requester issued or received regarding the session being terminated, per the matching rules defined in 8.2.7.

If the requester is a session participant, it must include at least one `<SessionIndex>` element in the request. If the requester is a session authority (or acting on its behalf), then it may omit any such elements to indicate the termination of all of the principal's applicable sessions.

> NOTE (informative) – PE38 (see OASIS PE:2006) clarifies the above paragraph as below:
>
> If the requester is a session participant, it must include at least one `<SessionIndex>` element in the request. (From 11.4 the session participant always receives a `SessionIndex` attribute in the `<saml:AuthnStatement>` elements that it receives to initiate the session). If the requester is a session authority (or acting on its behalf), then it may omit any such elements to indicate the termination of all of the principal's applicable sessions.

11.4.4.4.2 <LogoutResponse> usage

The `<Issuer>` element must be present and must contain the unique identifier of the responding entity; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

The responder must authenticate itself to the requester and ensure message integrity, either by signing the message or using a binding-specific mechanism.

11.4.4.5 Use of metadata

The endpoint element, `<md:SingleLogoutService>` describes supported bindings and location(s) to which an entity may send requests and responses using this profile. A requester, if encrypting the principal's identifier, can use the responder's `<md:KeyDescriptor>` element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

11.4.5 Name identifier management profile

In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged some form of persistent identifier for a principal with a service provider, allowing them to share a common identifier for some length of time. Subsequently, the identity provider may wish to notify the service provider of a change in the format and/or value that it will use to identify the same principal in the future. Alternatively, the service provider may wish to attach its own "alias" for the principal in order to ensure that the identity provider will include it when communicating with it in the future about the principal. Finally, one of the providers may wish to inform the other that it will no longer issue or accept messages using a particular identifier. To implement these scenarios, a profile of the SAML Name Identifier Management protocol is used.

> NOTE (informative) – PE12 (see OASIS PE:2006) suggests to rewrite the second sentence in the above paragraph as following:
>
> Subsequently, the identity provider may wish to notify the service provider of a change in the value that it will use to identify the same principal in the future.

The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A front-channel binding may be required, for example, in cases in which direct interaction between the user agent and the responding provider is required in order to effect the change.

11.4.5.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:SSO:nameid-mgmt`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

11.4.5.2 Profile overview

Figure 11-4 illustrates the basic template for the name identifier management profile.

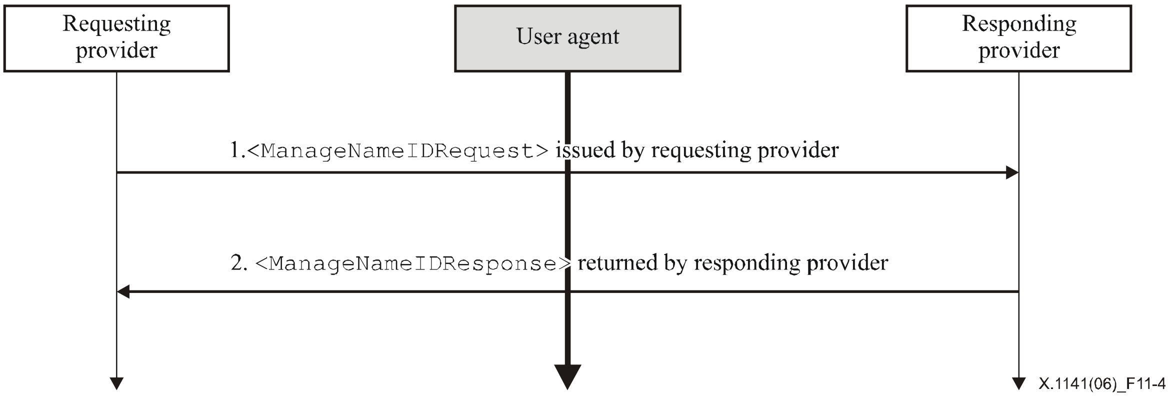
Requesting provider
User agent
Responding provider
1.<ManageNameIDRequest> issued by requesting provider
2. <ManageNameIDResponse> returned by responding provider
X.1141(06)_F11-4

Figure 11-4/X.1141 – Name identifier management profile

The grayed-out user agent illustrates that the message exchange may pass through the user agent or may be a direct exchange between system entities, depending on the SAML binding used to implement the profile.

The following steps are described by the profile. Within an individual step, there may be one or more actual message exchanges depending on the binding used for that step and other implementation-dependent behaviour.

1) **<ManageNameIDRequest> issued by requesting identity/service provider**

   In step 1, an identity or service provider initiates the profile by sending a `<ManageNameIDRequest>` message to another provider that it wishes to inform of a change. The request may be sent directly to the responding provider or sent indirectly through the user agent.

2) **<ManageNameIDResponse> issued by responding identity/service provider**

   In step 2, the responding provider (after processing the request) issues a `<ManageNameIDResponse>` message to the original requesting provider. The response may be returned directly to the requesting provider or indirectly through the user agent (if consistent with the form of the request in step 1).

11.4.5.3 Profile description

In the descriptions below, the following is referred to:

Name identifier management service

This is the name identifier management protocol endpoint at an identity or service provider to which the `<ManageNameIDRequest>` or `<ManageNameIDResponse>` messages (or an artifact representing them) are delivered. The same or different endpoints may be used for requests and responses.

11.4.5.3.1 <ManageNameIDRequest> issued by requesting identity/service provider

To initiate the profile, the requesting provider issues a `<ManageNameIDRequest>` message to another provider's name identifier management service request endpoint. Metadata may be used to determine the location of this endpoint and the bindings supported by the responding provider.

– **Synchronous bindings (Back-channel)**

  The requesting provider may use a synchronous binding, such as the SOAP binding (see clause 10), to send the request directly to the other provider. The requester must authenticate itself to the other provider, either by signing the `<ManageNameIDRequest>` or using any other binding-supported mechanism.

– **Asynchronous bindings (Front-channel)**

  Alternatively, the requesting provider may (if the principal's user agent is present) use an asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings (see clause 10) to send the request to the other provider through the user agent.

If the HTTP Redirect or POST binding is used, then the `<ManageNameIDRequest>` message is delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile defined in 11.4.6 is used by the other provider, which makes a callback to the requesting provider to retrieve the `<ManageNameIDRequest>` message, using for example the SOAP binding.

It is recommended that the HTTP exchanges in this step be made over TLS 1.0 to maintain confidentiality and message integrity. The `<ManageNameIDRequest>` message must be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.

Each of these bindings provide a RelayState mechanism that the requesting provider may use to associate the profile exchange with the original request. The requesting provider should reveal as little information as possible in the `RelayState` value unless the use of the profile does not require such privacy measures.

Profile-specific rules for the contents of the `<ManageNameIDRequest>` message are included in 11.4.5.4.1.

11.4.5.3.2 <ManageNameIDResponse> issued by responding identity/service provider

The recipient must process the `<ManageNameIDRequest>` message. After processing the message or upon encountering an error, the recipient must issue a `<ManageNameIDResponse>` message containing an appropriate status code to the requesting provider to complete the SAML protocol exchange.

– **Synchronous bindings (Back-channel)**

  If the requesting provider used a synchronous binding, such as the SOAP binding (see clause 10), the response is returned directly to complete the synchronous communication. The responder must authenticate itself to the requesting provider, either by signing the `<ManageNameIDResponse>` or using any other binding-supported mechanism.

– **Asynchronous bindings (Front-channel)**

  If the requesting provider used an asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings (see clause 10), then the `<ManageNameIDResponse>` (or artifact) is returned through the user agent to the requesting provider's name identifier management service response endpoint. Metadata may be used to determine the location of this endpoint and the bindings supported by the requesting provider. Any binding supported by both entities may be used.

If the HTTP Redirect or POST binding is used, then the `<ManageNameIDResponse>` message is delivered to the requesting provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile defined in 11.4.6 is used by the requesting provider, which makes a callback to the responding provider to retrieve the `<ManageNameIDResponse>` message, using for example the SOAP binding.

It is recommended that the HTTP exchanges in this step be made over TLS 1.0 to maintain confidentiality and message integrity. The `<ManageNameIDResponse>` message must be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the response issuer when the artifact is dereferenced.

Profile-specific rules for the contents of the `<ManageNameIDResponse>` message are included in 11.4.5.4.2.

11.4.5.4 Use of name identifier management protocol

This clause covers `ManageNameIDRequest` and `ManageNameIDResponse` usage.

11.4.5.4.1 <ManageNameIDRequest> usage

The `<Issuer>` element must be present and must contain the unique identifier of the requesting entity; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

The requester must authenticate itself to the responder and ensure message integrity, either by signing the message or using a binding-specific mechanism.

11.4.5.4.2 <ManageNameIDResponse> usage

The `<Issuer>` element must be present and must contain the unique identifier of the responding entity; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

The responder must authenticate itself to the requester and ensure message integrity, either by signing the message or using a binding-specific mechanism.

11.4.5.5 Use of metadata

The endpoint element `<md:ManageNameIDService>` describes supported bindings and location(s) to which an entity may send requests and responses using this profile. A requester, if encrypting the principal's identifier, can use the responder's `<md:KeyDescriptor>` element with a use attribute of encryption to determine an appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

11.4.6 Artifact Resolution profile

Clause 10 defines an Artifact Resolution protocol for dereferencing a SAML artifact into a corresponding protocol message. The HTTP Artifact binding in (see clause 10) leverages this mechanism to pass SAML protocol messages by reference. This profile describes the use of this protocol with a synchronous binding, such as the SOAP binding defined in clause 10.

11.4.6.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:artifact`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None

11.4.6.2 Profile overview

The message exchange and basic processing rules that govern this profile are largely defined in clause 8 that defines the messages to be exchanged, in combination with the binding used to exchange the messages. Clause 10 defines the binding of the message exchange to SOAP V1.1. Unless specifically noted in this Recommendation, all requirements defined in those specifications apply.

Figure 11-5 illustrates the basic template for the artifact resolution profile.

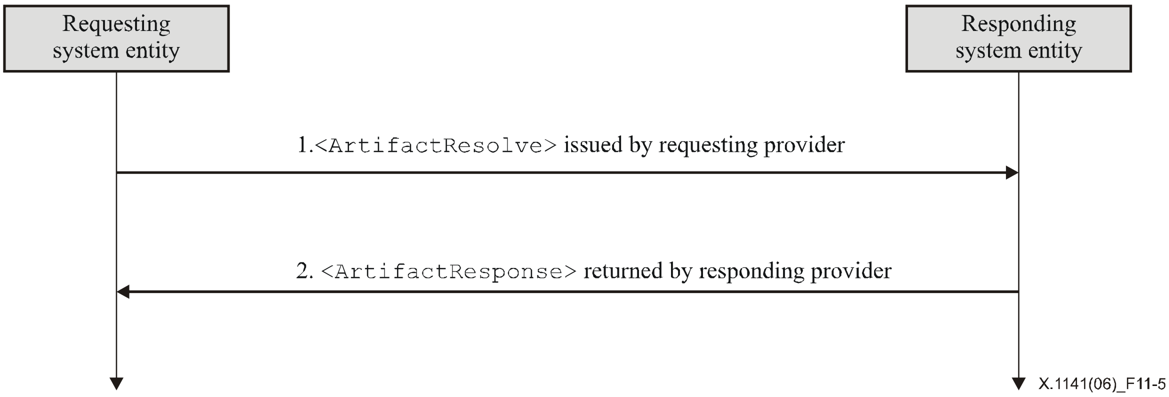
Requesting system entity
Responding system entity
1.<ArtifactResolve> issued by requesting provider
2. <ArtifactResponse> returned by responding provider
X.1141(06)_F11-5

Figure 11-5/X.1141 – Basic template artifact resolution profile

The following steps are described by the profile.

1) **<ArtifactResolve> issued by requesting entity**

   In step 1, a requester initiates the profile by sending an `<ArtifactResolve>` message to an artifact issuer.

2) **<ArtifactResponse> issued by responding entity**

   In step 2, the responder (after processing the request) issues an `<ArtifactResponse>` message to the requester.

11.4.6.3 Profile description

In the descriptions below, the following is referred to:

– **Artifact resolution service**

  This is the artifact resolution protocol endpoint at an artifact issuer to which `<ArtifactResolve>` messages are delivered.

11.4.6.3.1 <ArtifactResolve> issued by requesting entity

To initiate the profile, a requester, having received an artifact and determined the issuer using the `SourceID`, sends an `<ArtifactResolve>` message containing the artifact to an artifact issuer's artifact resolution service endpoint. Metadata may be used to determine the location of this endpoint and the bindings supported by the artifact issuer.

The requester must use a synchronous binding, such as the SOAP binding (see clause 10), to send the request directly to the artifact issuer. The requester should authenticate itself to the responder, either by signing the `<ArtifactResolve>` message or using any other binding-supported mechanism. Specific profiles that use the HTTP Artifact binding may impose additional requirements such that authentication is mandatory.

Profile-specific rules for the contents of the `<ArtifactResolve>` message are included in 11.4.6.4.1.

11.4.6.3.2 <ArtifactResponse> issued by responding entity

The artifact issuer must process the `<ArtifactResolve>` message as defined in clause 8. After processing the message or upon encountering an error, the artifact issuer must return an `<ArtifactResponse>` message containing an appropriate status code to the requester to complete the SAML protocol exchange. If successful, the dereferenced SAML protocol message corresponding to the artifact will also be included.

The responder must authenticate itself to the requester, either by signing the `<ArtifactResponse>` or using any other binding-supported mechanism.

Profile-specific rules for the contents of the `<ArtifactResponse>` message are included in 11.4.6.4.2.

11.4.6.4 Use of Artifact Resolution protocol

This clause covers `ArtifactResolve` and `ArtifactResponse` usage.

11.4.6.4.1 <ArtifactResolve> usage

The `<Issuer>` element must be present and must contain the unique identifier of the requesting entity; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

The requester should authenticate itself to the responder and ensure message integrity, either by signing the message or using a binding-specific mechanism. Specific profiles that use the HTTP Artifact binding may impose additional requirements such that authentication is mandatory.

11.4.6.4.2 <ArtifactResponse> usage

The `<Issuer>` element must be present and must contain the unique identifier of the artifact issuer; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity.`

The responder must authenticate itself to the requester and ensure message integrity, either by signing the message or using a binding-specific mechanism.

11.4.6.5 Use of metadata

Clause 9 defines an indexed endpoint element, `<md:ArtifactResolutionService>`, to describe supported bindings and location(s) to which a requester may send requests using this profile. The `index` attribute is used to distinguish the possible endpoints that may be specified by reference in the artifact's `EndpointIndex` field.

11.4.7 Assertion query/request profile

Clause 10 defines a protocol for requesting existing assertions by reference or by querying on the basis of a subject and additional statement-specific criteria. This profile describes the use of this protocol with a synchronous binding, such as the SOAP binding defined in clause 10.

11.4.7.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:query`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

11.4.7.2 Profile overview

The message exchange and basic processing rules that govern this profile are largely defined in clause 8 that defines the messages to be exchanged, in combination with the binding used to exchange the messages. Clause 10 defines the binding of the message exchange to SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

Figure 11-6 illustrates the basic template for the query/request profile.

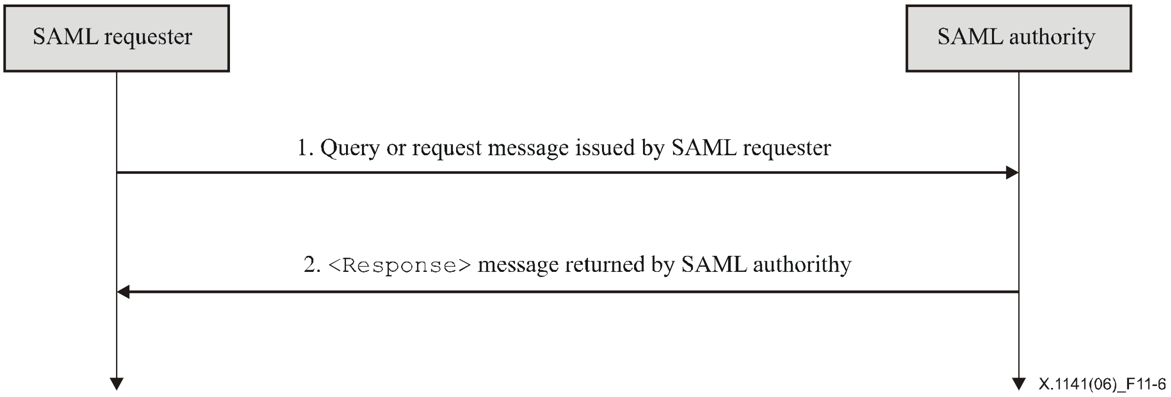
SAML requester
SAML authority
1. Query or request message issued by SAML requester
2. <Response> message returned by SAML authorithy
X.1141(06)_F11-6

Figure 11-6/X.1141 – Basic template query/request profile

The following steps are described by the profile:

1) **Query/request issued by SAML requester**

   In step 1, a SAML requester initiates the profile by sending an `<AssertionIDRequest>`, `<SubjectQuery>`, `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>` message to a SAML authority.

2) **<Response> issued by SAML authority**

   In step 2, the responding SAML authority (after processing the query or request) issues a `<Response>` message to the SAML requester.

11.4.7.3 Profile description

In the descriptions below, the following are referred to:

– **Query/request service**

  This is the query/request protocol endpoint at a SAML authority to which query or `<AssertionIDRequest>` messages are delivered.

11.4.7.3.1 Query/request issued by SAML requester

To initiate the profile, a SAML requester issues an `<AssertionIDRequest>`, `<SubjectQuery>`, `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>` message to a SAML authority's query/request service endpoint. Metadata may be used to determine the location of this endpoint and the bindings supported by the SAML authority.

The SAML requester must use a synchronous binding, such as the SOAP binding (see clause 10), to send the request directly to the identity provider. The requester should authenticate itself to the SAML authority either by signing the message or using any other binding-supported mechanism.

Profile-specific rules for the contents of the various messages are included in 11.4.7.4.1.

11.4.7.3.2 <Response> issued by SAML authority

The SAML authority must process the query or request message as defined in clause 8. After processing the message or upon encountering an error, the SAML authority must return a `<Response>` message containing an appropriate status code to the SAML requester to complete the SAML protocol exchange. If the request is successful in locating one or more matching assertions, they will also be included in the response.

The responder should authenticate itself to the requester, either by signing the `<Response>` or using any other binding-supported mechanism.

Profile-specific rules for the contents of the `<Response>` message are included in 11.4.7.4.2.

11.4.7.4 Use of query/request protocol

This clause defines the query/request protocol endpoint at a SAML authority to which query messages are delivered.

11.4.7.4.1 Query/request usage

The `<Issuer>` element must be present.

The requester should authenticate itself to the responder and ensure message integrity, either by signing the message or using a binding-specific mechanism.

11.4.7.4.2 <Response> usage

The `<Issuer>` element must be present and must contain the unique identifier of the responding SAML authority; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`. This need not necessarily match the `<Issuer>` element in the returned assertion(s).

The responder should authenticate itself to the requester and ensure message integrity, either by signing the message or using a binding-specific mechanism.

11.4.7.5 Use of metadata

Clause 9 defines several endpoint elements, `<md:AssertionIDRequestService>`, `<md:AuthnQueryService>`, `<md:AttributeService>`, and `<md:AuthzService>`, to describe supported bindings and location(s) to which a requester may send requests or queries using this profile.

The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can use that entity's `<md:KeyDescriptor>` element with a use attribute of `encryption` to determine an appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

The various role descriptors may contain `<md:NameIDFormat>`, `<md:AttributeProfile>`, and `<saml:Attribute>` elements (as applicable) to indicate the general ability to support particular name identifier formats, attribute profiles, or specific attributes and values. The ability to support any such features during a given request is dependent on policy and the discretion of the authority.

11.4.8 Name identifier mapping profile

Clause 8.2.6 defines a Name Identifier Mapping protocol for mapping a principal's name identifier into a different name identifier for the same principal. This profile describes the use of this protocol with a synchronous binding, such as the SOAP binding defined in clause 10, and additional guidelines for protecting the privacy of the principal with encryption and limiting the use of the mapped identifier.

11.4.8.1 Required information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:nameidmapping`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

11.4.8.2 Profile overview

The message exchange and basic processing rules that govern this profile are largely defined in clause 8 which defines the messages to be exchanged, in combination with the binding used to exchange the messages. Clause 10 defines the binding of the message exchange to SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

Figure 11-7 illustrates the basic template for the name identifier mapping profile.

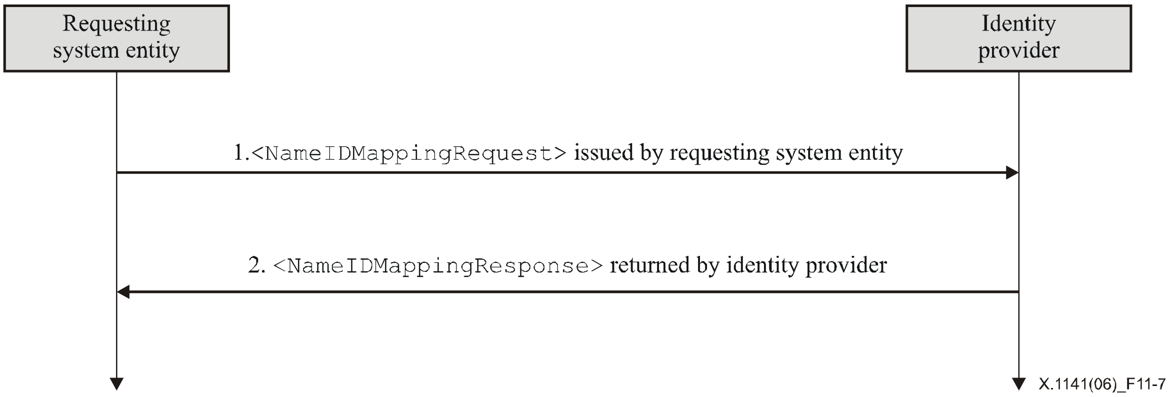
Requesting system entity
Identity provider
1.<NameIDMappingRequest> issued by requesting system entity
2. <NameIDMappingResponse> returned by identity provider
X.1141(06)_F11-7

Figure 11-7/X.1141 – Basic template name identifier profile

The following steps are described by the profile:

1) **<NameIDMappingRequest> issued by requesting entity**

   In step 1, a requester initiates the profile by sending a `<NameIDMappingRequest>` message to an identity provider.

2) **<NameIDMappingResponse> issued by identity provider**

   In step 2, the responding identity provider (after processing the request) issues a `<NameIDMappingResponse>` message to the requester.

11.4.8.3 Profile description

This clause uses name identifier mapping service, which is the name identifier mapping protocol endpoint at an identity provider to which `<NameIDMappingRequest>` messages are delivered.

11.4.8.3.1 <NameIDMappingRequest> issued by requesting entity

To initiate the profile, a requester issues a `<NameIDMappingRequest>` message to an identity provider's name identifier mapping service endpoint. Metadata may be used to determine the location of this endpoint and the bindings supported by the identity provider.

The requester must use a synchronous binding, such as the SOAP binding (see clause 10), to send the request directly to the identity provider. The requester must authenticate itself to the identity provider, either by signing the `<NameIDMappingRequest>` or using any other binding-supported mechanism.

Profile-specific rules for the contents of the `<NameIDMappingRequest>` message are included in 11.4.8.4.1.

11.4.8.3.2 <NameIDMappingResponse> issued by identity provider

The identity provider must process the `<ManageNameIDRequest>` message as defined in clause 8. After processing the message or upon encountering an error, the identity provider must return a `<NameIDMappingResponse>` message containing an appropriate status code to the requester to complete the SAML protocol exchange.

The responder must authenticate itself to the requester, either by signing the `<NameIDMappingResponse>` or using any other binding-supported mechanism.

Profile-specific rules for the contents of the `<NameIDMappingResponse>` message are included in 11.4.8.4.2.

11.4.8.4 Use of name identifier mapping protocol

Clause 8 defines a name identifier mapping protocol for mapping a principal's name identifier into a different name identifier for the same principal. This clause describes the use of this protocol and additional guidelines for protecting the privacy of the principal such as limiting the use of the mapped identifier.

11.4.8.4.1 <NameIDMappingRequest> usage

The `<Issuer>` element must be present.

The requester must authenticate itself to the responder and ensure message integrity, either by signing the message or using a binding-specific mechanism.

11.4.8.4.2 <NameIDMappingResponse> usage

The `<Issuer>` element must be present and must contain the unique identifier of the responding identity provider; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

The responder must authenticate itself to the requester and ensure message integrity, either by signing the message or using a binding-specific mechanism.

W3C Encryption, 2.2.3, defines the use of encryption to apply confidentiality to a name identifier. In most cases, the identity provider should encrypt the mapped name identifier it returns to the requester to protect the privacy of the principal. The requester can extract the `<EncryptedID>` element and place it in subsequent protocol messages or assertions.

Limiting use of mapped identifier

Additional limits on the use of the resulting identifier may be applied by the identity provider by returning the mapped name identifier in the form of an `<Assertion>` containing the identifier in its `<Subject>` but without any statements. The assertion is then encrypted and the result used as the `<EncryptedData>` element in the `<EncryptedID>` returned to the requester. The assertion may include a `<Conditions>` element to limit use, as defined by clause 8, such as time-based constraints or use by specific relying parties, and must be signed for integrity protection.

11.4.8.5 Use of metadata

This clause defines an endpoint element, `<md:NameIDMappingService>`, to describe supported bindings and location(s) to which a requester may send requests using this profile.

The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity's `<md:KeyDescriptor>` element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

11.4.9 SAML attribute profiles

Attribute profiles provide the definitions necessary to constrain SAML attribute expression when dealing with particular types of attribute information or when interacting with external systems that require greater strictness. This subclause specifies SAML basic attribute profile, X.500/LDAP profile and UUID profiles and XACML profile.

11.4.9.1 Basic attribute profile

The `Basic` attribute profile specifies simplified, but non-unique, naming of SAML attributes together with attribute values based on the built-in W3C Datatypes data types, eliminating the need for extension schemas to validate syntax.

Required information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:attribute:basic`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

SAML attribute naming

The `NameFormat` XML attribute in `<Attribute>` elements must be `urn:oasis:names:tc:SAML:2.0:attrname-format:basic`.

The `Name` XML attribute must adhere to the rules specified for that format, as defined by clause 8.

– **Attribute name comparison**

  Two `<Attribute>` elements refer to the same SAML attribute if, and only if, the values of their `Name` XML attributes are equal (in the sense that it is described in clause 8).

Profile-specific XML attributes

No additional XML attributes are defined for use with the `<Attribute>` element.

SAML attribute values

The schema type of the contents of the `<AttributeValue>` element must be drawn from one of the types defined in Annex A. The `xsi:type` attribute must be present and be given the appropriate value.

Example

```
<saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic"
             Name="FirstName">
       <saml:AttributeValue xsi:type="xs:string">By-
Tor</saml:AttributeValue>
</saml:Attribute>
```

11.4.9.2 X.500/LDAP attribute profile

Directories based on the set of ITU-T X.500-series and on IETF RFC 3377 are widely deployed. Directory schema is used to model information to be stored in these directories. In particular, in X.500, attribute type definitions are used to specify the syntax and other features of attributes, the basic information storage unit in a directory (this Recommendation refers to these as "directory attributes"). Directory attribute types are defined in schema in the X.500 and LDAP specifications themselves, schema in other public documents (such as the inetOrgperson schema (see IETF RFC 2798)), and schema defined for private purposes. In any of these cases, it is useful for deployers to take advantage of these directory attribute types in the context of SAML attribute statements, without having to manually create SAML-specific attribute definitions for them, and to do this in an interoperable fashion.

The X.500/LDAP attribute profile defines a common convention for the naming and representation of such attributes when expressed as SAML attributes.

Required information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500` (this is also the target namespace assigned in the corresponding X.500/LDAP profile schema in Annex A).

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

SAML attribute naming

The `NameFormat` XML attribute in `<Attribute>` elements must be `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

To construct attribute names, the URN `oid` namespace described in IETF RFC 3061 is used. In this approach the `Name` XML attribute is based on the object identifier assigned to the directory attribute type.

Example:

`urn:oid:2.5.4.3`

Since X.500 procedures require that every attribute type be identified with a unique object identifier, this naming scheme ensures that the derived SAML attribute names are unambiguous.

For purposes of human readability, there may also be a requirement for some applications to carry an optional string name together with the OID URN (as defined in IETF RFC 3061). The optional XML attribute `FriendlyName` (defined in clause 8) may be used for this purpose. If the definition of the directory attribute type includes one or more descriptors (short names) for the attribute type, the `FriendlyName` value, if present, should be one of the defined descriptors.

Two `<Attribute>` elements refer to the same SAML attribute if, and only if, their `Name` XML attribute values are equal in the sense of IETF RFC 3061. The `FriendlyName` attribute plays no role in the comparison.

Profile-Specific XML attributes

No additional XML attributes are defined for use with the `<Attribute>` element.

SAML attribute values

Directory attribute type definitions for use in native X.500 directories specify the syntax of the attribute using ASN.1. For use in LDAP, directory attribute definitions additionally include an LDAP syntax which specifies how attribute or assertion values conforming to the syntax are to be represented when transferred in the LDAP protocol (known as an LDAP-specific encoding). The LDAP-specific encoding commonly produces Unicode characters in UTF-8 form. This SAML attribute profile specifies the form of SAML attribute values only for those directory attributes which have LDAP syntaxes. Future extensions to this profile may define attribute value formats for directory attributes whose syntaxes specify other encodings.

To represent the encoding rules in use for a particular attribute value, the `<AttributeValue>` element must contain an XML attribute named `Encoding` defined in the XML namespace `urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500`.

For any directory attribute with a syntax whose LDAP-specific encoding exclusively produces UTF-8 character strings as values, the SAML attribute value is encoded as simply the UTF-8 string itself, as the content of the `<AttributeValue>` element, with no additional whitespace. In such cases, the `xsi:type` XML attribute must be set to **xs:string**. The profile-specific `Encoding` XML attribute is provided, with a value of `LDAP`.

A list of some LDAP attribute syntaxes (and associated OID) to which this applies is:

Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3
Bit String	1.3.6.1.4.1.1466.115.121.1.6
Boolean	1.3.6.1.4.1.1466.115.121.1.7
Country String	1.3.6.1.4.1.1466.115.121.1.11
DN	1.3.6.1.4.1.1466.115.121.1.12
Directory String	1.3.6.1.4.1.1466.115.121.1.15
Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24
IA5 String	1.3.6.1.4.1.1466.115.121.1.26
INTEGER	1.3.6.1.4.1.1466.115.121.1.27
LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54
Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30
Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31
Name And Optional UID	1.3.6.1.4.1.1466.115.121.1.34
Name Form Description	1.3.6.1.4.1.1466.115.121.1.35
Numeric String	1.3.6.1.4.1.1466.115.121.1.36
Object Class Description	1.3.6.1.4.1.1466.115.121.1.37
Octet String	1.3.6.1.4.1.1466.115.121.1.40
OID	1.3.6.1.4.1.1466.115.121.1.38
Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39
Postal Address	1.3.6.1.4.1.1466.115.121.1.41
Presentation Address	1.3.6.1.4.1.1466.115.121.1.43
Printable String	1.3.6.1.4.1.1466.115.121.1.44
Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58
Telephone Number	1.3.6.1.4.1.1466.115.121.1.50
UTC Time	1.3.6.1.4.1.1466.115.121.1.53

For all other LDAP syntaxes, the attribute value is encoded, as the content of the `<AttributeValue>` element, by base64-encoding the encompassing ASN.1 octet string-encoded LDAP attribute value. The `xsi:type` XML attribute must be set to **xs:base64Binary**. The profile-specific `Encoding` XML attribute is provided, with a value of "`LDAP`".

When comparing SAML attribute values for equality, the matching rules specified for the corresponding directory attribute type must be observed (case sensitivity, for example).

Profile-specific schema

The following schema listing shows how the profile-specific `Encoding` XML attribute is defined (see Annex A):

```
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <annotation>
        <documentation>
            Document identifier: saml-schema-x500-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
              V2.0 (March, 2005):
                Custom schema for X.500 attribute profile, first published
in SAML 2.0.
        </documentation>
    </annotation>
    <attribute name="Encoding" type="string"/>
</schema>
```

Example

The following is an example of a mapping of the "givenName" directory attribute, representing the SAML assertion subject's first name. Its object identifier is {joint-iso-itu-t(2) ds(5) attributeType(4) givenName(42)} and its LDAP syntax is Directory String:

```
<saml:Attribute
xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
             Name="urn:oid:2.5.4.42" FriendlyName="givenName">
       <saml:AttributeValue xsi:type="xs:string"
             x500:Encoding="LDAP">Steven</saml:AttributeValue>
</saml:Attribute>
```

11.4.9.3 UUID attribute profile

The UUID attribute profile standardizes the expression of UUID values as SAML attribute names and values. It is applicable when the attribute's source system is one that identifies an attribute or its value with a UUID.

Required information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:attribute:UUID`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

SAML attribute naming

The `NameFormat` XML attribute in `<Attribute>` elements must be `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

If the underlying representation of the attribute's name is a UUID, then the URN `uuid` namespace described in ITU-T Rec. X.667 is used. In this approach the `Name` XML attribute is based on the URN form of the underlying UUID that identifies the attribute.

Example:

```
urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

If the underlying representation of the attribute's name is not a UUID, then any form of URI may be used in the `Name` XML attribute.

For purposes of human readability, there may also be a requirement for some applications to carry an optional string name together with the URI. The optional XML attribute `FriendlyName` may be used for this purpose.

Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute values are equal in the sense of ITU-T Rec. X.667. The `FriendlyName` attribute plays no role in the comparison.

Profile-specific XML attributes

No additional XML attributes are defined for use with the `<Attribute>` element.

SAML attribute values

In cases in which the attribute's value is also a UUID, the same URN syntax described above must be used to express the value within the `<AttributeValue>` element. The `xsi:type` XML attribute must be set to **xs:anyURI**.

If the attribute's value is not a UUID, then there are no restrictions on the use of the `<AttributeValue>` element.

Example

The following is an example of a DCE Extended Registry Attribute, the "pre_auth_req" setting, which has a well-known UUID of 6c9d0ec8-dd2d-11cc-abdd-080009353559 and is integer-valued.

```
<saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"
             Name="urn:uuid:6c9d0ec8-dd2d-11cc-abdd-080009353559"
             FriendlyName="pre_auth_req">
      <saml:AttributeValue xsi:type="xs:integer">1</saml:AttributeValue>
</saml:Attribute>
```

11.4.9.4 XACML attribute profile

SAML attribute assertions may be used as input to authorization decisions made according to ITU-T Rec. X.1142. Since the SAML attribute format differs from the XACML attribute format, there is a mapping that must be performed. The XACML attribute profile facilitates this mapping by standardizing naming, value syntax, and additional attribute metadata. SAML attributes generated in conformance with this profile can be mapped automatically into XACML attributes and used as input to XACML authorization decisions.

Required information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML` (this is also the target namespace assigned in the corresponding XACML profile schema in Annex A).

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

SAML attribute naming

The `NameFormat` XML attribute in `<Attribute>` elements must be `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

The `Name` XML attribute must adhere to the rules specified for that format, as defined in clause 8.

For purposes of human readability, there may also be a requirement for some applications to carry an optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in clause 8) may be used for this purpose, but is not translatable into an XACML attribute equivalent.

Two `<Attribute>` elements refer to the same SAML attribute if, and only if, their `Name` XML attribute values are equal in a binary comparison. The `FriendlyName` attribute plays no role in the comparison.

Profile-specific XML attributes

XACML requires each attribute to carry an explicit data type. To supply this data type value, a new URI-valued XML attribute called `DataType` is defined in the XML namespace `urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML`.

SAML `<Attribute>` elements conforming to this profile must include the namespace-qualified `DataType` attribute, or the value is presumed to be `http://www.w3.org/2001/XMLSchema#string`.

If non-standard values are used, then each XACML PDP that will be consuming mapped SAML attributes with non-standard `DataType` values must be extended to support the new data types.

SAML attribute values

The syntax of the `<AttributeValue>` element's content must correspond to the data type expressed in the profile-specific `DataType` XML attribute appearing in the parent `<Attribute>` element. For data types corresponding to the types defined in clause 8, the `xsi:type` XML attribute should also be used on the `<AttributeValue>` element(s).

Profile-specific schema

The following schema listing shows how the profile-specific `DataType XML` attribute is defined (Annex A):

```
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <annotation>
        <documentation>
            Document identifier: saml-schema-xacml-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
            V2.0 (March, 2005):
              Custom schema for XACML attribute profile, first published in
SAML 2.0.
        </documentation>
    </annotation>
    <attribute name="DataType" type="anyURI"/>
</schema>
```

Example

The following is an example of a mapping of the "givenName" LDAP/X.500 attribute, representing the SAML assertion subject's first name. It also illustrates that a single SAML attribute can conform to multiple attribute profiles when they are compatible with each other.

```
<saml:Attribute
xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
       xmlns:ldapprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"
              xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"
              ldapprof:Encoding="LDAP"
              NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
              Name="urn:oid:2.5.4.42" FriendlyName="givenName">
       <saml:AttributeValue xsi:type="xs:string">By-
Tor</saml:AttributeValue>
</saml:Attribute>
```

NOTE (informative) – PE39 (see OASIS PE:2006) clarifies the above example as below:

```
<saml:Attribute
  xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
     xmlns:ldapprof="urn:oasis:names:tc:SAML:2.0:profiles:AttributeValue:LDAP"
          xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"
          ldapprof:Encoding="LDAP"
          NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
          Name="urn:oid:2.5.4.42" FriendlyName="givenName">
     <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>
</saml:Attribute>
```

12 SAML authentication context

This Recommendation defines a syntax for the definition of authentication context declarations and an initial list of authentication context classes.

12.1 Authentication context concepts

If a relying party is to rely on the authentication of a principal by an authentication authority, the relying party may require information additional to the assertion itself in order to assess the level of confidence they can place in that assertion. This Recommendation defines an XML Schema for the creation of Authentication Context declarations – XML documents that allow the authentication authority to provide to the relying party this additional information.

Additionally, this Recommendation defines a number of Authentication Context classes; categories into which many Authentication Context declarations will fall, thereby simplifying their interpretation.

SAML does not prescribe a single technology, protocol, or policy for the processes by which authentication authorities issue identities to principals and by which those principals subsequently authenticate themselves to the authentication authority. Different authentication authorities will choose different technologies, follow different processes, and be bound by different legal obligations with respect to how they authenticate principals.

The choices that an authentication authority makes here will be driven in large part by the requirements of the relying parties with which the authentication authority interacts. These requirements themselves will be determined by the nature of the service (that is, the sensitivity of any information exchanged, the associated financial value, the relying parties' risk tolerance, etc.) that the relying party will be providing to the principal.

Consequently, for anything other than trivial services, if the relying party is to place sufficient confidence in the authentication assertions it receives from an authentication authority, it will be necessary for it to know which technologies, protocols, and processes were used or followed for the original authentication mechanism on which the authentication assertion is based. Armed with this information and trusting the origin of the actual assertion, the relying party will be better able to make an informed entitlements decision regarding what services the subject of the authentication assertion should be allowed to access.

Authentication context is defined as the information, additional to the authentication assertion itself, that the relying party may require before it makes an entitlements decision with respect to an authentication assertion. Such context may include, but is not limited to, the actual authentication method used.

12.2 Authentication context declaration

If a relying party is to rely on the authentication of another entity by an authentication authority, the relying party may require information additional to the authentication itself to allow it to put the authentication into a risk-management context. This information could include:

- The initial user identification mechanisms (for example, face-to-face, online, shared secret).
- The mechanisms for minimizing compromise of credentials (for example, credential renewal frequency, client-side key generation).
- The mechanisms for storing and protecting credentials (for example, smartcard, password rules).
- The authentication mechanism or method (for example, password).

The variations and permutations in the characteristics listed above guarantee that not all authentication assertions will be the same with respect to the confidence that a relying party can place in it; a particular authentication assertion will be characterized by the values for each of these (and other) variables.

A SAML authentication authority can deliver to a relying party the additional authentication context information in the form of an authentication context declaration, an XML document either inserted directly or referenced within the authentication assertion that the authentication authority provides to the relying party.

SAML requesters are able to request that an authentication comply with a specified authentication context by identifying that context in an authentication request. A requester may also specify that an authentication must be conducted with an authentication context that *exceeds* some stated value (for some agreed definition of "exceeds").

12.2.1 Data model

A particular authentication context declaration defined in this Recommendation will capture characteristics of the processes, procedures, and mechanisms by which the authentication authority verified the subject before issuing an identity, protects the secrets on which subsequent authentications are based, and the mechanisms used for this authentication. These characteristics are categorized in the Authentication Context schema as follows:

- Identification – Characteristics that describe the processes and mechanism the authentication authority uses to initially create an association between a subject and the identity (or name) by which the subject will be known.
- Technical Protection – Characteristics that describe how the "secret" (the knowledge or possession of which allows the subject to authenticate to the authentication authority) is kept secure.
- Operational Protection – Characteristics that describe procedural security controls employed by the authentication authority (for example, security audits, records archival).
- Authentication Method – Characteristics that define the mechanisms by which the subject of the issued assertion authenticates to the authentication authority (for example, a password versus a smartcard).

- Governing Agreements – Characteristics that describe the legal framework (e.g., liability constraints and contractual obligations) underlying the authentication event and/or its associated technical authentication infrastructure.

12.2.2 Extensibility

The authentication context declaration schema has well-defined extensibility points through the `<Extension>` element. Authentication authorities can use this element to insert additional authentication context details for the SAML assertions they issue (assuming that the consuming relying party will be able to understand these extensions). These additional elements must be in a separate XML Namespace to that of the authentication context declaration base or class schema that applies to the declaration itself.

12.2.3 Processing rules

Additional processing rules for authentication context declarations are specified in clause 8, these processing rules amount to deployments sharing common interpretations of the relative strength or quality of particular authentication context declarations and cannot be expressed in absolute terms or provided as rules that implementations must follow.

12.2.4 Schema

This clause in non-normative.

Listing of a complete Authentication Context Types XML Schema and the Authentication Context XML schema itself, used for the validation of individual generalized declarations, is provided in Appendix VI.

12.3 Authentication context classes

The number of permutations of different characteristics ensures that there is a theoretically infinite number of unique authentication contexts. The implication is that, in theory, any particular relying party would be expected to be able to parse arbitrary authentication context declarations and, more importantly, to analyse the declaration in order to assess the "quality" of the associated authentication assertion. Making such an assessment is non-trivial.

Fortunately, an optimization is possible. In practice many authentication contexts will fall into categories determined by industry practices and technology. For instance, many B2C web browser authentication contexts will be (partially) defined by the principal authenticating to the authentication authority through the presentation of a password over an TLS protected session. In the enterprise world, certificate-based authentication will be common. Of course, the full authentication context is not limited to the specifics of how the principal authenticated. Nevertheless, the authentication method is often the most *visible* characteristic and, as such, can serve as a useful classifier for a class of related authentication contexts.

The concept is expressed in this Recommendation as a definition of a series of a*uthentication context classes*. Each class defines a proper subset of the full set of authentication contexts. Classes have been chosen as representative of the current practices and technologies for authentication technologies, and provide asserting and relying parties a convenient shorthand when referring to authentication context issues.

For instance, an authentication authority may include with the complete authentication context declaration it provides to a relying party an assertion that the authentication context also belongs to an authentication context class. For some relying parties, this assertion is sufficient detail for it to be able to assign an appropriate level of confidence to the associated authentication assertion. Other relying parties might prefer to examine the complete authentication context declaration itself. Likewise, the ability to refer to an authentication context class rather than being required to list the complete details of a specific authentication context declaration will simplify how the relying party can express its desires and/or requirements to an authentication authority.

12.3.1 Advantages of authentication context classes

The introduction of the additional layer of classes and the definition of an initial list of representative and flexible classes are expected to:

- Make it easier for the authentication authority and relying party to come to an agreement on what are acceptable authentication contexts by giving them a framework for discussion.
- Make it easier for relying parties to indicate their preferences when requesting a step-up authentication assertion from an authentication authority.
- Simplify for relying parties the burden of processing authentication context declarations by giving them the option of being satisfied by the associated class.
- Insulate relying parties from the impact of new authentication technologies.

- Make it easier for authentication authorities to publish their authentication capabilities, for example, through WSDL.

12.3.2 Processing rules

Further processing rules for authentication context classes are described in clause 8. In most respects, these processing rules amount to deployments sharing common interpretations of the relative strength or quality of particular authentication context classes and cannot be expressed in absolute terms or provided as rules that implementations must follow.

12.3.3 Extensibility

As does the core authentication context declaration schema, the separate authentication context class schemas allow the `<Extension>` element in certain locations of the tree structure. In general, where the `<Extension>` element occurred as a child of an `<xs:choice>` element, this option was removed in creating the appropriate class schema definition as a restriction of the base type. When the `<Extension>` element occurred as an optional child of an `<xs:sequence>` element, the `<Extension>` element was allowed to remain in addition to any required elements.

Consequently, authentication context declarations can include the `<Extension>` element (with additional elements in different namespaces) and still conform to authentication context class schemas (if they meet the other requirements of the schema of course).

The authentication context class schemas restrict type definitions in the base authentication context schema. As an extension point, the authentication context class schemas themselves can be further restricted – their type definitions serving as base types in some other schema (potentially defined by some community wishing a more tightly defined authentication context class). To prevent logical inconsistencies, any such schema extensions can only further constrain the type definitions of the class schema. To enforce this constraint, the authentication context class schemas are defined with the `finalDefault="extension"` attribute on the `<schema>` element to prevent this type of derivation.

12.3.4 Schemas

Authentication context classes are listed in the following subclauses. The classes are listed in alphabetical order; no other ranking is implied by the order of classes. Implementers can choose which classes to support as guided by the conformance guidelines in this Recommendation (see clause 13). Classes are uniquely identified by URIs with the following initial stem:

```
urn:oasis:names:tc:SAML:2.0:ac:classes
```

The class schemas are defined as restrictions of parts of the base authentication context "types" schema. XML instances that validate against a given authentication context class schema are said to *conform* to that authentication context class.

Because the class schema imports and redefines the elements and types into the class schema namespace, a class-conforming authentication context declaration does not simultaneously validate against the base authentication context schema.

12.3.4.1 Internet protocol

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The Internet Protocol class is applicable when a principal is authenticated through the use of a provided IP address.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
  targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol
        Document identifier: saml-schema-authn-context-ip-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="IPAddress"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.2 InternetProtocolPassword

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocolPassword`

Note that this URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The Internet Protocol Password class is applicable when a principal is authenticated through the use of a provided IP address, in addition to a username/password.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
  targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocolPassword"
  xmlns:ac="urn:oasis:names:tc:SAML:2.0:ac"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocolPassword"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">
  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocolPassword
        Document identifier: saml-schema-authn-context-ippword-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="Password"/>
            <xs:element ref="IPAddress"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.3 Kerberos

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

This class is applicable when the principal has authenticated using a password to a local authentication authority, in order to acquire a Kerberos ticket. That Kerberos ticket is then used for subsequent network authentication.

> NOTE 1 – It is possible for the authentication authority to indicate (via this context class) a pre-authentication data type which was used by the Kerberos key distribution centre (IETF RFC 1510) when authenticating the principal. The method used by the authentication authority to obtain this information is outside of the scope of this Recommendation, but it is strongly

recommended that a trusted method be deployed to pass the pre-authentication data type and any other Kerberos related context details (e.g., ticket lifetime) to the authentication authority.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos
        Document identifier: saml-schema-authn-context-kerberos-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
              <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
```

```
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="SharedSecretChallengeResponse"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SharedSecretChallengeResponseType">
      <xs:complexContent>
        <xs:restriction base="SharedSecretChallengeResponseType">
          <xs:attribute name="method" type="xs:anyURI"
fixed="urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

An example of an XML instance conforming to this class schema is as follows:

```
<AuthenticationContextDeclaration
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos">

  <AuthnMethod>

    <PrincipalAuthenticationMechanism preauth="0">
      <RestrictedPassword>
        <Length min="4"/>
      </RestrictedPassword>
    </PrincipalAuthenticationMechanism>

    <Authenticator>
      <AuthenticatorSequence>
        <SharedSecretChallengeResponse
method="urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos"/>
      </AuthenticatorSequence>
    </Authenticator>

  </AuthnMethod>

</AuthenticationContextDeclaration>
```

NOTE 2 – The use of SSL is presented in Appendix IV.

12.3.4.4 MobileOneFactorUnregistered

URI: urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorUnregistered

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

Reflects no mobile customer registration procedures and an authentication of the mobile device without requiring explicit end-user interaction. This context class authenticates only the device and never the user; it is useful when services other than the mobile operator want to add a secure device authentication to their authentication process.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema

targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorUnre
gistered"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"

xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorUnregistered"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">
```

```
    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorUnregistered
        Document identifier: saml-schema-authn-context-mobileonefactor-
unreg-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"
minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="ZeroKnowledge"/>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
```

```
              <xs:element ref="SSL"/>
              <xs:element ref="MobileNetworkNoEncryption"/>
              <xs:element ref="MobileNetworkRadioEncryption"/>
              <xs:element ref="MobileNetworkEndToEndEncryption"/>
              <xs:element ref="WTLS"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="OperationalProtectionType">
      <xs:complexContent>
        <xs:restriction base="OperationalProtectionType">
          <xs:sequence>
            <xs:element ref="SecurityAudit"/>
            <xs:element ref="DeactivationCallCenter"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TechnicalProtectionBaseType">
      <xs:complexContent>
        <xs:restriction base="TechnicalProtectionBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PrivateKeyProtection"/>
              <xs:element ref="SecretKeyProtection"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrivateKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="PrivateKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SecretKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="SecretKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyStorageType">
      <xs:complexContent>
        <xs:restriction base="KeyStorageType">
          <xs:attribute name="medium" use="required">
            <xs:simpleType>
              <xs:restriction base="mediumType">
```

```
                <xs:enumeration value="MobileDevice"/>
                <xs:enumeration value="MobileAuthCard"/>
                <xs:enumeration value="smartcard"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SecurityAuditType">
      <xs:complexContent>
        <xs:restriction base="SecurityAuditType">
          <xs:sequence>
            <xs:element ref="SwitchAudit"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="IdentificationType">
      <xs:complexContent>
        <xs:restriction base="IdentificationType">
          <xs:sequence>
            <xs:element ref="GoverningAgreements"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="nym">
            <xs:simpleType>
              <xs:restriction base="nymType">
                <xs:enumeration value="anonymity"/>
                <xs:enumeration value="pseudonymity"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

NOTE – The use of SSL is presented in Appendix IV.

12.3.4.5 MobileTwoFactorUnregistered

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorUnregistered`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

Reflects no mobile customer registration procedures and a two-factor based authentication, such as secure device and user PIN. This context class is useful when a service other than the mobile operator wants to link their customer ID to a mobile supplied two-factor authentication service by capturing mobile phone data at enrolment.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorUnre
gistered"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"

xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorUnregistered"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">
```

```
    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorUnregistered
        Document identifier: saml-schema-authn-context-mobiletwofactor-
unreg-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"
minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="ZeroKnowledge"/>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
              <xs:element ref="ComplexAuthenticator"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ComplexAuthenticatorType">
      <xs:complexContent>
        <xs:restriction base="ComplexAuthenticatorType">
          <xs:sequence>
```

```
            <xs:choice>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
            </xs:choice>
            <xs:element ref="Password"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SSL"/>
              <xs:element ref="MobileNetworkNoEncryption"/>
              <xs:element ref="MobileNetworkRadioEncryption"/>
              <xs:element ref="MobileNetworkEndToEndEncryption"/>
              <xs:element ref="WTLS"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="OperationalProtectionType">
      <xs:complexContent>
        <xs:restriction base="OperationalProtectionType">
          <xs:sequence>
            <xs:element ref="SecurityAudit"/>
            <xs:element ref="DeactivationCallCenter"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TechnicalProtectionBaseType">
      <xs:complexContent>
        <xs:restriction base="TechnicalProtectionBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PrivateKeyProtection"/>
              <xs:element ref="SecretKeyProtection"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrivateKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="PrivateKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyActivation"/>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SecretKeyProtectionType">
      <xs:complexContent>
```

```
        <xs:restriction base="SecretKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyActivation"/>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyStorageType">
      <xs:complexContent>
        <xs:restriction base="KeyStorageType">
          <xs:attribute name="medium" use="required">
            <xs:simpleType>
              <xs:restriction base="mediumType">
                <xs:enumeration value="MobileDevice"/>
                <xs:enumeration value="MobileAuthCard"/>
                <xs:enumeration value="smartcard"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SecurityAuditType">
      <xs:complexContent>
        <xs:restriction base="SecurityAuditType">
          <xs:sequence>
            <xs:element ref="SwitchAudit"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="IdentificationType">
      <xs:complexContent>
        <xs:restriction base="IdentificationType">
          <xs:sequence>
            <xs:element ref="GoverningAgreements"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="nym">
            <xs:simpleType>
              <xs:restriction base="nymType">
                <xs:enumeration value="anonymity"/>
                <xs:enumeration value="pseudonymity"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.6 MobileOneFactorContract

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorContract`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

Reflects mobile contract customer registration procedures and a single factor authentication. For example, a digital signing device with tamper resistant memory for key storage, such as the mobile MSISDN, but no required PIN or biometric for real-time user authentication.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema

targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorContract
"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorContract"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorContract
        Document identifier: saml-schema-authn-context-mobileonefactor-reg-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="ZeroKnowledge"/>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SSL"/>
              <xs:element ref="MobileNetworkNoEncryption"/>
              <xs:element ref="MobileNetworkRadioEncryption"/>
              <xs:element ref="MobileNetworkEndToEndEncryption"/>
              <xs:element ref="WTLS"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="OperationalProtectionType">
      <xs:complexContent>
        <xs:restriction base="OperationalProtectionType">
          <xs:sequence>
            <xs:element ref="SecurityAudit"/>
            <xs:element ref="DeactivationCallCenter"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TechnicalProtectionBaseType">
      <xs:complexContent>
        <xs:restriction base="TechnicalProtectionBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PrivateKeyProtection"/>
              <xs:element ref="SecretKeyProtection"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrivateKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="PrivateKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
```

```
    <xs:complexType name="SecretKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="SecretKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyStorageType">
      <xs:complexContent>
        <xs:restriction base="KeyStorageType">
          <xs:attribute name="medium" use="required">
            <xs:simpleType>
              <xs:restriction base="mediumType">
                <xs:enumeration value="smartcard"/>
                <xs:enumeration value="MobileDevice"/>
                <xs:enumeration value="MobileAuthCard"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SecurityAuditType">
      <xs:complexContent>
        <xs:restriction base="SecurityAuditType">
          <xs:sequence>
            <xs:element ref="SwitchAudit"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="IdentificationType">
      <xs:complexContent>
        <xs:restriction base="IdentificationType">
          <xs:sequence>
            <xs:element ref="PhysicalVerification"/>
            <xs:element ref="WrittenConsent"/>
            <xs:element ref="GoverningAgreements"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="nym">
            <xs:simpleType>
              <xs:restriction base="nymType">
                <xs:enumeration value="anonymity"/>
                <xs:enumeration value="verinymity"/>
                <xs:enumeration value="pseudonymity"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.7 MobileTwoFactorContract

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

Reflects mobile contract customer registration procedures and a two-factor based authentication. For example, a digital signing device with tamper resistant memory for key storage, such as a GSM SIM, that requires explicit proof of user identity and intent, such as a PIN or biometric.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorCont
ract"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract
        Document identifier: saml-schema-authn-context-mobiletwofactor-reg-
2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"
minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
```

```
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="ZeroKnowledge"/>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
              <xs:element ref="ComplexAuthenticator"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ComplexAuthenticatorType">
      <xs:complexContent>
        <xs:restriction base="ComplexAuthenticatorType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
            </xs:choice>
            <xs:element ref="Password"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SSL"/>
              <xs:element ref="MobileNetworkNoEncryption"/>
              <xs:element ref="MobileNetworkRadioEncryption"/>
              <xs:element ref="MobileNetworkEndToEndEncryption"/>
              <xs:element ref="WTLS"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="OperationalProtectionType">
      <xs:complexContent>
        <xs:restriction base="OperationalProtectionType">
          <xs:sequence>
            <xs:element ref="SecurityAudit"/>
            <xs:element ref="DeactivationCallCenter"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TechnicalProtectionBaseType">
      <xs:complexContent>
        <xs:restriction base="TechnicalProtectionBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PrivateKeyProtection"/>
              <xs:element ref="SecretKeyProtection"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrivateKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="PrivateKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyActivation"/>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SecretKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="SecretKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyActivation"/>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyStorageType">
      <xs:complexContent>
        <xs:restriction base="KeyStorageType">
          <xs:attribute name="medium" use="required">
            <xs:simpleType>
              <xs:restriction base="mediumType">
                <xs:enumeration value="MobileDevice"/>
                <xs:enumeration value="MobileAuthCard"/>
                <xs:enumeration value="smartcard"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SecurityAuditType">
      <xs:complexContent>
        <xs:restriction base="SecurityAuditType">
          <xs:sequence>
            <xs:element ref="SwitchAudit"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="IdentificationType">
      <xs:complexContent>
        <xs:restriction base="IdentificationType">
          <xs:sequence>
            <xs:element ref="PhysicalVerification"/>
            <xs:element ref="WrittenConsent"/>
            <xs:element ref="GoverningAgreements"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="nym">
            <xs:simpleType>
              <xs:restriction base="nymType">
                <xs:enumeration value="anonymity"/>
                <xs:enumeration value="verinymity"/>
                <xs:enumeration value="pseudonymity"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

NOTE – The use of SSL is presented in Appendix IV.

12.3.4.8 Password

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:Password`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The Password class is applicable when a principal authenticates to an authentication authority through the presentation of a password over an unprotected HTTP session.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:Password"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:Password"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:Password
        Document identifier: saml-schema-authn-context-pword-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
```

```
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"
minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

Following is an example of an XML instance that conforms to the context class schema:

```
<AuthenticationContextDeclaration
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:Password">

  <AuthnMethod>
    <Authenticator>
      <AuthenticatorSequence>
        <RestrictedPassword>
          <Length min="4"/>
        </RestrictedPassword>
      </AuthenticatorSequence>
    </Authenticator>
  </AuthnMethod>

</AuthenticationContextDeclaration>
```

12.3.4.9 PasswordProtectedTransport

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The PasswordProtectedTransport class is applicable when a principal authenticates to an authentication authority through the presentation of a password over a protected session.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTr
ansport"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
        Document identifier: saml-schema-authn-context-ppt-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"
minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SSL"/>
              <xs:element ref="MobileNetworkRadioEncryption"/>
              <xs:element ref="MobileNetworkEndToEndEncryption"/>
              <xs:element ref="WTLS"/>
              <xs:element ref="IPSec"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

NOTE – The use of SSL is presented in Appendix IV.

12.3.4.10 PreviousSession

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The PreviousSession class is applicable when a principal had authenticated to an authentication authority at some point in the past using any authentication context supported by that authentication authority. Consequently, a subsequent authentication event that the authentication authority will assert to the relying party may be significantly separated in time from the principal's current resource access request.

The context for the previously authenticated session is explicitly not included in this context class because the user has not authenticated during this session, and so the mechanism that the user employed to authenticate in a previous session should not be used as part of a decision on whether to now allow access to a resource.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession
        Document identifier: saml-schema-authn-context-session-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"
minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="PreviousSession"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.11 Public key – X.509

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:X509`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The X509 context class indicates that the principal authenticated by means of a digital signature where the key was validated as part of an X.509 public key infrastructure.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:X509"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:X509"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:X509
        Document identifier: saml-schema-authn-context-x509-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="DigSig"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PublicKeyType">
      <xs:complexContent>
        <xs:restriction base="PublicKeyType">
          <xs:attribute name="keyValidation" type="xs:anyURI"
fixed="urn:oasis:names:tc:SAML:2.0:ac:classes:X509"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.12 Public key – PGP

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:PGP`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The PGP context class indicates that the principal authenticated by means of a digital signature where the key was validated as part of a PGP public key infrastructure.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:PGP"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:PGP"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:PGP
        Document identifier: saml-schema-authn-context-pgp-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="DigSig"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PublicKeyType">
      <xs:complexContent>
        <xs:restriction base="PublicKeyType">
          <xs:attribute name="keyValidation"
fixed="urn:oasis:names:tc:SAML:2.0:ac:classes:PGP"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.13 Public key – SPKI

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The SPKI context class indicates that the principal authenticated by means of a digital signature where the key was validated via an SPKI infrastructure.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI
        Document identifier: saml-schema-authn-context-spki-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="DigSig"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PublicKeyType">
      <xs:complexContent>
        <xs:restriction base="PublicKeyType">
          <xs:attribute name="keyValidation"
fixed="urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.14 Public key – XML digital signature

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:XMLDSig`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

This context class indicates that the principal authenticated by means of a digital signature according to the processing rules specified in W3C XML Signature.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:XMLDSig"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:XMLDSig"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:XMLDSig
        Document identifier: saml-schema-authn-context-xmldsig-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="DigSig"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PublicKeyType">
      <xs:complexContent>
        <xs:restriction base="PublicKeyType">
          <xs:attribute name="keyValidation" type="xs:anyURI"
fixed="urn:ietf:rfc:3075"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.15 Smartcard

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:Smartcard`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The Smartcard class is identified when a principal authenticates to an authentication authority using a smartcard.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:Smartcard"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:Smartcard"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">
```

```
    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:Smartcard
        Document identifier: saml-schema-authn-context-smartcard-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="Smartcard"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.16 SmartcardPKI

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The SmartcardPKI class is applicable when a principal authenticates to an authentication authority through a two-factor authentication mechanism using a smartcard with enclosed private key and a PIN.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI
        Document identifier: saml-schema-authn-context-smartcardpki-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TechnicalProtectionBaseType">
      <xs:complexContent>
        <xs:restriction base="TechnicalProtectionBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PrivateKeyProtection"/>
            </xs:choice>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
```

```
          <xs:sequence>
            <xs:element ref="Smartcard"/>
            <xs:element ref="ActivationPin"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrivateKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="PrivateKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyActivation"/>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyActivationType">
      <xs:complexContent>
        <xs:restriction base="KeyActivationType">
          <xs:sequence>
            <xs:element ref="ActivationPin"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyStorageType">
      <xs:complexContent>
        <xs:restriction base="KeyStorageType">
          <xs:attribute name="medium" use="required">
            <xs:simpleType>
              <xs:restriction base="mediumType">
                <xs:enumeration value="smartcard"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.17 SoftwarePKI

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:SoftwarePKI`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The SoftwarePKI class is applicable when a principal uses an X.509 certificate stored in software to authenticate to the authentication authority.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:SoftwarePKI"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:SoftwarePKI"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:SoftwarePKI
        Document identifier: saml-schema-authn-context-softwarepki-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TechnicalProtectionBaseType">
      <xs:complexContent>
        <xs:restriction base="TechnicalProtectionBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PrivateKeyProtection"/>
            </xs:choice>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="ActivationPin"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrivateKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="PrivateKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyActivation"/>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyActivationType">
      <xs:complexContent>
        <xs:restriction base="KeyActivationType">
          <xs:sequence>
            <xs:element ref="ActivationPin"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyStorageType">
      <xs:complexContent>
        <xs:restriction base="KeyStorageType">
          <xs:attribute name="medium" use="required">
            <xs:simpleType>
              <xs:restriction base="mediumType">
                <xs:enumeration value="memory"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>
</xs:schema>
```

12.3.4.18 Telephony

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:Telephony`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

This class is used to indicate that the principal authenticated via the provision of a fixed-line telephone number, transported via a telephony protocol such as ADSL.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:Telephony"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:Telephony"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:Telephony
        Document identifier: saml-schema-authn-context-telephony-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"
minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
```

```
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="SubscriberLineNumber"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PSTN"/>
              <xs:element ref="ISDN"/>
              <xs:element ref="ADSL"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.19 Telephony (nomadic)

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:NomadTelephony`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

Indicates that the principal is "roaming" (perhaps using a phone card) and authenticates via the means of the line number, a user suffix, and a password element.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:NomadTelephony"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:NomadTelephony"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:NomadTelephony
        Document identifier: saml-schema-authn-context-nomad-telephony-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"
minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="Password"/>
            <xs:element ref="SubscriberLineNumber"/>
            <xs:element ref="UserSuffix"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PSTN"/>
              <xs:element ref="ISDN"/>
              <xs:element ref="ADSL"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.20 Telephony (personalized)

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:PersonalTelephony`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

This class is used to indicate that the principal authenticated via the provision of a fixed-line telephone number and a user suffix, transported via a telephony protocol such as ADSL.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:PersonalizedTelephony"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:PersonalizedTelephony"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:PersonalizedTelephony
        Document identifier: saml-schema-authn-context-personal-telephony-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="SubscriberLineNumber"/>
            <xs:element ref="UserSuffix"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PSTN"/>
              <xs:element ref="ISDN"/>
              <xs:element ref="ADSL"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.21 Telephony (authenticated)

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:AuthenticatedTelephony`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

Indicates that the principal authenticated via the means of the line number, a user suffix, and a password element.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:AuthenticatedTeleph
ony"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:AuthenticatedTelephony"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:AuthenticatedTelephony
        Document identifier: saml-schema-authn-context-auth-telephony-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"
minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="Password"/>
            <xs:element ref="SubscriberLineNumber"/>
            <xs:element ref="UserSuffix"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PSTN"/>
              <xs:element ref="ISDN"/>
              <xs:element ref="ADSL"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
  </xs:redefine>

</xs:schema>
```

12.3.4.22 Secure remote password

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:SecureRemotePassword`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The SecureRemotePassword class is applicable when the authentication was performed by means of secure remote password as specified in IETF RFC 2945.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:SecureRemotePassword"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:SecureRemotePassword"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:SecureRemotePassword
        Document identifier: saml-schema-authn-context-srp-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="SharedSecretChallengeResponse"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SharedSecretChallengeResponseType">
      <xs:complexContent>
        <xs:restriction base="SharedSecretChallengeResponseType">
          <xs:attribute name="method" type="xs:anyURI" fixed="urn:ietf:rfc:2945"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

12.3.4.23 TLS certificate-based client authentication

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

This class indicates that the principal authenticated by means of a client certificate, secured with the TLS transport.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">
  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient
        Document identifier: saml-schema-authn-context-sslcert-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="DigSig"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PublicKeyType">
      <xs:complexContent>
        <xs:restriction base="PublicKeyType">
          <xs:attribute name="keyValidation" type="xs:anyURI"
fixed="urn:oasis:names:tc:SAML:2.0:ac:classes:X509"/>
```

```
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SSL"/>
              <xs:element ref="WTLS"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

NOTE – The use of SSL is presented in Appendix IV.

12.3.4.24 TimeSyncToken

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:TimeSyncToken`

This URI is also used as the target namespace in the corresponding authentication context class schema in Annex A.

The TimeSyncToken class is applicable when a principal authenticates through a time synchronization token.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:TimeSyncToken"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:TimeSyncToken"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:TimeSyncToken
        Document identifier: saml-schema-authn-context-timesync-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"
minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"
minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="Token"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TokenType">
      <xs:complexContent>
        <xs:restriction base="TokenType">
          <xs:sequence>
            <xs:element ref="TimeSyncToken"/>
            <xs:element ref="Extension" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TimeSyncTokenType">
      <xs:complexContent>
        <xs:restriction base="TimeSyncTokenType">
          <xs:attribute name="DeviceType" use="required">
            <xs:simpleType>
              <xs:restriction base="DeviceTypeType">
                <xs:enumeration value="hardware"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>

          <xs:attribute name="SeedLength" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:integer">
                <xs:minInclusive value="64"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>

          <xs:attribute name="DeviceInHand" use="required">
            <xs:simpleType>
              <xs:restriction base="booleanType">
                <xs:enumeration value="true"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
```

```
  </xs:redefine>

</xs:schema>
```

12.3.4.25 Unspecified

URI: `urn:oasis:names:tc:SAML:2.0:ac:classes:Unspecified`

The Unspecified class indicates that the authentication was performed by unspecified means.

13 Conformance requirements for SAML

This clause describes features that are mandatory and optional for implementations claiming conformance to SAML.

This Recommendation defines a number of named profiles. Each profile (other than attribute profiles) describes details of selected SAML message flows and can also be viewed as indivisible functionality that could be implemented by a software component. Implementation of a profile involves use of a binding for each message exchange included in the profile. A binding can be viewed as a specific implementation technique for achieving a message exchange.

This clause enumerates all of the different profiles defined in this Recommendation. For each profile, the relevant SAML V2.0 message flows are listed, and for each message flow the set of possible bindings is also described. The combination of profile, message exchange and a selected binding is termed a SAML V2.0 *feature*.

The clause also describes the conformance matrix for SAML V2.0. A number of different *operational modes* or roles are identified. The conformance matrix describes the feature set that must be implemented by each operational mode.

13.1 SAML profiles and possible implementations

Table 1 enumerates all of the profiles defined by the SAML profiles. For each profile, the message protocol flows found within the profile are also described. For each message flow, a list of relevant bindings is given in the final column.

Table 1/X.1141 – Possible implementations

Profile	Message flows	Binding
Web SSO	`<AuthnRequest>` from SP to IdP	HTTP Redirect
		HTTP POST
		HTTP Artifact
	IdP `<Response>` to SP	HTTP POST
		HTTP Artifact
Enhanced Client/Proxy SSO	ECP to SP, SP to ECP to IdP	PAOS
	IdP to ECP to SP, SP to ECP	PAOS
Identity Provider discovery	Cookie setter	HTTP
	Cookie getter	HTTP
Single Logout	`<LogoutRequest>`	HTTP Redirect
		HTTP POST
		HTTP Artifact
		SOAP
	`<LogoutResponse>`	HTTP Redirect
		HTTP POST
		HTTP Artifact
		SOAP
Name Identifier Management	`<ManageNameIDRequest>`	HTTP Redirect
		HTTP POST
		HTTP Artifact
		SOAP
	`<ManageNameIDResponse>`	HTTP Redirect
		SOAP
Artifact Resolution	`<ArtifactResolve>`, `<ArtifactResponse>`	SOAP

Table 1/X.1141 – Possible implementations

Profile	Message flows	Binding
Authentication Query	`<AuthnQuery>, <Response>`	SOAP
Attribute Query	`<AttributeQuery>, <Response>`	SOAP
Authorization Decision Query	`<AuthzDecisionQuery>, <Response>`	SOAP
Request for Assertion by Identifier	`<AssertionIDRequest>, <Response>`	SOAP
Name Identifier Mapping	`<NameIDMappingRequest>, <NameIDMappingResponse>`	SOAP
SAML URI binding	GET, HTTP Response	HTTP
UUID attribute profile		
DCE PAC attribute profile		
X.500 attribute profile		
XACML attribute profile		
Metadata	Consumption	
	Exchange	

13.2 Conformance

This clause describes the technical conformance requirements for SAML V2.0.

13.2.1 Operational modes

This Recommendation uses the phrase "operational mode" to describe a role that a software component can play in conforming to SAML. The operational modes are as follows:

- IdP – Identity provider
- IdP Lite – Identity provider lite
- SP – Service provider
- SP Lite – Service provider lite
- ECP – Enhanced client/proxy
- SAML Attribute authority
- SAML Authorization decision authority
- SAML Authentication authority
- SAML Requester

13.2.2 Feature matrix

The following matrices (see Table 2) identify unique sets of conformance requirements by means of a triple taken from Table 1 with the form: profile, message(s), binding. The message component is not always included when it is obvious from context.

Table 2/X.1141 – Feature matrix

Feature	IdP	IdP lite	SP	SP lite	ECP
Web SSO, `<AuthnRequest>`, HTTP redirect	Must	Must	Must	Must	N/A
Web SSO, `<Response>`, HTTP POST	Must	Must	Must	Must	N/A
Web SSO, `<Response>`, HTTP artifact	Must	Must	Must	Must	N/A
Artifact Resolution, SOAP	Must	Must	Must	Must	N/A
Enhanced Client/Proxy SSO, PAOS	Must	Must	Must	Must	Must
Name Identifier Management, HTTP redirect (IdP-initiated)	Must	Must not	Must	Must not	N/A
Name Identifier Management, SOAP (IdP-initiated)	Must	Must not	Optional	Must not	N/A
Name Identifier Management, HTTP redirect NOTE (informative) – PE11 (see OASIS PE:2006) suggests to append (SP-initiated)	Must	Must not	Must	Must not	N/A
Name Identifier Management, SOAP (SP-initiated)	Must	Must not	Optional	Must not	N/A
Single Logout (IdP-initiated) – HTTP redirect	Must	Must	Must	Must	N/A
Single Logout (IdP-initiated) – SOAP	Must	Optional	Must	Optional	N/A
Single Logout (SP-initiated) – HTTP redirect	Must	Must	Must	Must	N/A
Single Logout (SP-initiated) – SOAP	Must	Optional	Must	Optional	N/A
Identity Provider Discovery (cookie)	Must	Must	Optional	Optional	N/A

NOTE 1 (informative) – PE16 (see OASIS PE:2006) suggests to replace N/A with "Optional" in last row of last column of Table 2.

NOTE 2 (informative) – PE25 (see OASIS PE:2006) suggests to add the following to the end of Table 2 as below:

Feature	IdP	IdP Lite	SP	SP Lite	ECP
Metadata Structures	Optional	Optional	Optional	Optional	N/A
Metadata Interoperation	Optional	Optional	Optional	Optional	N/A

NOTE 3 (informative) – PE29 (see OASIS PE:2006) suggests to add the following to the end of Table 2 as below:

Feature	IdP	IdP Lite	SP	SP Lite	ECP
Request for Assertion Identifier	Optional	N/A	N/A	N/A	N/A
SAML URI binding	Optional	N/A	N/A	N/A	N/A

Table 3 summarizes operational modes that extend the IdP or SP modes defined above. These are to be understood as a combination of an IdP or SP mode from the table above with the corresponding extended feature set below.

Table 3/X.1141 – Extended IdP, SP

Feature	IdP extended	SP extended
Identity Provider proxy	Must	Must
Name identifier mapping, SOAP	Must	Must

Table 4 summarizes conformance requirements for SAML authorities and requesters.

Table 4/X.1141 – SAML authority and requester matrix

Feature	SAML authentication authority	SAML attribute authority	SAML authorization decision authority	SAML requester
Authentication Query, SOAP	Must	Optional	Optional	Optional
Attribute Query, SOAP	Optional	Must	Optional	Optional
Authorization Decision Query, SOAP	Optional	Optional	Must	Optional
Request for Assertion by Identifier, SOAP	Must	Must	Must	Optional
SAML URI Binding	Must	Must	Must	Optional

NOTE 4 (informative) – PE25 and PE42 (see OASIS PE:2006) suggest to modify Table 4 as follows:

Feature	SAML authentication authority	SAML attribute authority	SAML authorization decision authority	SAML requester
Authentication Query, SOAP	Must	N/A	N/A	Optional
Attribute Query, SOAP	N/A	MUST	N/A	Optional
Authorization Decision Query, SOAP	N/A	N/A	Must	Optional
Request for Assertion by Identifier, SOAP	Must	Must	Must	Optional
SAML URI Binding	Must	Must	Must	Optional
Metadata Structures	Optional	Optional	Optional	Optional
Metadata Interoperation	Optional	Optional	Optional	Optional

13.2.3 Implementation of SAML-defined identifiers

All relevant operational modes must implement the following SAML-defined identifiers:

- All Attribute Name Format identifiers defined in clause 8.
- All Name Identifier Format identifiers defined in clause 8.

Conforming SAML implementations must permit the use of all identifier constants (see 8.1 and 8.2) when producing and consuming SAML messages. SAML message producers must be able to create messages and SAML message consumers must be able to process messages with any of the constants defined in these clauses.

Persistent name identifiers and transient name identifiers define normative processing rules for the producer of such identifiers. All normative processing rules must be supported by conforming implementations. The remaining identifiers specify no normative processing rules. Hence, generation and consumption of these identifiers is meaningful only when the generating and consuming parties have externally-defined agreement on the semantic interpretation of the identifiers.

NOTE – In this context, "process" means that the implementation must successfully parse and handle the identifier without failing or returning an error. How the implementation deals with the identifier once it is processed at this level is out of scope for this Recommendation.

A SAML implementation may provide the facilities described above through direct implementation support for the identifiers or through the use of supported programming interfaces. Interfaces provided for this purpose must allow the SAML implementation to be programmatically extended to handle all identifiers that are not natively handled by the implementation.

13.2.4 Implementation of encrypted elements

All relevant operational modes must be able to process or generate the following encrypted elements in any context where they are required to process or generate the corresponding unencrypted elements, namely `<saml:NameID>`, `<saml:Assertion>`, or `<saml:Attribute>`:

- `<saml:EncryptedID>`
- `<saml:EncryptedAssertion>`
- `<saml:EncryptedAttribute>`

13.2.5 Security models for SOAP and URI bindings

The following security models are mandatory to implement for all profiles implemented using the SOAP binding as well as for the SAML URI binding. SAML authorities and requesters must implement the following authentication methods:

- No client or server authentication.
- HTTP basic authentication with and without TLS 1.0. The SAML requester must preemptively send the authorization header with the initial request.
- HTTP over TLS 1.0 server authentication with server-side certificate.
- HTTP over TLS 1.0 mutual authentication with both server-side and a client-side certificate.

If a SAML authority uses TLS 1.0, it must use a server-side certificate.

NOTE 1 (informative) – PE25 (see OASIS PE:2006) suggests to add a new subclause on Metadata Structures as below:

Implementations claiming conformance to SAML may declare each operational mode's conformance to SAML Metadata through election of the Metadata Structures option. With respect to each operational mode, such conformance entails the following:

Implementing SAML metadata according to the extensible SAML Metadata format in all cases where an interoperating peer has the option, as stated in SAML specifications, of depending on the existence of SAML metadata. Electing the metadata structures option has the effect of requiring such metadata be available to the interoperating peer. The Metadata Interoperation feature, described below, provides a means of satisfying this requirement.

Referencing, consuming, and adherence to the SAML metadata, according to, of an interoperating peer when the known metadata relevant to that peer and the particular operation, and the current exchange, has expired or is no longer valid in cache, provided the metadata is available and is not prohibited by policy or the particular operation and that specific exchange.

NOTE 2 (informative) – PE25 (see OASIS PE:2006) suggests to add a new subclause on metadata interoperation as below:

Election of the metadata interoperation option requires the implementation offer, in addition to any other mechanism, the well-known location publication and resolution mechanism described in SAML metadata in clause 9.

13.3 XML digital signature and XML encryption

SAML V2.0 uses XML Signature to implement XML signing and encryption functionality for integrity, and source authentication. SAML V2.0 uses XML Encryption to implement confidentiality, including encrypted identifiers, encrypted assertions, and encrypted attributes.

13.3.1 XML Signature algorithms

W3C XML Signature, 6.1, mandates use of the following:

- Digest: SHA-1;
- MAC: HMAC-SHA1;
- XML Canonicalization: CanonicalXML (without comments);
- Transform: Enveloped Signature.

Therefore, they must be implemented by compliant SAML V2.0 implementations.

In addition, to enable interoperability, the following must be implemented by compliant SAML V2.0 implementations:

- Signature: RSAwithSHA1 (recommended in W3C Signature, needed for interoperability).

Although XML Signature mandates the DSAwithSHA1 signature algorithm, it is not required by SAML V2.0, but is recommended.

NOTE – NIST (National Institute of Standards and Technology) now encourages the use of SHA-256 (Secure Hash Algorithm with 256-bit encoded keys) instead of SHA-1.

13.3.2 XML Encryption algorithms

- W3C XML Encryption, 5.2.1 and 5.2.2 mandate use of the following algorithms: Block encryption: Triple DES, AES-128, AES-256.
- Key Transport: RSA-v1.5, RSA-OAEP.

Therefore, the above algorithms must be implemented by compliant SAML V2.0 implementations.

13.4 Use of TLS 1.0

In any SAML V2.0 use of TLS 1.0, servers must authenticate to clients using an X.509 v3 certificate. The client must establish server identity based on contents of the certificate (typically through examination of the certificate's subject DN field).

13.4.1 SAML SOAP and URI binding

TLS-capable implementations must implement the TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher suite and may implement the TLS_RSA_AES_128_CBC_SHA cipher suite.

FIPS TLS-capable implementations must implement the corresponding TLS_RSA_FIPS_WITH_3DES_EDE_CBC_SHA cipher suite and may implement the corresponding TLS_RSA_FIPS_AES_128_CBC_SHA cipher suite.

13.4.2 Web SSO profiles of SAML

TLS-capable implementations must implement the TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher suite (see IETF RFC 2246).

Annex A

SAML schemas

This annex forms an integral part of this Recommendation. It provides a listing of required SAML Schemas.

A.1 SAML Schema Assertion

This is a listing of the SAML Schema Assertion.

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
20020212/xmldsig-core-schema.xsd"/>
    <import namespace="http://www.w3.org/2001/04/xmlenc#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-
schema.xsd"/>
    <annotation>
        <documentation>
            Document identifier: saml-schema-assertion-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
            V1.0 (November, 2002):
              Initial Standard Schema.
            V1.1 (September, 2003):
              Updates within the same V1.0 namespace.
            V2.0 (March, 2005):
              New assertion schema for SAML V2.0 namespace.
        </documentation>
    </annotation>
    <attributeGroup name="IDNameQualifiers">
        <attribute name="NameQualifier" type="string" use="optional"/>
        <attribute name="SPNameQualifier" type="string" use="optional"/>
    </attributeGroup>
    <element name="BaseID" type="saml:BaseIDAbstractType"/>
    <complexType name="BaseIDAbstractType" abstract="true">
        <attributeGroup ref="saml:IDNameQualifiers"/>
    </complexType>
    <element name="NameID" type="saml:NameIDType"/>
    <complexType name="NameIDType">
```

```
        <simpleContent>
            <extension base="string">
                <attributeGroup ref="saml:IDNameQualifiers"/>
                <attribute name="Format" type="anyURI" use="optional"/>
                <attribute name="SPProvidedID" type="string" use="optional"/>
            </extension>
        </simpleContent>
    </complexType>
    <complexType name="EncryptedElementType">
        <sequence>
            <element ref="xenc:EncryptedData"/>
            <element ref="xenc:EncryptedKey" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
    </complexType>
    <element name="EncryptedID" type="saml:EncryptedElementType"/>
    <element name="Issuer" type="saml:NameIDType"/>
    <element name="AssertionIDRef" type="NCName"/>
    <element name="AssertionURIRef" type="anyURI"/>
    <element name="Assertion" type="saml:AssertionType"/>
    <complexType name="AssertionType">
        <sequence>
            <element ref="saml:Issuer"/>
            <element ref="ds:Signature" minOccurs="0"/>
            <element ref="saml:Subject" minOccurs="0"/>
            <element ref="saml:Conditions" minOccurs="0"/>
            <element ref="saml:Advice" minOccurs="0"/>
            <choice minOccurs="0" maxOccurs="unbounded">
                <element ref="saml:Statement"/>
                <element ref="saml:AuthnStatement"/>
                <element ref="saml:AuthzDecisionStatement"/>
                <element ref="saml:AttributeStatement"/>
            </choice>
        </sequence>
        <attribute name="Version" type="string" use="required"/>
        <attribute name="ID" type="ID" use="required"/>
        <attribute name="IssueInstant" type="dateTime" use="required"/>
    </complexType>
    <element name="Subject" type="saml:SubjectType"/>
    <complexType name="SubjectType">
        <choice>
            <sequence>
                <choice>
                    <element ref="saml:BaseID"/>
                    <element ref="saml:NameID"/>
                    <element ref="saml:EncryptedID"/>
                </choice>
                <element ref="saml:SubjectConfirmation" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
            <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
        </choice>
    </complexType>
    <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
    <complexType name="SubjectConfirmationType">
        <sequence>
            <choice minOccurs="0">
                <element ref="saml:BaseID"/>
                <element ref="saml:NameID"/>
                <element ref="saml:EncryptedID"/>
            </choice>
            <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
        </sequence>
        <attribute name="Method" type="anyURI" use="required"/>
    </complexType>
    <element name="SubjectConfirmationData"
type="saml:SubjectConfirmationDataType"/>
    <complexType name="SubjectConfirmationDataType" mixed="true">
        <complexContent>
            <restriction base="anyType">
                <sequence>
```

```
                    <any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
                <attribute name="NotBefore" type="dateTime" use="optional"/>
                <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
                <attribute name="Recipient" type="anyURI" use="optional"/>
                <attribute name="InResponseTo" type="NCName" use="optional"/>
                <attribute name="Address" type="string" use="optional"/>
                <anyAttribute namespace="##other" processContents="lax"/>
            </restriction>
        </complexContent>
    </complexType>
    <complexType name="KeyInfoConfirmationDataType" mixed="false">
        <complexContent>
            <restriction base="saml:SubjectConfirmationDataType">
                <sequence>
                    <element ref="ds:KeyInfo" maxOccurs="unbounded"/>
                </sequence>
            </restriction>
        </complexContent>
    </complexType>
    <element name="Conditions" type="saml:ConditionsType"/>
    <complexType name="ConditionsType">
        <choice minOccurs="0" maxOccurs="unbounded">
            <element ref="saml:Condition"/>
            <element ref="saml:AudienceRestriction"/>
            <element ref="saml:OneTimeUse"/>
            <element ref="saml:ProxyRestriction"/>
        </choice>
        <attribute name="NotBefore" type="dateTime" use="optional"/>
        <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
    </complexType>
    <element name="Condition" type="saml:ConditionAbstractType"/>
    <complexType name="ConditionAbstractType" abstract="true"/>
    <element name="AudienceRestriction" type="saml:AudienceRestrictionType"/>
    <complexType name="AudienceRestrictionType">
        <complexContent>
            <extension base="saml:ConditionAbstractType">
                <sequence>
                    <element ref="saml:Audience" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="Audience" type="anyURI"/>
    <element name="OneTimeUse" type="saml:OneTimeUseType" />
    <complexType name="OneTimeUseType">
        <complexContent>
            <extension base="saml:ConditionAbstractType"/>
        </complexContent>
    </complexType>
    <element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
    <complexType name="ProxyRestrictionType">
    <complexContent>
        <extension base="saml:ConditionAbstractType">
            <sequence>
                <element ref="saml:Audience" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
            <attribute name="Count" type="nonNegativeInteger" use="optional"/>
        </extension>
   </complexContent>
    </complexType>
    <element name="Advice" type="saml:AdviceType"/>
    <complexType name="AdviceType">
        <choice minOccurs="0" maxOccurs="unbounded">
            <element ref="saml:AssertionIDRef"/>
            <element ref="saml:AssertionURIRef"/>
            <element ref="saml:Assertion"/>
            <element ref="saml:EncryptedAssertion"/>
            <any namespace="##other" processContents="lax"/>
        </choice>
    </complexType>
    <element name="EncryptedAssertion" type="saml:EncryptedElementType"/>
    <element name="Statement" type="saml:StatementAbstractType"/>
    <complexType name="StatementAbstractType" abstract="true"/>
    <element name="AuthnStatement" type="saml:AuthnStatementType"/>
    <complexType name="AuthnStatementType">
        <complexContent>
            <extension base="saml:StatementAbstractType">
                <sequence>
                    <element ref="saml:SubjectLocality" minOccurs="0"/>
                    <element ref="saml:AuthnContext"/>
                </sequence>
                <attribute name="AuthnInstant" type="dateTime" use="required"/>
                <attribute name="SessionIndex" type="string" use="optional"/>
                <attribute name="SessionNotOnOrAfter" type="dateTime"
use="optional"/>
            </extension>
        </complexContent>
    </complexType>
    <element name="SubjectLocality" type="saml:SubjectLocalityType"/>
    <complexType name="SubjectLocalityType">
        <attribute name="Address" type="string" use="optional"/>
        <attribute name="DNSName" type="string" use="optional"/>
    </complexType>
    <element name="AuthnContext" type="saml:AuthnContextType"/>
    <complexType name="AuthnContextType">
        <sequence>
            <choice>
                <sequence>
                    <element ref="saml:AuthnContextClassRef"/>
                    <choice minOccurs="0">
                        <element ref="saml:AuthnContextDecl"/>
                        <element ref="saml:AuthnContextDeclRef"/>
                    </choice>
                </sequence>
                <choice>
                    <element ref="saml:AuthnContextDecl"/>
                    <element ref="saml:AuthnContextDeclRef"/>
                </choice>
            </choice>
            <element ref="saml:AuthenticatingAuthority" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
    </complexType>
    <element name="AuthnContextClassRef" type="anyURI"/>
    <element name="AuthnContextDeclRef" type="anyURI"/>
    <element name="AuthnContextDecl" type="anyType"/>
    <element name="AuthenticatingAuthority" type="anyURI"/>
    <element name="AuthzDecisionStatement"
type="saml:AuthzDecisionStatementType"/>
    <complexType name="AuthzDecisionStatementType">
        <complexContent>
            <extension base="saml:StatementAbstractType">
                <sequence>
                    <element ref="saml:Action" maxOccurs="unbounded"/>
                    <element ref="saml:Evidence" minOccurs="0"/>
                </sequence>
                <attribute name="Resource" type="anyURI" use="required"/>
                <attribute name="Decision" type="saml:DecisionType"
use="required"/>
            </extension>
        </complexContent>
    </complexType>
    <simpleType name="DecisionType">
        <restriction base="string">
            <enumeration value="Permit"/>
            <enumeration value="Deny"/>
            <enumeration value="Indeterminate"/>
        </restriction>
    </simpleType>
    <element name="Action" type="saml:ActionType"/>
    <complexType name="ActionType">
        <simpleContent>
            <extension base="string">
                <attribute name="Namespace" type="anyURI" use="required"/>
            </extension>
        </simpleContent>
    </complexType>
    <element name="Evidence" type="saml:EvidenceType"/>
    <complexType name="EvidenceType">
        <choice maxOccurs="unbounded">
            <element ref="saml:AssertionIDRef"/>
            <element ref="saml:AssertionURIRef"/>
            <element ref="saml:Assertion"/>
            <element ref="saml:EncryptedAssertion"/>
        </choice>
    </complexType>
    <element name="AttributeStatement" type="saml:AttributeStatementType"/>
    <complexType name="AttributeStatementType">
        <complexContent>
            <extension base="saml:StatementAbstractType">
                <choice maxOccurs="unbounded">
                    <element ref="saml:Attribute"/>
                    <element ref="saml:EncryptedAttribute"/>
                </choice>
            </extension>
        </complexContent>
    </complexType>
    <element name="Attribute" type="saml:AttributeType"/>
    <complexType name="AttributeType">
        <sequence>
            <element ref="saml:AttributeValue" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
        <attribute name="Name" type="string" use="required"/>
        <attribute name="NameFormat" type="anyURI" use="optional"/>
        <attribute name="FriendlyName" type="string" use="optional"/>
        <anyAttribute namespace="##other" processContents="lax"/>
    </complexType>
    <element name="AttributeValue" type="anyType" nillable="true"/>
    <element name="EncryptedAttribute" type="saml:EncryptedElementType"/>
</schema>
```

A.2 SAML Schema Authentication Context

This is the SAML Authentication Context Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="urn:oasis:names:tc:SAML:2.0:ac"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac"
  blockDefault="substitution"
  version="2.0">

  <xs:annotation>
    <xs:documentation>
      Document identifier: saml-schema-authn-context-2.0
      Location: http://docs.oasis-open.org/security/saml/v2.0/
      Revision history:
        V2.0 (March, 2005):
          New core authentication context schema for SAML V2.0.
          This is just an include of all types from the Shema
          referred to in the include statement below.
    </xs:documentation>
  </xs:annotation>

  <xs:include schemaLocation="saml-schema-authn-context-types-2.0.xsd"/>
```

```
</xs:schema>
```

A.3 SAML Schema Authentication Context AuthenticatedTelephony

This is the telephony related SAML Authentication Context Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:AuthenticatedTelephony"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:AuthenticatedTelephony"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:AuthenticatedTelephony
        Document identifier: saml-schema-authn-context-auth-telephony-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="Password"/>
            <xs:element ref="SubscriberLineNumber"/>
            <xs:element ref="UserSuffix"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PSTN"/>
              <xs:element ref="ISDN"/>
              <xs:element ref="ADSL"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.4 SAML Schema Authentication Context IP

This listing provides IP specific SAML Authentication Context Schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
  targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol
        Document identifier: saml-schema-authn-context-ip-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="IPAddress"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.5 SAML Schema Authentication Context IPPWord

This listing provides the Internet protocol password (IPPWord) SAML Authentication Context Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocolPassword"
  xmlns:ac="urn:oasis:names:tc:SAML:2.0:ac"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocolPassword"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocolPassword
        Document identifier: saml-schema-authn-context-ippword-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
```

```
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="Password"/>
            <xs:element ref="IPAddress"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.6 SAML Schema Authentication Context Kerberos

This listing provides SAML Kerberos Authentication Schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos
        Document identifier: saml-schema-authn-context-kerberos-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
```

```
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="SharedSecretChallengeResponse"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SharedSecretChallengeResponseType">
      <xs:complexContent>
        <xs:restriction base="SharedSecretChallengeResponseType">
          <xs:attribute name="method" type="xs:anyURI"
fixed="urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>
</xs:schema>
```

A.7 SAML Schema Authentication Context MobileOneFactor-reg

This listing contains the SAML context class schema for registered MobileOneFactorContract.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorContract"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorContract"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorContract
        Document identifier: saml-schema-authn-context-mobileonefactor-reg-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="ZeroKnowledge"/>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SSL"/>
              <xs:element ref="MobileNetworkNoEncryption"/>
              <xs:element ref="MobileNetworkRadioEncryption"/>
              <xs:element ref="MobileNetworkEndToEndEncryption"/>
              <xs:element ref="WTLS"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="OperationalProtectionType">
      <xs:complexContent>
        <xs:restriction base="OperationalProtectionType">
          <xs:sequence>
```

```
          <xs:element ref="SecurityAudit"/>
          <xs:element ref="DeactivationCallCenter"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="TechnicalProtectionBaseType">
    <xs:complexContent>
      <xs:restriction base="TechnicalProtectionBaseType">
        <xs:sequence>
          <xs:choice>
            <xs:element ref="PrivateKeyProtection"/>
            <xs:element ref="SecretKeyProtection"/>
          </xs:choice>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="PrivateKeyProtectionType">
    <xs:complexContent>
      <xs:restriction base="PrivateKeyProtectionType">
        <xs:sequence>
          <xs:element ref="KeyStorage"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="SecretKeyProtectionType">
    <xs:complexContent>
      <xs:restriction base="SecretKeyProtectionType">
        <xs:sequence>
          <xs:element ref="KeyStorage"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="KeyStorageType">
    <xs:complexContent>
      <xs:restriction base="KeyStorageType">
        <xs:attribute name="medium" use="required">
          <xs:simpleType>
            <xs:restriction base="mediumType">
              <xs:enumeration value="smartcard"/>
              <xs:enumeration value="MobileDevice"/>
              <xs:enumeration value="MobileAuthCard"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="SecurityAuditType">
    <xs:complexContent>
      <xs:restriction base="SecurityAuditType">
        <xs:sequence>
          <xs:element ref="SwitchAudit"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
```

```
    <xs:complexType name="IdentificationType">
      <xs:complexContent>
        <xs:restriction base="IdentificationType">
          <xs:sequence>
            <xs:element ref="PhysicalVerification"/>
            <xs:element ref="WrittenConsent"/>
            <xs:element ref="GoverningAgreements"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="nym">
            <xs:simpleType>
              <xs:restriction base="nymType">
                <xs:enumeration value="anonymity"/>
                <xs:enumeration value="verinymity"/>
                <xs:enumeration value="pseudonymity"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.8 SAML Schema Authentication Context MobileOneFactor-unreg

This listing contains the SAML context class schema for un-registered MobileOneFactorContract.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorUnregister
ed"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorUnregistered"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorUnregistered
        Document identifier: saml-schema-authn-context-mobileonefactor-unreg-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="ZeroKnowledge"/>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SSL"/>
              <xs:element ref="MobileNetworkNoEncryption"/>
              <xs:element ref="MobileNetworkRadioEncryption"/>
              <xs:element ref="MobileNetworkEndToEndEncryption"/>
              <xs:element ref="WTLS"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="OperationalProtectionType">
      <xs:complexContent>
        <xs:restriction base="OperationalProtectionType">
          <xs:sequence>
            <xs:element ref="SecurityAudit"/>
            <xs:element ref="DeactivationCallCenter"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TechnicalProtectionBaseType">
      <xs:complexContent>
        <xs:restriction base="TechnicalProtectionBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PrivateKeyProtection"/>
              <xs:element ref="SecretKeyProtection"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrivateKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="PrivateKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SecretKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="SecretKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyStorageType">
      <xs:complexContent>
        <xs:restriction base="KeyStorageType">
          <xs:attribute name="medium" use="required">
            <xs:simpleType>
              <xs:restriction base="mediumType">
                <xs:enumeration value="MobileDevice"/>
                <xs:enumeration value="MobileAuthCard"/>
                <xs:enumeration value="smartcard"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SecurityAuditType">
      <xs:complexContent>
        <xs:restriction base="SecurityAuditType">
          <xs:sequence>
            <xs:element ref="SwitchAudit"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="IdentificationType">
      <xs:complexContent>
        <xs:restriction base="IdentificationType">
          <xs:sequence>
            <xs:element ref="GoverningAgreements"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="nym">
            <xs:simpleType>
              <xs:restriction base="nymType">
                <xs:enumeration value="anonymity"/>
                <xs:enumeration value="pseudonymity"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.9 SAML Schema Authentication Context MobileTwoFactor-reg

This listing contains the SAML context class schema for registered MobileTwoFactorContract.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract
        Document identifier: saml-schema-authn-context-mobiletwofactor-reg-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="ZeroKnowledge"/>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
              <xs:element ref="ComplexAuthenticator"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ComplexAuthenticatorType">
      <xs:complexContent>
        <xs:restriction base="ComplexAuthenticatorType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
            </xs:choice>
            <xs:element ref="Password"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SSL"/>
              <xs:element ref="MobileNetworkNoEncryption"/>
              <xs:element ref="MobileNetworkRadioEncryption"/>
              <xs:element ref="MobileNetworkEndToEndEncryption"/>
              <xs:element ref="WTLS"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="OperationalProtectionType">
      <xs:complexContent>
        <xs:restriction base="OperationalProtectionType">
          <xs:sequence>
            <xs:element ref="SecurityAudit"/>
            <xs:element ref="DeactivationCallCenter"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TechnicalProtectionBaseType">
      <xs:complexContent>
        <xs:restriction base="TechnicalProtectionBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PrivateKeyProtection"/>
              <xs:element ref="SecretKeyProtection"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrivateKeyProtectionType">
```

```
    <xs:complexContent>
      <xs:restriction base="PrivateKeyProtectionType">
        <xs:sequence>
          <xs:element ref="KeyActivation"/>
          <xs:element ref="KeyStorage"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="SecretKeyProtectionType">
    <xs:complexContent>
      <xs:restriction base="SecretKeyProtectionType">
        <xs:sequence>
          <xs:element ref="KeyActivation"/>
          <xs:element ref="KeyStorage"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="KeyStorageType">
    <xs:complexContent>
      <xs:restriction base="KeyStorageType">
        <xs:attribute name="medium" use="required">
          <xs:simpleType>
            <xs:restriction base="mediumType">
              <xs:enumeration value="MobileDevice"/>
              <xs:enumeration value="MobileAuthCard"/>
              <xs:enumeration value="smartcard"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="SecurityAuditType">
    <xs:complexContent>
      <xs:restriction base="SecurityAuditType">
        <xs:sequence>
          <xs:element ref="SwitchAudit"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="IdentificationType">
    <xs:complexContent>
      <xs:restriction base="IdentificationType">
        <xs:sequence>
          <xs:element ref="PhysicalVerification"/>
          <xs:element ref="WrittenConsent"/>
          <xs:element ref="GoverningAgreements"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="nym">
          <xs:simpleType>
            <xs:restriction base="nymType">
              <xs:enumeration value="anonymity"/>
              <xs:enumeration value="verinymity"/>
              <xs:enumeration value="pseudonymity"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
  </xs:redefine>

</xs:schema>
```

A.10 SAML Schema Authentication Context MobileTwoFactor-unreg

This listing contains the SAML context class schema for MobileTwoFactorUnregistered.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorUnregister
ed"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorUnregistered"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorUnregistered
        Document identifier: saml-schema-authn-context-mobiletwofactor-unreg-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="ZeroKnowledge"/>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
              <xs:element ref="ComplexAuthenticator"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ComplexAuthenticatorType">
      <xs:complexContent>
        <xs:restriction base="ComplexAuthenticatorType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SharedSecretChallengeResponse"/>
              <xs:element ref="SharedSecretDynamicPlaintext"/>
            </xs:choice>
            <xs:element ref="Password"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SSL"/>
              <xs:element ref="MobileNetworkNoEncryption"/>
              <xs:element ref="MobileNetworkRadioEncryption"/>
              <xs:element ref="MobileNetworkEndToEndEncryption"/>
              <xs:element ref="WTLS"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="OperationalProtectionType">
      <xs:complexContent>
        <xs:restriction base="OperationalProtectionType">
          <xs:sequence>
            <xs:element ref="SecurityAudit"/>
            <xs:element ref="DeactivationCallCenter"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TechnicalProtectionBaseType">
      <xs:complexContent>
        <xs:restriction base="TechnicalProtectionBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PrivateKeyProtection"/>
              <xs:element ref="SecretKeyProtection"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrivateKeyProtectionType">
      <xs:complexContent>
```

```
      <xs:restriction base="PrivateKeyProtectionType">
        <xs:sequence>
          <xs:element ref="KeyActivation"/>
          <xs:element ref="KeyStorage"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="SecretKeyProtectionType">
    <xs:complexContent>
      <xs:restriction base="SecretKeyProtectionType">
        <xs:sequence>
          <xs:element ref="KeyActivation"/>
          <xs:element ref="KeyStorage"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="KeyStorageType">
    <xs:complexContent>
      <xs:restriction base="KeyStorageType">
        <xs:attribute name="medium" use="required">
          <xs:simpleType>
            <xs:restriction base="mediumType">
              <xs:enumeration value="MobileDevice"/>
              <xs:enumeration value="MobileAuthCard"/>
              <xs:enumeration value="smartcard"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="SecurityAuditType">
    <xs:complexContent>
      <xs:restriction base="SecurityAuditType">
        <xs:sequence>
          <xs:element ref="SwitchAudit"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="IdentificationType">
    <xs:complexContent>
      <xs:restriction base="IdentificationType">
        <xs:sequence>
          <xs:element ref="GoverningAgreements"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="nym">
          <xs:simpleType>
            <xs:restriction base="nymType">
              <xs:enumeration value="anonymity"/>
              <xs:enumeration value="pseudonymity"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

</xs:redefine>
```

```
</xs:schema>
```

A.11 SAML Schema Authentication Context NomadTelephony

This listing contains SAML NomadTelephony Authentication Schema. Nomad telephony indicates that the principal is "roaming" (perhaps using a phone card) and authenticates via the means of the line number, a user suffix, and a password element.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:NomadTelephony"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:NomadTelephony"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:NomadTelephony
        Document identifier: saml-schema-authn-context-nomad-telephony-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="Password"/>
            <xs:element ref="SubscriberLineNumber"/>
            <xs:element ref="UserSuffix"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PSTN"/>
              <xs:element ref="ISDN"/>
              <xs:element ref="ADSL"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.12 SAML Schema Authentication Context PersonalizedTelephony

This listing provides SAML Authentication Schema for personal telephony.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:PersonalizedTelephony"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:PersonalizedTelephony"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:PersonalizedTelephony
        Document identifier: saml-schema-authn-context-personal-telephony-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
```

```
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="SubscriberLineNumber"/>
            <xs:element ref="UserSuffix"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PSTN"/>
              <xs:element ref="ISDN"/>
              <xs:element ref="ADSL"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.13 SAML Schema Authentication Context PGP

This listing provides SAML Authentication Schema for PGP.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:PGP"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:PGP"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:PGP
        Document identifier: saml-schema-authn-context-pgp-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
```

```
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="DigSig"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PublicKeyType">
      <xs:complexContent>
        <xs:restriction base="PublicKeyType">
          <xs:attribute name="keyValidation"
fixed="urn:oasis:names:tc:SAML:2.0:ac:classes:PGP"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.14 SAML Schema Authentication Context PPT

This listing contains the SAML Authentication Schema for password protected transport.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTranspor
t"
```

```
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
        Document identifier: saml-schema-authn-context-ppt-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SSL"/>
              <xs:element ref="MobileNetworkRadioEncryption"/>
              <xs:element ref="MobileNetworkEndToEndEncryption"/>
              <xs:element ref="WTLS"/>
              <xs:element ref="IPSec"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.15 SAML Schema Authentication Context Password

This listing contains SAML Authentication Context Password Schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:Password"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:Password"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:Password
        Document identifier: saml-schema-authn-context-pword-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.16 SAML Schema Authentication Context PreviousSession

This listing contains SAML Authentication Context Schema for PreviousSession. The PreviousSession class is applicable when a principal had authenticated to an authentication authority at some point in the past using any authentication context supported by that authentication authority.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession
        Document identifier: saml-schema-authn-context-session-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
```

```
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="PreviousSession"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.17 SAML Schema Authentication Context Smartcard

This is the smartcard SAML Authentication Context Schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:Smartcard"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:Smartcard"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:Smartcard
        Document identifier: saml-schema-authn-context-smartcard-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
```

```
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="Smartcard"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.18 SAML Schema Authentication Context SmartardPKI

This is the PKI smartcard SAML Authentication Context Schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI
        Document identifier: saml-schema-authn-context-smartcardpki-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TechnicalProtectionBaseType">
      <xs:complexContent>
        <xs:restriction base="TechnicalProtectionBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PrivateKeyProtection"/>
            </xs:choice>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="Smartcard"/>
            <xs:element ref="ActivationPin"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrivateKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="PrivateKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyActivation"/>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyActivationType">
      <xs:complexContent>
        <xs:restriction base="KeyActivationType">
          <xs:sequence>
            <xs:element ref="ActivationPin"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyStorageType">
      <xs:complexContent>
        <xs:restriction base="KeyStorageType">
          <xs:attribute name="medium" use="required">
            <xs:simpleType>
              <xs:restriction base="mediumType">
                <xs:enumeration value="smartcard"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.19 SAML Schema Authentication Context SoftwarePKI

This is the software PKI SAML Authentication Context Schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:SoftwarePKI"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:SoftwarePKI"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:SoftwarePKI
        Document identifier: saml-schema-authn-context-softwarepki-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TechnicalProtectionBaseType">
      <xs:complexContent>
        <xs:restriction base="TechnicalProtectionBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PrivateKeyProtection"/>
            </xs:choice>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="ActivationPin"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="DigSig"/>
              <xs:element ref="AsymmetricDecryption"/>
              <xs:element ref="AsymmetricKeyAgreement"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrivateKeyProtectionType">
      <xs:complexContent>
        <xs:restriction base="PrivateKeyProtectionType">
          <xs:sequence>
            <xs:element ref="KeyActivation"/>
            <xs:element ref="KeyStorage"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyActivationType">
      <xs:complexContent>
        <xs:restriction base="KeyActivationType">
          <xs:sequence>
            <xs:element ref="ActivationPin"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="KeyStorageType">
      <xs:complexContent>
        <xs:restriction base="KeyStorageType">
          <xs:attribute name="medium" use="required">
            <xs:simpleType>
              <xs:restriction base="mediumType">
                <xs:enumeration value="memory"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.20 SAML Schema Authentication Context SPKI

This is the public key SAML Authentication Context Schema. The SPKI context class indicates that the principal authenticated by means of a digital signature where the key was validated via an SPKI Infrastructure.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI
        Document identifier: saml-schema-authn-context-spki-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
```

```
          <xs:sequence>
            <xs:element ref="DigSig"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PublicKeyType">
      <xs:complexContent>
        <xs:restriction base="PublicKeyType">
          <xs:attribute name="keyValidation"
fixed="urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.21 SAML Schema Authentication Context SRP

This is the SRP [see IETF RFC 2945] SAML Authentication Context Schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:SecureRemotePassword"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:SecureRemotePassword"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier:
urn:oasis:names:tc:SAML:2.0:ac:classes:SecureRemotePassword
        Document identifier: saml-schema-authn-context-srp-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="SharedSecretChallengeResponse"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="SharedSecretChallengeResponseType">
      <xs:complexContent>
        <xs:restriction base="SharedSecretChallengeResponseType">
          <xs:attribute name="method" type="xs:anyURI"
fixed="urn:ietf:rfc:2945"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.22 SAML Schema Authentication Context Telephony

This is the telephony SAML Authentication Context Schema. This is used when the principal is authenticated via the provision of a fixed-line telephone number and transported via a telephony protocol.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:Telephony"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:Telephony"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:Telephony
        Document identifier: saml-schema-authn-context-telephony-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
```

```
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="SubscriberLineNumber"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
          <xs:sequence>
            <xs:choice>
              <xs:element ref="PSTN"/>
              <xs:element ref="ISDN"/>
              <xs:element ref="ADSL"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.23 SAML Schema Authentication Context TimeSync

This is the TimeSynchToken SAML Authentication Context Schema. TimeSyncToken is applicable when a principal authenticates through a time synchronization token.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:TimeSyncToken"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:TimeSyncToken"
  finalDefault="extension"
  blockDefault="substitution"
```

```
version="2.0">

<xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

  <xs:annotation>
    <xs:documentation>
      Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:TimeSyncToken
      Document identifier: saml-schema-authn-context-timesync-2.0
      Location: http://docs.oasis-open.org/security/saml/v2.0/
      Revision history:
        V2.0 (March, 2005):
          New authentication_u99 context class schema for SAML V2.0.
    </xs:documentation>
  </xs:annotation>

  <xs:complexType name="AuthnContextDeclarationBaseType">
    <xs:complexContent>
      <xs:restriction base="AuthnContextDeclarationBaseType">
        <xs:sequence>
          <xs:element ref="Identification" minOccurs="0"/>
          <xs:element ref="TechnicalProtection" minOccurs="0"/>
          <xs:element ref="OperationalProtection" minOccurs="0"/>
          <xs:element ref="AuthnMethod"/>
          <xs:element ref="GoverningAgreements" minOccurs="0"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="ID" type="xs:ID" use="optional"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="AuthnMethodBaseType">
    <xs:complexContent>
      <xs:restriction base="AuthnMethodBaseType">
        <xs:sequence>
          <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
          <xs:element ref="Authenticator"/>
          <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="PrincipalAuthenticationMechanismType">
    <xs:complexContent>
      <xs:restriction base="PrincipalAuthenticationMechanismType">
        <xs:sequence>
          <xs:element ref="Token"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="TokenType">
    <xs:complexContent>
      <xs:restriction base="TokenType">
        <xs:sequence>
          <xs:element ref="TimeSyncToken"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="TimeSyncTokenType">
    <xs:complexContent>
      <xs:restriction base="TimeSyncTokenType">
        <xs:attribute name="DeviceType" use="required">
          <xs:simpleType>
            <xs:restriction base="DeviceTypeType">
```

```
                <xs:enumeration value="hardware"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>

          <xs:attribute name="SeedLength" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:integer">
                <xs:minInclusive value="64"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>

          <xs:attribute name="DeviceInHand" use="required">
            <xs:simpleType>
              <xs:restriction base="booleanType">
                <xs:enumeration value="true"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.24 SAML Schema Authentication Context types

This is the SAML Authentication Context types Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  version="2.0">

  <xs:annotation>
    <xs:documentation>
      Document identifier: saml-schema-authn-context-types-2.0
      Location: http://docs.oasis-open.org/security/saml/v2.0/
      Revision history:
          V2.0 (March, 2005):
          New core authentication context schema types for SAML V2.0.
    </xs:documentation>
  </xs:annotation>

  <xs:element name="AuthenticationContextDeclaration"
type="AuthnContextDeclarationBaseType">
    <xs:annotation>
      <xs:documentation>
        A particular assertion_u111 on an identity
        provider's part with respect to the authentication
        context associated with an authentication assertion.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="Identification" type="IdentificationType">
    <xs:annotation>
      <xs:documentation>
        Refers to those characteristics that describe the
        processes and mechanisms
        the Authentication Authority uses to initially create
        an association between_u97 ? Principal
        and the identity (or name) by which the Principal will
        be known
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="PhysicalVerification">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 that identification has been
        performed in a physical
        face-to-face meeting with the principal and not in an
        online manner.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:attribute name="credentialLevel">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="primary"/>
            <xs:enumeration value="secondary"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="WrittenConsent" type="ExtensionOnlyType"/>

  <xs:element name="TechnicalProtection" type="TechnicalProtectionBaseType">
    <xs:annotation>
      <xs:documentation>
        Refers to those characterstics that describe how the
        'secret'  (the knowledge or possession
        of which allows the Principal to authenticate to the
        Authentication Authority) is kept secure
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="SecretKeyProtection" type="SecretKeyProtectionType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 the types and strengths of
        facilities
        of a UA used to protect a shared secret key from
        unauthorized access and/or use.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="PrivateKeyProtection" type="PrivateKeyProtectionType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 the types and strengths of
        facilities
        of a UA used to protect a private key from
        unauthorized access and/or use.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="KeyActivation" type="KeyActivationType">
    <xs:annotation>
      <xs:documentation>The actions that must be performed
        before the private key_u99 can be used. </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="KeySharing" type="KeySharingType">
    <xs:annotation>
      <xs:documentation>Whether or not the private key_u105 is shared
        with the certificate authority.</xs:documentation>
    </xs:annotation>
```

```
</xs:element>

<xs:element name="KeyStorage" type="KeyStorageType">
  <xs:annotation>
    <xs:documentation>
      In which medium is the_u107 key stored.
      memory - the key is stored in memory.
      smartcard - the key is_u115 stored in a smartcard.
      token - the key is stored in a hardware token.
      MobileDevice - the key_u105 is stored in a mobile device.
      MobileAuthCard - the key is stored in a mobile
      authentication card.
    </xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="SubscriberLineNumber" type="ExtensionOnlyType"/>
<xs:element name="UserSuffix" type="ExtensionOnlyType"/>

<xs:element name="Password" type="PasswordType">
  <xs:annotation>
    <xs:documentation>
      This element indicates_u116 that a password (or passphrase)
      has been used to
      authenticate the Principal to a remote system.
    </xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="ActivationPin" type="ActivationPinType">
  <xs:annotation>
    <xs:documentation>
      This element indicates_u116 that a Pin (Personal
      Identification Number)_u104 has been used to authenticate the Principal
      to some local system in order to activate a key.
    </xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="Token" type="TokenType">
  <xs:annotation>
    <xs:documentation>
      This element indicates_u116 that a hardware or software
      token is used
      as a method of identifying the Principal.
    </xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="TimeSyncToken" type="TimeSyncTokenType">
  <xs:annotation>
    <xs:documentation>
      This element indicates_u116 that a time synchronization
      token is used to identify the Principal. hardware -
      the time synchonization
      token has been implemented in hardware. software - the
      time synchronization
      token has been implemented in software. SeedLength -
      the length, in bits, of the
      random seed used in the time synchronization token.
    </xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="Smartcard" type="ExtensionOnlyType">
  <xs:annotation>
    <xs:documentation>
      This element indicates_u116 that a smartcard is used to
      identify the Principal.
    </xs:documentation>
  </xs:annotation>
  </xs:element>

  <xs:element name="Length" type="LengthType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 the minimum and/or maximum
        ASCII length of the password which is enforced (by the UA or the
        IdP). In other words, this is the minimum and/or maximum number of
        ASCII characters required to represent a valid password.
        min - the minimum number of ASCII characters required
        in a valid password, as enforced by the UA or the IdP.
        max - the maximum number of ASCII characters required
        in a valid password, as enforced by the UA or the IdP.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="ActivationLimit" type="ActivationLimitType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 the length of time for which an
        PIN-based authentication is valid.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="Generation">
    <xs:annotation>
      <xs:documentation>
        Indicates whether the password was chosen by the
        Principal or auto-supplied by the Authentication Authority.
        principal chosen - the Principal is allowed to choose
        the value of the password. This is true even if
        the initial password is chosen at random by the UA or
        the IdP and the Principal is then free to change
        the password.
        automatic - the password is chosen by the UA or the
        IdP to be cryptographically strong in some sense,
        or to satisfy certain password rules, and that the
        Principal is not free to change it or to choose a new password.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType>
      <xs:attribute name="mechanism" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="principalchosen"/>
            <xs:enumeration value="automatic"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="AuthnMethod" type="AuthnMethodBaseType">
    <xs:annotation>
      <xs:documentation>
        Refers to those characteristics that define the
        mechanisms by which the Principal authenticates to the Authentication
        Authority.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="PrincipalAuthenticationMechanism"
type="PrincipalAuthenticationMechanismType">
    <xs:annotation>
      <xs:documentation>
        The method that a Principal employs to perform
        authentication to local system components.
```

```
   </xs:documentation>
  </xs:annotation>
 </xs:element>

 <xs:element name="Authenticator" type="AuthenticatorBaseType">
  <xs:annotation>
   <xs:documentation>
    The method applied to validate a principal's
    authentication across a network
   </xs:documentation>
  </xs:annotation>
 </xs:element>

 <xs:element name="ComplexAuthenticator" type="ComplexAuthenticatorType">
  <xs:annotation>
   <xs:documentation>
    Supports Authenticators with nested combinations of
    additional complexity.
   </xs:documentation>
  </xs:annotation>
 </xs:element>

 <xs:element name="PreviousSession" type="ExtensionOnlyType">
  <xs:annotation>
   <xs:documentation>
    Indicates that the Principal has been strongly
    authenticated in a previous session during which the IdP has set a
    cookie in the UA. During the present session the Principal has only
    been authenticated by the UA returning the cookie to the IdP.
   </xs:documentation>
  </xs:annotation>
 </xs:element>

 <xs:element name="ResumeSession" type="ExtensionOnlyType">
  <xs:annotation>
   <xs:documentation>
    Rather like PreviousSession but using stronger
    security. A secret that was established in a previous session with
    the Authentication Authority has been cached by the local system and
    is now re-used (e.g. a_u77 master Secret is used to derive new session
    keys in TLS, SSL, WTLS).
   </xs:documentation>
  </xs:annotation>
 </xs:element>

 <xs:element name="ZeroKnowledge" type="ExtensionOnlyType">
  <xs:annotation>
   <xs:documentation>
    This element indicates_u116 that the Principal has been
    authenticated by a zero knowledge technique as specified in ISO/IEC
    9798-5.
   </xs:documentation>
  </xs:annotation>
 </xs:element>

 <xs:element name="SharedSecretChallengeResponse"
type="SharedSecretChallengeResponseType"/>

 <xs:complexType name="SharedSecretChallengeResponseType">
  <xs:annotation>
   <xs:documentation>
    This element indicates_u116 that the Principal has been
    authenticated by a challenge-response protocol utilizing shared secret
    keys and symmetric cryptography.
   </xs:documentation>
  </xs:annotation>
  <xs:sequence>
   <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="method" type="xs:anyURI" use="optional"/>
 </xs:complexType>
```

```
  <xs:element name="DigSig" type="PublicKeyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 that the Principal has been
        authenticated by a mechanism which involves the Principal computing a
        digital signature over_u97 at least challenge data provided by the IdP.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="AsymmetricDecryption" type="PublicKeyType">
    <xs:annotation>
      <xs:documentation>
        The local system has a_u112 Private key but it is used
        in decryption mode, rather than signature mode. For example, the
        Authentication Authority generates a secret and encrypts it using the
        local system's public key: the local system then proves it has
        decrypted the secret.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="AsymmetricKeyAgreement" type="PublicKeyType">
    <xs:annotation>
      <xs:documentation>
        The local system has a_u112 Private key and uses it for
        shared secret key agreement with the Authentication Authority (e.g.
        via Diffie Helman).
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:complexType name="PublicKeyType">
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="keyValidation" use="optional"/>
  </xs:complexType>

  <xs:element name="IPAddress" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 that the Principal has been
        authenticated through connection from a particular IP address.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="SharedSecretDynamicPlaintext" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        The local system and Authentication Authority
        share a secret key. The local system uses this to encrypt a
        randomised string to pass to the Authentication Authority.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="AuthenticatorTransportProtocol"
type="AuthenticatorTransportProtocolType">
    <xs:annotation>
      <xs:documentation>
        The protocol across which Authenticator information is
        transferred to an Authentication Authority verifier.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="HTTP" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 that the Authenticator has been
        transmitted using bare_u72 HTTP utilizing no additional security
        protocols.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="IPSec" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 that the Authenticator has been
        transmitted using a transport mechanism protected by an IPSEC session.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="WTLS" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 that the Authenticator has been
        transmitted using a transport mechanism protected by a WTLS session.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="MobileNetworkNoEncryption" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 that the Authenticator has been
        transmitted solely across a mobile network using no additional
        security mechanism.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="MobileNetworkRadioEncryption" type="ExtensionOnlyType"/>
  <xs:element name="MobileNetworkEndToEndEncryption" type="ExtensionOnlyType"/>

  <xs:element name="SSL" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 that the Authenticator has been
        transmitted using a transport mechnanism protected by an SSL or TLS
        session.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="PSTN" type="ExtensionOnlyType"/>
  <xs:element name="ISDN" type="ExtensionOnlyType"/>
  <xs:element name="ADSL" type="ExtensionOnlyType"/>

  <xs:element name="OperationalProtection" type="OperationalProtectionType">
    <xs:annotation>
      <xs:documentation>
        Refers to those characteristics that describe
        procedural security controls employed by the Authentication Authority.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="SecurityAudit" type="SecurityAuditType"/>
  <xs:element name="SwitchAudit" type="ExtensionOnlyType"/>
  <xs:element name="DeactivationCallCenter" type="ExtensionOnlyType"/>

  <xs:element name="GoverningAgreements" type="GoverningAgreementsType">
    <xs:annotation>
      <xs:documentation>
        Provides a mechanism for linking to external (likely
        human readable) documents in which additional business agreements,
        (e.g. liability constraints, obligations, etc.) can be placed.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="GoverningAgreementRef" type="GoverningAgreementRefType"/>

  <xs:simpleType name="nymType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="anonymity"/>
      <xs:enumeration value="verinymity"/>
      <xs:enumeration value="pseudonymity"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="AuthnContextDeclarationBaseType">
    <xs:sequence>
      <xs:element ref="Identification" minOccurs="0"/>
      <xs:element ref="TechnicalProtection" minOccurs="0"/>
      <xs:element ref="OperationalProtection" minOccurs="0"/>
      <xs:element ref="AuthnMethod" minOccurs="0"/>
      <xs:element ref="GoverningAgreements" minOccurs="0"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:ID" use="optional"/>
  </xs:complexType>

  <xs:complexType name="IdentificationType">
    <xs:sequence>
      <xs:element ref="PhysicalVerification" minOccurs="0"/>
      <xs:element ref="WrittenConsent" minOccurs="0"/>
      <xs:element ref="GoverningAgreements" minOccurs="0"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="nym" type="nymType">
      <xs:annotation>
        <xs:documentation>
          This attribute indicates whether or not the
          Identification mechanisms allow the actions of the Principal to be
          linked to an actual end user.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>

  <xs:complexType name="TechnicalProtectionBaseType">
    <xs:sequence>
      <xs:choice minOccurs="0">
        <xs:element ref="PrivateKeyProtection"/>
        <xs:element ref="SecretKeyProtection"/>
      </xs:choice>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="OperationalProtectionType">
    <xs:sequence>
      <xs:element ref="SecurityAudit" minOccurs="0"/>
      <xs:element ref="DeactivationCallCenter" minOccurs="0"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="AuthnMethodBaseType">
    <xs:sequence>
      <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
      <xs:element ref="Authenticator" minOccurs="0"/>
      <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
```

```xml
<xs:complexType name="GoverningAgreementsType">
  <xs:sequence>
    <xs:element ref="GoverningAgreementRef" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="GoverningAgreementRefType">
  <xs:attribute name="governingAgreementRef" type="xs:anyURI" use="required"/>
</xs:complexType>

<xs:complexType name="PrincipalAuthenticationMechanismType">
  <xs:sequence>
    <xs:element ref="Password" minOccurs="0"/>
    <xs:element ref="RestrictedPassword" minOccurs="0"/>
    <xs:element ref="Token" minOccurs="0"/>
    <xs:element ref="Smartcard" minOccurs="0"/>
    <xs:element ref="ActivationPin" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="preauth" type="xs:integer" use="optional"/>
</xs:complexType>

<xs:group name="AuthenticatorChoiceGroup">
  <xs:choice>
    <xs:element ref="PreviousSession"/>
    <xs:element ref="ResumeSession"/>
    <xs:element ref="DigSig"/>
    <xs:element ref="Password"/>
    <xs:element ref="RestrictedPassword"/>
    <xs:element ref="ZeroKnowledge"/>
    <xs:element ref="SharedSecretChallengeResponse"/>
    <xs:element ref="SharedSecretDynamicPlaintext"/>
    <xs:element ref="IPAddress"/>
    <xs:element ref="AsymmetricDecryption"/>
    <xs:element ref="AsymmetricKeyAgreement"/>
    <xs:element ref="SubscriberLineNumber"/>
    <xs:element ref="UserSuffix"/>
    <xs:element ref="ComplexAuthenticator"/>
  </xs:choice>
</xs:group>

<xs:group name="AuthenticatorSequenceGroup">
  <xs:sequence>
    <xs:element ref="PreviousSession" minOccurs="0"/>
    <xs:element ref="ResumeSession" minOccurs="0"/>
    <xs:element ref="DigSig" minOccurs="0"/>
    <xs:element ref="Password" minOccurs="0"/>
    <xs:element ref="RestrictedPassword" minOccurs="0"/>
    <xs:element ref="ZeroKnowledge" minOccurs="0"/>
    <xs:element ref="SharedSecretChallengeResponse" minOccurs="0"/>
    <xs:element ref="SharedSecretDynamicPlaintext" minOccurs="0"/>
    <xs:element ref="IPAddress" minOccurs="0"/>
    <xs:element ref="AsymmetricDecryption" minOccurs="0"/>
    <xs:element ref="AsymmetricKeyAgreement" minOccurs="0"/>
    <xs:element ref="SubscriberLineNumber" minOccurs="0"/>
    <xs:element ref="UserSuffix" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:group>

<xs:complexType name="AuthenticatorBaseType">
  <xs:sequence>
    <xs:group ref="AuthenticatorChoiceGroup"/>
    <xs:group ref="AuthenticatorSequenceGroup"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ComplexAuthenticatorType">
  <xs:sequence>
    <xs:group ref="AuthenticatorChoiceGroup"/>
```

```
      <xs:group ref="AuthenticatorSequenceGroup"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="AuthenticatorTransportProtocolType">
    <xs:sequence>
      <xs:choice minOccurs="0">
        <xs:element ref="HTTP"/>
        <xs:element ref="SSL"/>
        <xs:element ref="MobileNetworkNoEncryption"/>
        <xs:element ref="MobileNetworkRadioEncryption"/>
        <xs:element ref="MobileNetworkEndToEndEncryption"/>
        <xs:element ref="WTLS"/>
        <xs:element ref="IPSec"/>
        <xs:element ref="PSTN"/>
        <xs:element ref="ISDN"/>
        <xs:element ref="ADSL"/>
      </xs:choice>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="KeyActivationType">
    <xs:sequence>
      <xs:element ref="ActivationPin" minOccurs="0"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="KeySharingType">
    <xs:attribute name="sharing" type="xs:boolean" use="required"/>
  </xs:complexType>

  <xs:complexType name="PrivateKeyProtectionType">
    <xs:sequence>
      <xs:element ref="KeyActivation" minOccurs="0"/>
      <xs:element ref="KeyStorage" minOccurs="0"/>
      <xs:element ref="KeySharing" minOccurs="0"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="PasswordType">
    <xs:sequence>
      <xs:element ref="Length" minOccurs="0"/>
      <xs:element ref="Alphabet" minOccurs="0"/>
      <xs:element ref="Generation" minOccurs="0"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ExternalVerification" type="xs:anyURI" use="optional"/>
  </xs:complexType>

  <xs:element name="RestrictedPassword" type="RestrictedPasswordType"/>

  <xs:complexType name="RestrictedPasswordType">
    <xs:complexContent>
      <xs:restriction base="PasswordType">
        <xs:sequence>
          <xs:element name="Length" type="RestrictedLengthType" minOccurs="1"/>
          <xs:element ref="Generation" minOccurs="0"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="ExternalVerification" type="xs:anyURI"
use="optional"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="RestrictedLengthType">
    <xs:complexContent>
      <xs:restriction base="LengthType">
        <xs:attribute name="min" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:integer">
              <xs:minInclusive value="3"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="max" type="xs:integer" use="optional"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="ActivationPinType">
    <xs:sequence>
      <xs:element ref="Length" minOccurs="0"/>
      <xs:element ref="Alphabet" minOccurs="0"/>
      <xs:element ref="Generation" minOccurs="0"/>
      <xs:element ref="ActivationLimit" minOccurs="0"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="Alphabet" type="AlphabetType"/>
  <xs:complexType name="AlphabetType">
    <xs:attribute name="requiredChars" type="xs:string" use="required"/>
    <xs:attribute name="excludedChars" type="xs:string" use="optional"/>
    <xs:attribute name="case" type="xs:string" use="optional"/>
  </xs:complexType>

  <xs:complexType name="TokenType">
    <xs:sequence>
      <xs:element ref="TimeSyncToken"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="DeviceTypeType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="hardware"/>
      <xs:enumeration value="software"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="booleanType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="true"/>
      <xs:enumeration value="false"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="TimeSyncTokenType">
    <xs:attribute name="DeviceType" type="DeviceTypeType" use="required"/>
    <xs:attribute name="SeedLength" type="xs:integer" use="required"/>
    <xs:attribute name="DeviceInHand" type="booleanType" use="required"/>
  </xs:complexType>

  <xs:complexType name="ActivationLimitType">
    <xs:choice>
      <xs:element ref="ActivationLimitDuration"/>
      <xs:element ref="ActivationLimitUsages"/>
      <xs:element ref="ActivationLimitSession"/>
    </xs:choice>
  </xs:complexType>

  <xs:element name="ActivationLimitDuration" type="ActivationLimitDurationType">
    <xs:annotation>
      <xs:documentation>
        This element indicates_u116 that the Key Activation Limit is
        defined as a specific duration of time.
      </xs:documentation>
    </xs:annotation>
```

```
</xs:element>

<xs:element name="ActivationLimitUsages" type="ActivationLimitUsagesType">
  <xs:annotation>
    <xs:documentation>
      This element indicates_u116 that the Key Activation Limit is
      defined as a number of_u117 usages.
    </xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="ActivationLimitSession" type="ActivationLimitSessionType">
  <xs:annotation>
    <xs:documentation>
      This element indicates_u116 that the Key Activation Limit is
      the session.
    </xs:documentation>
  </xs:annotation>
</xs:element>

<xs:complexType name="ActivationLimitDurationType">
  <xs:attribute name="duration" type="xs:duration" use="required"/>
</xs:complexType>

<xs:complexType name="ActivationLimitUsagesType">
  <xs:attribute name="number" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="ActivationLimitSessionType"/>

<xs:complexType name="LengthType">
  <xs:attribute name="min" type="xs:integer" use="required"/>
  <xs:attribute name="max" type="xs:integer" use="optional"/>
</xs:complexType>

<xs:simpleType name="mediumType">
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="memory"/>
    <xs:enumeration value="smartcard"/>
    <xs:enumeration value="token"/>
    <xs:enumeration value="MobileDevice"/>
    <xs:enumeration value="MobileAuthCard"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="KeyStorageType">
  <xs:attribute name="medium" type="mediumType" use="required"/>
</xs:complexType>

<xs:complexType name="SecretKeyProtectionType">
  <xs:sequence>
    <xs:element ref="KeyActivation" minOccurs="0"/>
    <xs:element ref="KeyStorage" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="SecurityAuditType">
  <xs:sequence>
    <xs:element ref="SwitchAudit" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ExtensionOnlyType">
  <xs:sequence>
    <xs:element ref="Extension" minOccurs="0"  maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="Extension" type="ExtensionType"/>
```

```
  <xs:complexType name="ExtensionType">
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

A.25 SAML Schema Authentication Context X.509

This is the X.509 SAML Authentication Context Schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:X509"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:X509"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:X509
        Document identifier: saml-schema-authn-context-x509-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="DigSig"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PublicKeyType">
      <xs:complexContent>
        <xs:restriction base="PublicKeyType">
          <xs:attribute name="keyValidation" type="xs:anyURI"
fixed="urn:oasis:names:tc:SAML:2.0:ac:classes:X509"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.26 SAML Schema Authentication Context XMLDSig

This is the XML digital signature SAML Authentication Context Schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:XMLDSig"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:XMLDSig"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:XMLDSig
        Document identifier: saml-schema-authn-context-xmldsig-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="DigSig"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PublicKeyType">
      <xs:complexContent>
        <xs:restriction base="PublicKeyType">
          <xs:attribute name="keyValidation" type="xs:anyURI"
fixed="urn:ietf:rfc:3075"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

A.27 SAML Schema ECP

This is the SAML Schema listing of the enhanced client or proxy (ECP) profile.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
        schemaLocation="saml-schema-protocol-2.0.xsd"/>
    <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="saml-schema-assertion-2.0.xsd"/>
    <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
        schemaLocation="http://schemas.xmlsoap.org/soap/envelope/"/>
    <annotation>
        <documentation>
            Document identifier: saml-schema-ecp-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
              V2.0 (March, 2005):
                Custom schema for ECP profile, first published in SAML 2.0.
        </documentation>
    </annotation>

    <element name="Request" type="ecp:RequestType"/>
    <complexType name="RequestType">
        <sequence>
            <element ref="saml:Issuer"/>
            <element ref="samlp:IDPList" minOccurs="0"/>
        </sequence>
        <attribute ref="S:mustUnderstand" use="required"/>
        <attribute ref="S:actor" use="required"/>
        <attribute name="ProviderName" type="string" use="optional"/>
        <attribute name="IsPassive" type="boolean" use="optional"/>
    </complexType>

    <element name="Response" type="ecp:ResponseType"/>
    <complexType name="ResponseType">
        <attribute ref="S:mustUnderstand" use="required"/>
        <attribute ref="S:actor" use="required"/>
        <attribute name="AssertionConsumerServiceURL" type="anyURI"
use="required"/>
    </complexType>

    <element name="RelayState" type="ecp:RelayStateType"/>
    <complexType name="RelayStateType">
        <simpleContent>
            <extension base="string">
                <attribute ref="S:mustUnderstand" use="required"/>
                <attribute ref="S:actor" use="required"/>
            </extension>
        </simpleContent>
    </complexType>
</schema>
```

A.28 SAML Schema metadata

This is the listing for the SAML metadata Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
20020212/xmldsig-core-schema.xsd"/>
    <import namespace="http://www.w3.org/2001/04/xmlenc#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-
schema.xsd"/>
    <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="saml-schema-assertion-2.0.xsd"/>
    <import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd"/>
    <annotation>
        <documentation>
            Document identifier: saml-schema-metadata-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
              V2.0 (March, 2005):
                Schema for SAML metadata, first published in SAML 2.0.
        </documentation>
    </annotation>

    <simpleType name="entityIDType">
        <restriction base="anyURI">
            <maxLength value="1024"/>
        </restriction>
    </simpleType>
    <complexType name="localizedNameType">
        <simpleContent>
            <extension base="string">
                <attribute ref="xml:lang" use="required"/>
            </extension>
        </simpleContent>
    </complexType>
    <complexType name="localizedURIType">
        <simpleContent>
            <extension base="anyURI">
                <attribute ref="xml:lang" use="required"/>
            </extension>
        </simpleContent>
    </complexType>

    <element name="Extensions" type="md:ExtensionsType"/>
    <complexType final="#all" name="ExtensionsType">
        <sequence>
            <any namespace="##other" processContents="lax"
maxOccurs="unbounded"/>
        </sequence>
    </complexType>

    <complexType name="EndpointType">
        <sequence>
            <any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
        <attribute name="Binding" type="anyURI" use="required"/>
        <attribute name="Location" type="anyURI" use="required"/>
        <attribute name="ResponseLocation" type="anyURI" use="optional"/>
        <anyAttribute namespace="##other" processContents="lax"/>
    </complexType>

    <complexType name="IndexedEndpointType">
        <complexContent>
            <extension base="md:EndpointType">
                <attribute name="index" type="unsignedShort" use="required"/>
                <attribute name="isDefault" type="boolean" use="optional"/>
            </extension>
        </complexContent>
    </complexType>

    <element name="EntitiesDescriptor" type="md:EntitiesDescriptorType"/>
    <complexType name="EntitiesDescriptorType">
        <sequence>
            <element ref="ds:Signature" minOccurs="0"/>
            <element ref="md:Extensions" minOccurs="0"/>
            <choice minOccurs="1" maxOccurs="unbounded">
                <element ref="md:EntityDescriptor"/>
                <element ref="md:EntitiesDescriptor"/>
            </choice>
        </sequence>
        <attribute name="validUntil" type="dateTime" use="optional"/>
        <attribute name="cacheDuration" type="duration" use="optional"/>
        <attribute name="ID" type="ID" use="optional"/>
        <attribute name="Name" type="string" use="optional"/>
    </complexType>

    <element name="EntityDescriptor" type="md:EntityDescriptorType"/>
    <complexType name="EntityDescriptorType">
        <sequence>
            <element ref="ds:Signature" minOccurs="0"/>
            <element ref="md:Extensions" minOccurs="0"/>
            <choice>
                <choice maxOccurs="unbounded">
                    <element ref="md:RoleDescriptor"/>
                    <element ref="md:IDPSSODescriptor"/>
                    <element ref="md:SPSSODescriptor"/>
                    <element ref="md:AuthnAuthorityDescriptor"/>
                    <element ref="md:AttributeAuthorityDescriptor"/>
                    <element ref="md:PDPDescriptor"/>
                </choice>
                <element ref="md:AffiliationDescriptor"/>
            </choice>
            <element ref="md:Organization" minOccurs="0"/>
            <element ref="md:ContactPerson" minOccurs="0" maxOccurs="unbounded"/>
            <element ref="md:AdditionalMetadataLocation" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
        <attribute name="entityID" type="md:entityIDType" use="required"/>
        <attribute name="validUntil" type="dateTime" use="optional"/>
        <attribute name="cacheDuration" type="duration" use="optional"/>
        <attribute name="ID" type="ID" use="optional"/>
        <anyAttribute namespace="##other" processContents="lax"/>
    </complexType>

    <element name="Organization" type="md:OrganizationType"/>
    <complexType name="OrganizationType">
        <sequence>
            <element ref="md:Extensions" minOccurs="0"/>
            <element ref="md:OrganizationName" maxOccurs="unbounded"/>
            <element ref="md:OrganizationDisplayName" maxOccurs="unbounded"/>
            <element ref="md:OrganizationURL" maxOccurs="unbounded"/>
        </sequence>
        <anyAttribute namespace="##other" processContents="lax"/>
    </complexType>
    <element name="OrganizationName" type="md:localizedNameType"/>
    <element name="OrganizationDisplayName" type="md:localizedNameType"/>
    <element name="OrganizationURL" type="md:localizedURIType"/>
    <element name="ContactPerson" type="md:ContactType"/>
    <complexType name="ContactType">
        <sequence>
            <element ref="md:Extensions" minOccurs="0"/>
            <element ref="md:Company" minOccurs="0"/>
            <element ref="md:GivenName" minOccurs="0"/>
            <element ref="md:SurName" minOccurs="0"/>
            <element ref="md:EmailAddress" minOccurs="0" maxOccurs="unbounded"/>
            <element ref="md:TelephoneNumber" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
        <attribute name="contactType" type="md:ContactTypeType" use="required"/>
        <anyAttribute namespace="##other" processContents="lax"/>
    </complexType>
    <element name="Company" type="string"/>
    <element name="GivenName" type="string"/>
    <element name="SurName" type="string"/>
    <element name="EmailAddress" type="anyURI"/>
    <element name="TelephoneNumber" type="string"/>
    <simpleType name="ContactTypeType">
        <restriction base="string">
            <enumeration value="technical"/>
            <enumeration value="support"/>
            <enumeration value="administrative"/>
            <enumeration value="billing"/>
            <enumeration value="other"/>
        </restriction>
    </simpleType>

    <element name="AdditionalMetadataLocation"
type="md:AdditionalMetadataLocationType"/>
    <complexType name="AdditionalMetadataLocationType">
        <simpleContent>
            <extension base="anyURI">
                <attribute name="namespace" type="anyURI" use="required"/>
            </extension>
        </simpleContent>
    </complexType>

    <element name="RoleDescriptor" type="md:RoleDescriptorType"/>
    <complexType name="RoleDescriptorType" abstract="true">
        <sequence>
            <element ref="ds:Signature" minOccurs="0"/>
            <element ref="md:Extensions" minOccurs="0"/>
            <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
            <element ref="md:Organization" minOccurs="0"/>
            <element ref="md:ContactPerson" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="ID" type="ID" use="optional"/>
        <attribute name="validUntil" type="dateTime" use="optional"/>
        <attribute name="cacheDuration" type="duration" use="optional"/>
        <attribute name="protocolSupportEnumeration" type="md:anyURIListType"
use="required"/>
        <attribute name="errorURL" type="anyURI" use="optional"/>
        <anyAttribute namespace="##other" processContents="lax"/>
    </complexType>
    <simpleType name="anyURIListType">
        <list itemType="anyURI"/>
    </simpleType>

    <element name="KeyDescriptor" type="md:KeyDescriptorType"/>
    <complexType name="KeyDescriptorType">
        <sequence>
            <element ref="ds:KeyInfo"/>
            <element ref="md:EncryptionMethod" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
        <attribute name="use" type="md:KeyTypes" use="optional"/>
    </complexType>
    <simpleType name="KeyTypes">
        <restriction base="string">
            <enumeration value="encryption"/>
            <enumeration value="signing"/>
        </restriction>
    </simpleType>
    <element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>

    <complexType name="SSODescriptorType" abstract="true">
        <complexContent>
            <extension base="md:RoleDescriptorType">
                <sequence>
                    <element ref="md:ArtifactResolutionService" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="md:SingleLogoutService" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="md:ManageNameIDService" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="ArtifactResolutionService" type="md:IndexedEndpointType"/>
    <element name="SingleLogoutService" type="md:EndpointType"/>
    <element name="ManageNameIDService" type="md:EndpointType"/>
    <element name="NameIDFormat" type="anyURI"/>
    <element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>
    <complexType name="IDPSSODescriptorType">
        <complexContent>
            <extension base="md:SSODescriptorType">
                <sequence>
                    <element ref="md:SingleSignOnService" maxOccurs="unbounded"/>
                    <element ref="md:NameIDMappingService" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="md:AssertionIDRequestService" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="md:AttributeProfile" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="saml:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
                <attribute name="WantAuthnRequestsSigned" type="boolean"
use="optional"/>
            </extension>
        </complexContent>
    </complexType>
    <element name="SingleSignOnService" type="md:EndpointType"/>
    <element name="NameIDMappingService" type="md:EndpointType"/>
    <element name="AssertionIDRequestService" type="md:EndpointType"/>
    <element name="AttributeProfile" type="anyURI"/>

    <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
    <complexType name="SPSSODescriptorType">
        <complexContent>
            <extension base="md:SSODescriptorType">
                <sequence>
                    <element ref="md:AssertionConsumerService"
maxOccurs="unbounded"/>
                    <element ref="md:AttributeConsumingService" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
                <attribute name="AuthnRequestsSigned" type="boolean"
use="optional"/>
                <attribute name="WantAssertionsSigned" type="boolean"
use="optional"/>
            </extension>
        </complexContent>
    </complexType>
    <element name="AssertionConsumerService" type="md:IndexedEndpointType"/>
    <element name="AttributeConsumingService"
type="md:AttributeConsumingServiceType"/>
    <complexType name="AttributeConsumingServiceType">
        <sequence>
            <element ref="md:ServiceName" maxOccurs="unbounded"/>
            <element ref="md:ServiceDescription" minOccurs="0"
maxOccurs="unbounded"/>
            <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="index" type="unsignedShort" use="required"/>
        <attribute name="isDefault" type="boolean" use="optional"/>
    </complexType>
    <element name="ServiceName" type="md:localizedNameType"/>
    <element name="ServiceDescription" type="md:localizedNameType"/>
    <element name="RequestedAttribute" type="md:RequestedAttributeType"/>
    <complexType name="RequestedAttributeType">
        <complexContent>
            <extension base="saml:AttributeType">
                <attribute name="isRequired" type="boolean" use="optional"/>
            </extension>
        </complexContent>
    </complexType>

    <element name="AuthnAuthorityDescriptor"
type="md:AuthnAuthorityDescriptorType"/>
    <complexType name="AuthnAuthorityDescriptorType">
        <complexContent>
            <extension base="md:RoleDescriptorType">
```

```
                <sequence>
                    <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>
                    <element ref="md:AssertionIDRequestService" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="AuthnQueryService" type="md:EndpointType"/>

    <element name="PDPDescriptor" type="md:PDPDescriptorType"/>
    <complexType name="PDPDescriptorType">
        <complexContent>
            <extension base="md:RoleDescriptorType">
                <sequence>
                    <element ref="md:AuthzService" maxOccurs="unbounded"/>
                    <element ref="md:AssertionIDRequestService" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="AuthzService" type="md:EndpointType"/>

    <element name="AttributeAuthorityDescriptor"
type="md:AttributeAuthorityDescriptorType"/>
    <complexType name="AttributeAuthorityDescriptorType">
        <complexContent>
            <extension base="md:RoleDescriptorType">
                <sequence>
                    <element ref="md:AttributeService" maxOccurs="unbounded"/>
                    <element ref="md:AssertionIDRequestService" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="md:AttributeProfile" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="saml:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="AttributeService" type="md:EndpointType"/>

    <element name="AffiliationDescriptor" type="md:AffiliationDescriptorType"/>
    <complexType name="AffiliationDescriptorType">
        <sequence>
            <element ref="ds:Signature" minOccurs="0"/>
            <element ref="md:Extensions" minOccurs="0"/>
            <element ref="md:AffiliateMember" maxOccurs="unbounded"/>
            <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="affiliationOwnerID" type="md:entityIDType"
use="required"/>
        <attribute name="validUntil" type="dateTime" use="optional"/>
        <attribute name="cacheDuration" type="duration" use="optional"/>
        <attribute name="ID" type="ID" use="optional"/>
        <anyAttribute namespace="##other" processContents="lax"/>
    </complexType>
    <element name="AffiliateMember" type="md:entityIDType"/>
</schema>
```

A.29 SAML Schema protocol

This is the listing for the Schema for the SAML protocol.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="saml-schema-assertion-2.0.xsd"/>
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
20020212/xmldsig-core-schema.xsd"/>
    <annotation>
        <documentation>
            Document identifier: saml-schema-protocol-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
            V1.0 (November, 2002):
              Initial Standard_u83 schema.
            V1.1 (September, 2003):
              Updates within the same V1.0 namespace.
            V2.0 (March, 2005):
              New protocol schema based in a SAML V2.0 namespace.
     </documentation>
    </annotation>
    <complexType name="RequestAbstractType" abstract="true">
        <sequence>
            <element ref="saml:Issuer" minOccurs="0"/>
            <element ref="ds:Signature" minOccurs="0"/>
            <element ref="samlp:Extensions" minOccurs="0"/>
        </sequence>
        <attribute name="ID" type="ID" use="required"/>
        <attribute name="Version" type="string" use="required"/>
        <attribute name="IssueInstant" type="dateTime" use="required"/>
        <attribute name="Destination" type="anyURI" use="optional"/>
          <attribute name="Consent" type="anyURI" use="optional"/>
    </complexType>
    <element name="Extensions" type="samlp:ExtensionsType"/>
    <complexType name="ExtensionsType">
        <sequence>
            <any namespace="##other" processContents="lax"
maxOccurs="unbounded"/>
        </sequence>
    </complexType>
    <complexType name="StatusResponseType">
          <sequence>
            <element ref="saml:Issuer" minOccurs="0"/>
            <element ref="ds:Signature" minOccurs="0"/>
            <element ref="samlp:Extensions" minOccurs="0"/>
            <element ref="samlp:Status"/>
          </sequence>
          <attribute name="ID" type="ID" use="required"/>
          <attribute name="InResponseTo" type="NCName" use="optional"/>
          <attribute name="Version" type="string" use="required"/>
          <attribute name="IssueInstant" type="dateTime" use="required"/>
          <attribute name="Destination" type="anyURI" use="optional"/>
          <attribute name="Consent" type="anyURI" use="optional"/>
    </complexType>
    <element name="Status" type="samlp:StatusType"/>
    <complexType name="StatusType">
        <sequence>
            <element ref="samlp:StatusCode"/>
            <element ref="samlp:StatusMessage" minOccurs="0"/>
            <element ref="samlp:StatusDetail" minOccurs="0"/>
        </sequence>
    </complexType>
    <element name="StatusCode" type="samlp:StatusCodeType"/>
    <complexType name="StatusCodeType">
        <sequence>
            <element ref="samlp:StatusCode" minOccurs="0"/>
        </sequence>
        <attribute name="Value" type="anyURI" use="required"/>
    </complexType>
    <element name="StatusMessage" type="string"/>
    <element name="StatusDetail" type="samlp:StatusDetailType"/>
    <complexType name="StatusDetailType">
        <sequence>
            <any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
    </complexType>
    <element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>
    <complexType name="AssertionIDRequestType">
          <complexContent>
            <extension base="samlp:RequestAbstractType">
                <sequence>
                    <element ref="saml:AssertionIDRef" maxOccurs="unbounded"/>
                </sequence>
            </extension>
          </complexContent>
    </complexType>
    <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
    <complexType name="SubjectQueryAbstractType" abstract="true">
          <complexContent>
            <extension base="samlp:RequestAbstractType">
                <sequence>
                    <element ref="saml:Subject"/>
                </sequence>
            </extension>
          </complexContent>
    </complexType>
    <element name="AuthnQuery" type="samlp:AuthnQueryType"/>
    <complexType name="AuthnQueryType">
        <complexContent>
            <extension base="samlp:SubjectQueryAbstractType">
                <sequence>
                    <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
                </sequence>
                <attribute name="SessionIndex" type="string" use="optional"/>
            </extension>
        </complexContent>
    </complexType>
    <element name="RequestedAuthnContext"
type="samlp:RequestedAuthnContextType"/>
    <complexType name="RequestedAuthnContextType">
        <choice>
            <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
            <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>
        </choice>
        <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
use="optional"/>
    </complexType>
    <simpleType name="AuthnContextComparisonType">
        <restriction base="string">
            <enumeration value="exact"/>
            <enumeration value="minimum"/>
            <enumeration value="maximum"/>
            <enumeration value="better"/>
        </restriction>
    </simpleType>
    <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
    <complexType name="AttributeQueryType">
        <complexContent>
            <extension base="samlp:SubjectQueryAbstractType">
                <sequence>
```

```
                    <element ref="saml:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>
    <complexType name="AuthzDecisionQueryType">
        <complexContent>
            <extension base="samlp:SubjectQueryAbstractType">
                <sequence>
                    <element ref="saml:Action" maxOccurs="unbounded"/>
                    <element ref="saml:Evidence" minOccurs="0"/>
                </sequence>
                <attribute name="Resource" type="anyURI" use="required"/>
            </extension>
        </complexContent>
    </complexType>
    <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
    <complexType name="AuthnRequestType">
        <complexContent>
            <extension base="samlp:RequestAbstractType">
                <sequence>
                    <element ref="saml:Subject" minOccurs="0"/>
                    <element ref="samlp:NameIDPolicy" minOccurs="0"/>
                    <element ref="saml:Conditions" minOccurs="0"/>
                    <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
                    <element ref="samlp:Scoping" minOccurs="0"/>
                </sequence>
                <attribute name="ForceAuthn" type="boolean" use="optional"/>
                <attribute name="IsPassive" type="boolean" use="optional"/>
                <attribute name="ProtocolBinding" type="anyURI" use="optional"/>
                <attribute name="AssertionConsumerServiceIndex"
type="unsignedShort" use="optional"/>
                <attribute name="AssertionConsumerServiceURL" type="anyURI"
use="optional"/>
                <attribute name="AttributeConsumingServiceIndex"
type="unsignedShort" use="optional"/>
                <attribute name="ProviderName" type="string" use="optional"/>
            </extension>
        </complexContent>
    </complexType>
    <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
    <complexType name="NameIDPolicyType">
        <attribute name="Format" type="anyURI" use="optional"/>
        <attribute name="SPNameQualifier" type="string" use="optional"/>
        <attribute name="AllowCreate" type="boolean" use="optional"/>
    </complexType>
    <element name="Scoping" type="samlp:ScopingType"/>
    <complexType name="ScopingType">
        <sequence>
            <element ref="samlp:IDPList" minOccurs="0"/>
            <element ref="samlp:RequesterID" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
        <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
    </complexType>
    <element name="RequesterID" type="anyURI"/>
    <element name="IDPList" type="samlp:IDPListType"/>
    <complexType name="IDPListType">
        <sequence>
            <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
            <element ref="samlp:GetComplete" minOccurs="0"/>
        </sequence>
    </complexType>
    <element name="IDPEntry" type="samlp:IDPEntryType"/>
    <complexType name="IDPEntryType">
        <attribute name="ProviderID" type="anyURI" use="required"/>
        <attribute name="Name" type="string" use="optional"/>
        <attribute name="Loc" type="anyURI" use="optional"/>
    </complexType>
    <element name="GetComplete" type="anyURI"/>
    <element name="Response" type="samlp:ResponseType"/>
    <complexType name="ResponseType">
          <complexContent>
            <extension base="samlp:StatusResponseType">
                <choice minOccurs="0" maxOccurs="unbounded">
                    <element ref="saml:Assertion"/>
                    <element ref="saml:EncryptedAssertion"/>
                </choice>
            </extension>
          </complexContent>
    </complexType>
    <element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
    <complexType name="ArtifactResolveType">
          <complexContent>
            <extension base="samlp:RequestAbstractType">
                <sequence>
                    <element ref="samlp:Artifact"/>
                </sequence>
            </extension>
          </complexContent>
    </complexType>
    <element name="Artifact" type="string"/>
    <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
    <complexType name="ArtifactResponseType">
          <complexContent>
            <extension base="samlp:StatusResponseType">
                <sequence>
                    <any namespace="##any" processContents="lax" minOccurs="0"/>
                </sequence>
            </extension>
          </complexContent>
    </complexType>
    <element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
    <complexType name="ManageNameIDRequestType">
          <complexContent>
            <extension base="samlp:RequestAbstractType">
                <sequence>
                    <choice>
                        <element ref="saml:NameID"/>
                        <element ref="saml:EncryptedID"/>
                    </choice>
                    <choice>
                        <element ref="samlp:NewID"/>
                        <element ref="samlp:NewEncryptedID"/>
                        <element ref="samlp:Terminate"/>
                    </choice>
                </sequence>
            </extension>
          </complexContent>
    </complexType>
    <element name="NewID" type="string"/>
    <element name="NewEncryptedID" type="saml:EncryptedElementType"/>
    <element name="Terminate" type="samlp:TerminateType"/>
    <complexType name="TerminateType"/>
    <element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>
    <element name="LogoutRequest" type="samlp:LogoutRequestType"/>
    <complexType name="LogoutRequestType">
        <complexContent>
            <extension base="samlp:RequestAbstractType">
                <sequence>
                    <choice>
                        <element ref="saml:BaseID"/>
                        <element ref="saml:NameID"/>
                        <element ref="saml:EncryptedID"/>
                    </choice>
                    <element ref="samlp:SessionIndex" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
                <attribute name="Reason" type="string" use="optional"/>
                <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
            </extension>
        </complexContent>
    </complexType>
    <element name="SessionIndex" type="string"/>
    <element name="LogoutResponse" type="samlp:StatusResponseType"/>
    <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
    <complexType name="NameIDMappingRequestType">
        <complexContent>
            <extension base="samlp:RequestAbstractType">
                <sequence>
                    <choice>
                        <element ref="saml:BaseID"/>
                        <element ref="saml:NameID"/>
                        <element ref="saml:EncryptedID"/>
                    </choice>
                    <element ref="samlp:NameIDPolicy"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="NameIDMappingResponse"
type="samlp:NameIDMappingResponseType"/>
    <complexType name="NameIDMappingResponseType">
        <complexContent>
            <extension base="samlp:StatusResponseType">
                <choice>
                    <element ref="saml:NameID"/>
                    <element ref="saml:EncryptedID"/>
                </choice>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

A.30 SAML Schema X.500

This is the listing for the X.500 SAML.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <annotation>
        <documentation>
            Document identifier: saml-schema-x500-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
              V2.0 (March, 2005):
                Custom schema for X.500 attribute profile, first published in SAML 2.0.
        </documentation>
    </annotation>
    <attribute name="Encoding" type="string"/>
</schema>
```

A.31 SAML Schema XACML

This is the listing for the XACML SAML.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <annotation>
        <documentation>
            Document identifier: saml-schema-xacml-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
            V2.0 (March, 2005):
              Custom schema for XACML attribute profile, first published in SAML 2.0.
        </documentation>
    </annotation>
    <attribute name="DataType" type="anyURI"/>
</schema>
```

Appendix I

Security and privacy considerations

Security and privacy must be addressed in a systemic manner, considering human issues such as social engineering attacks, policy issues, key management and trust management, secure implementation and other factors outside the scope of this appendix. Security technical solutions have a cost, so requirements and policy alternatives must also be considered, as must legal and regulatory requirements.

This appendix summarizes general security issues and approaches as well as specific threats and countermeasures for the use of SAML assertions, protocols, bindings and profiles in a secure manner that maintains privacy. This appendix describes and analyses the security and privacy properties SAML. The intent is to provide information to architects and implementers of SAML-based systems about the following:

- The privacy issues to be considered and how SAML architecture addresses these issues.
- The threats, and thus security risks, to which a SAML-based system is subject.
- The security risks the SAML architecture addresses, and how it does so.
- The security risks it does not address.
- Recommendations for countermeasures that mitigate those security risks.

I.1 Privacy

SAML includes the ability to make statements about the attributes and authorizations of authenticated entities. There are very many common situations in which the information carried in these statements is something that one or more of the parties to a communication would desire to keep accessible to as restricted as possible a set of entities. Statements of medical or financial attributes are simple examples of such cases.

Many countries and jurisdictions have laws and regulations regarding privacy and these should be considered when deploying a SAML based system. Parties making statements, issuing assertions, conveying assertions, and consuming assertions must be aware of these potential privacy concerns and should attempt to address them in their implementations of SAML-aware systems.

I.2 Confidentiality

Perhaps the most important aspect of ensuring privacy to parties in a SAML-enabled transaction is the ability to carry out the transaction with a guarantee of confidentiality. In other words, can the information in an assertion be conveyed from the issuer to the intended audience, and only the intended audience, without making it accessible to any other parties?

It is technically possible to convey information confidentially. All parties to SAML-enabled transactions should analyse each of their steps in the interaction (and any subsequent uses of data obtained from the transactions) to ensure that information that should be kept confidential is actually being kept so.

It should also be noted that simply obscuring the contents of assertions may not be adequate protection of privacy. There are many cases where just the availability of the information that a given user (or IP address) was accessing a given service may constitute a breach of privacy (for example, information that a user accessed a medical testing facility for an assertion may be enough to breach privacy without knowing the contents of the assertion). Partial solutions to these problems can be provided by various techniques for anonymous interaction as described in the next clauses.

I.3 Pseudonymity and anonymity

There are no definitions of anonymity that are satisfying for all cases. Many definitions deal with the simple case of a sender and a message, and discuss "anonymity" in terms of not being able to link a given sender to a sent message, or a message back to a sender. And while that definition is adequate for the "one off" case, it ignores the aggregation of information that is possible over time based on behaviour rather than an identifier.

In SAML is it helpful to think about anonymity as being "within a set". This notion is relevant to SAML because of the use of authorities. Even if a Subject is "anonymous", that subject is still identifiable as a member of the set of Subjects within the domain of the relevant authority. SAML-enabled systems are limited to "partial anonymity" at best because of the use of authorities. An entity about whom an assertion is made is already identifiable as one of the pool of entities in a relationship with the issuing authority.

The limitations on anonymity can be much worse than simple authority association, depending on how identifiers are employed, as reuse of pseudonymous identifiers allows accretion of potentially identifying information. Additionally, users of SAML-enabled systems can also make the breach of anonymity worse by their actions.

Apart from legal identity, any identifier for a Subject can be considered a pseudonym. And even notions like "holder of key" can be considered as serving as the equivalent of a pseudonym in linking an action (or set of actions) to a Subject. Even a description such as "the user that just requested access to object XYZ at time 23:34" can serve as an equivalent of a pseudonym.

Thus, that with respect to "ability to harm", it makes no difference whether the user is described with an identifier or described by behaviour (for example, use of a key or performance of an action).

What does make a difference is how often the particular equivalent of a pseudonym is used. Anonymity gives a taxonomy of pseudonyms starting from personal pseudonyms (like nicknames) that are used all the time, through various types of role pseudonyms (such as Secretary of Defense), on to "one-time-use" pseudonyms.

Only one-time-use pseudonyms can give you anonymity (within SAML, consider this as "anonymity within a set"). However, the more often a given pseudonym is used, the more is the risk to anonymity. In other words, reuse of a pseudonym allows additional potentially identifying information to be associated with the pseudonym. Over time, this will lead to an accretion that can uniquely identify the identity associated with a pseudonym.

Origin site authorities (such as authentication authorities and attribute authorities) can provide a degree of "partial anonymity" by employing one-time-use identifiers or keys (for the "holder of key" case). This anonymity is "partial" at best because the Subject is necessarily confined to the set of Subjects in a relationship with the Authority. This set may be further reduced (thus further reducing anonymity) when aggregating attributes are used that further subset the user community at the origin site. Users who truly care about anonymity must take care to disguise or avoid unusual patterns of behaviour that could serve to "de-anonymize" them over time.

I.4 Security

The following clauses discuss security considerations.

I.4.1 Background

Communication between computer-based systems is subject to a variety of threats, and these threats carry some level of associated risk. The nature of the risk depends on a host of factors, including the nature of the communications, the nature of the communicating systems, the communication mediums, the communication environment, the end-system environments, and so on.

SAML is intended to aid deployers in establishing security contexts for application-level computer-based communications within or between security domains. In this role, SAML transfers authentication data, supporting end systems' ability to protect against unauthorized usage. Communications security is directly applicable to the design of SAML. Systems security is of interest mostly in the context of SAML's threat models.

I.4.2 Scope

Some areas that impact broadly on the overall security of a system that uses SAML are explicitly outside the scope of SAML. While this Recommendation does not address these areas, they should always be considered when reviewing the security of a system. In particular, these issues are important, but currently beyond the scope of SAML:

- Initial authentication: SAML allows statements to be made about acts of authentication that have occurred, but includes no requirements or specifications for these acts of authentication. Consumers of authentication assertions should be wary of blindly trusting these assertions unless and until they know the basis on which they were made. Confidence in the assertions must never exceed the confidence that the asserting party has correctly arrived at the conclusions asserted.
- Trust Model: In many cases, the security of a SAML conversation will depend on the underlying trust model, which is typically based on a key management infrastructure (for example, PKI or secret key). For example, SOAP messages secured by means of XML Signature are secured only insofar as the keys used in the exchange can be trusted. Undetected compromised keys or revoked certificates, for example, could allow a breach of security. Even failure to require a certificate opens the door for impersonation attacks. PKI setup is not trivial and must be implemented correctly in order for layers built on top of it (such as parts of SAML) to be secure.

Suitable implementations of security protocols is necessary to maintain the security of a system, including secure random or pseudo-random number generation and secure key storage.

I.4.3 SAML threat model

The general Internet threat model described in the IETF guidelines for security considerations is the basis for the SAML threat model. We assume here that the two or more endpoints of a SAML transaction are uncompromised, but that the attacker has complete control over the communications channel.

Additionally, due to the nature of SAML as a multi-party authentication and authorization statement protocol, cases must be considered where one or more of the parties in a legitimate SAML transaction – who operate legitimately within their role for that transaction – attempt to use information gained from a previous transaction maliciously in a subsequent transaction.

The following scenarios describe possible attacks:

– **Collusion**: The secret cooperation between two or more system entities to launch an attack, for example:
  - Collusion between principal and service provider;
  - Collusion between principal and identity provider;
  - Collusion between identity provider and service provider;
  - Collusion among two or more principals;
  - Collusion between two or more service providers;
  - Collusion between two or more identity providers.

– **Denial-of-service attacks**: The prevention of authorized access to a system resource or the delaying of system operations and functions.

– **Man-in-the-middle attacks**: A form of active wiretapping attack in which the attacker intercepts and selectively modifies communicated data to masquerade as one or more of the entities involved in a communication association.

– **Replay attacks**: An attack in which a valid data transmission is maliciously or fraudulently repeated, either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of a masquerade attack.

– **Session hijacking**: A form of active wiretapping in which the attacker seizes control of a previously established communication association.

In all cases, the local mechanisms that systems will use to decide whether or not to generate assertions are out of scope. Thus, threats arising from the details of the original login at an authentication authority, for example, are out of scope as well. If an authority issues a false assertion, then the threats arising from the consumption of that assertion by downstream systems are explicitly out of scope.

The direct consequence of such a scoping is that the security of a system based on assertions as inputs is only as good as the security of the system used to generate those assertions, and of the correctness of the data and processing on which the generated assertions are based. When determining what issuers to trust, particularly in cases where the assertions will be used as inputs to authentication or authorization decisions, the risk of security compromises arising from the consumption of false but validly issued assertions is a large one. Trust policies between asserting and relying parties should always be written to include significant consideration of liability and implementations should provide an appropriate audit trail.

I.5 Security techniques

The following clauses describe security techniques and various stock technologies available for their implementation in SAML deployments.

I.5.1 Authentication

Authentication here means the ability of a party to a transaction to determine the identity of the other party in the transaction. This authentication may be in one direction or it may be bilateral.

– **Active session**: Non-persistent authentication is provided by the communications channel used to transport a SAML message. This authentication may be unilateral – from the session initiator to the receiver – or bilateral. The specific method will be determined by the communications protocol used. For instance, the use of a secure network protocol, such as TLS or the IP security protocol, provides the SAML message sender with the ability to authenticate the destination for the TCP/IP environment.

– **Message-level**: W3C XML Signature and OASIS WSS provide methods of creating a persistent "authentication" that is tightly coupled to a document. This method does not independently guarantee that the sender of the message is in fact that signer (and indeed, in many cases where intermediaries are involved, this is explicitly not the case). Any method that allows the persistent confirmation of the involvement of a uniquely resolvable entity with a given subset of an XML message is sufficient to meet this requirement.

I.5.2 Confidentiality

Confidentiality means that the contents of a message can be read only by the desired recipients and not anyone else who encounters the message.

- **In transit**: Use of a secure network protocol such as TLS or the IP Security Protocol provides transient confidentiality of a message as it is transferred between two nodes.
- **Message-level**: XML Encryption provides for the selective encryption of XML documents. This encryption method provides persistent, selective confidentiality of elements within an XML message.

I.5.3 Data integrity

Data integrity is the ability to confirm that a given message as received is unaltered from the version of the message that was sent.

- **In transit**: Use of a secure network protocol such as TLS or the IP Security Protocol may be configured to provide integrity protection for the packets transmitted via the network connection.
- **Message-level**: XML Signature provides a method of creating a persistent guarantee of the unaltered nature of a message that is tightly coupled to that message. Any method that allows the persistent confirmation of the unaltered nature of a given subset of an XML message is sufficient to meet this requirement.

I.5.4 Notes on key management

Many points in this appendix will refer to the ability of systems to provide authentication, data integrity, and confidentiality via various schemes involving digital signature and encryption. For all these schemes, the security provided by the scheme is limited based on the key management systems that are in place. Some specific limitations are detailed below.

1) **Access to the key**: It is assumed that, if key-based systems are going to be used for authentication, data integrity, and non-repudiation, security is in place to guarantee that access to a private or secret key representing a principal is not available to inappropriate parties. For example, a digital signature created with Bob's private key is only proof of Bob's involvement to the extent that Bob is the only one with access to the key. In general, access to keys should be kept to the minimum set of entities possible (particularly important for corporate or organizational keys) and should be protected with passphrases and other means. Standard security precautions (don't write down the passphrase, when you're away from a computer don't leave a window with the key accessed open, and so on) apply.

2) **Binding of identity to key**: For a key-based system to be used for authentication, there must be some trusted binding of identity to key. Verifying a digital signature on a document can determine if the document is unaltered since it was signed, and that it was actually signed by a given key. However, this does not confirm that the key used is actually the key of a specific individual appropriate for the time and purpose. Verifying the binding of a key to a party requires additional validation.

   This key-to-individual binding must be established. Common solutions include local directories that store both identifiers and key – which is simple to understand but difficult to maintain – or the use of certificates. Using certificates can provide a scalable means to associate a key with an identity, but requires mechanisms to manage the certificate lifecycle and changes to the status of the binding (e.g., an employee leaves and no longer has a corporate identity). One common approach is to use a public key infrastructure (PKI).

   In this case a set of trusted root certifying authorities (CAs) are identified for each consumer of signatures – answering the question "Whom do I trust to make statements of identity-to-key binding?" Verification of a signature then becomes a process of first verifying the signature (to determine that the signature was done by the key in question and that the message has not changed) and then validating the certificate chain (to determine that the key is bound to the right identity) and validating that the binding is still appropriate. Validating the binding requires steps to be taken to ensure that the binding is currently valid – a certificate typically has a "lifetime" built into it, but if a key is compromised during the life of the certificate then the key-to-identity binding contained in the certificate becomes invalid while the certificate is still valid on its face. Also, certificates often depend on associations that may end before their lifetime expires (for example, certificates that should become invalid when someone changes employers, etc.). A proper key management system is thus quite strong but very complex. Verifying a signature ends up being a process of verifying the document-to-key binding, then verifying the key-to-identity binding, as well as the current validity of the key and certificate.

I.5.5 TLS cipher suites

The use of HTTP over SSL 3.0 (see Appendix IV) or TLS 1.0, or use of URLs with the HTTPS URL scheme, is strongly recommended at many places in this Recommendation.

Unless otherwise specified, in any SAML binding's use of SSL 3.0 or TLS 1.0, servers must authenticate to clients using an X.509 v3 certificate. The client must establish server identity based on contents of the certificate (typically through examination of the certificate's subject DN field).

SSL/TLS can be configured to use many different cipher suites, not all of which are adequate to provide "best practices" security. A cipher suite combines four kinds of security features, and is given a name in [SSL]. Before data flows over a SSL connection, both ends attempt to negotiate a cipher suite. This lets them establish an appropriate quality of protection for their communications, within the constraints of the particular mechanism combinations which are available. The features associated with a cipher suite are:

SSL defines many key exchange algorithms. Some mechanisms provide for server authentication. However, anonymous key exchange mechanisms are also supported. (Anonymous key exchange algorithms are subject to "man-in-the-middle" attacks, and are not recommended in the SAML context.) The "RSA" authenticated key exchange algorithm is currently the most interoperable algorithm (the RSA algorithm patent has expired). Another important key exchange algorithm is the authenticated Diffie-Hellman "DHE_DSS" key exchange, which has no patent-related implementation constraints.

Whether the key exchange algorithm is freely exportable from the United States of America. Exportable algorithms must use short (512-bit) public keys for key exchange and short (40-bit) symmetric keys for encryption. Keys of these lengths have been successfully attacked, and their use is not recommended.

The fastest encryption algorithm option is the RC4 stream cipher; DES and variants (DES40, 3DES-EDE) as well as AES are also supported in "cipher block chaining" (CBC) mode. Other modes are also supported, refer to the TLS documentation.

Null encryption is an option in some cipher suites. Null encryption performs no encryption; in such cases SSL/TLS is used only to authenticate and provide integrity protection. Cipher suites with null encryption do not provide confidentiality, and must not be used in cases where confidentiality is a requirement and is not obtained by means other than SSL/TLS.

The digest algorithm used for the message authentication code. FCC has recently recommended to use SHA-256 and IETF has decided to follow.

I.6 General SAML security considerations

The following clauses analyse the security risks in using and implementing SAML and describe countermeasures to mitigate the risks.

I.6.1 SAML assertions

At the level of the SAML assertion itself, there is little to be said about security concerns – most concerns arise during communications in the request/response protocol, or during the attempt to use SAML by means of one of the bindings. The consumer is, of course, always expected to honour the validity interval of the assertion and any `<OneTimeUse>` elements that are present in the assertion.

However, one issue at the assertion level bears analysis: an assertion, once issued, is out of the control of the issuer. This fact has a number of ramifications. For example, the issuer has no control over how long the assertion will be persisted in the systems of the consumer; nor does the issuer have control over the parties with whom the consumer will share the assertion information. These concerns are over and above concerns about a malicious attacker who can see the contents of assertions that pass over the wire unencrypted (or insufficiently encrypted).

While efforts have been made to address many of these issues within the SAML Recommendation, nothing contained in this Recommendation will erase the requirement for careful consideration of what to put in an assertion. At all times, issuers should consider the possible consequences if the information in the assertion is stored on a remote site, where it can be directly misused, or exposed to potential hackers, or possibly stored for more creatively fraudulent uses. Issuers should also consider the possibility that the information in the assertion could be shared with other parties, or even made public, either intentionally or inadvertently.

I.6.2 SAML protocol

This clause describes security considerations for the SAML request-response protocol itself, apart from any threats arising from use of a particular protocol binding.

– **Denial of service**

The SAML protocol is susceptible to a denial of service (DoS) attack. Handling a SAML request is potentially a very expensive operation, including parsing the request message (typically involving construction of a DOM tree), database/assertion store lookup (potentially on an unindexed key), construction of a response message, and potentially one or more digital signature operations. Thus, the effort required by an attacker generating requests is much lower than the effort needed to handle those requests.

1) **Requiring client authentication at a lower level**

Requiring clients to authenticate at some level below the SAML protocol level (for example, using the SOAP over HTTP binding, with HTTP over TLS/SSL, and with a requirement for client-side certificates that have a trusted Certificate Authority at their root) will provide traceability in the case of a DoS attack.

If the authentication is used only to provide traceability, then this does not in itself prevent the attack from occurring, but does function as a deterrent.

If the authentication is coupled with some access control system, then DoS attacks from non-insiders are effectively blocked. (It is possible that overloading the client-authentication scheme could still function as a denial-of-service attack on the SAML service, but that this attack needs to be dealt with in the context of the client authentication scheme chosen.)

Whatever system of client authentication is used, it should provide the ability to resolve a unique originator for each request, and should not be subject to forgery. (For example, in the traceability-only case, logging the IP address is insufficient since this information can easily be spoofed.)

2) **Requiring signed requests**

Requiring a signed request also lessens the order of the asymmetry between the work done by requester and responder. The additional work required of the responder to verify the signature is a relatively small percentage of the total work required of the responder, while the process of calculating the digital signature represents a relatively large amount of work for the requester. Narrowing this asymmetry decreases the risk associated with a DoS attack.

However, that an attacker can theoretically capture a signed message and then replay it continually, getting around this requirement. This situation can be avoided by requiring the use of the XML Signature element `<ds:SignatureProperties>` containing a timestamp; the timestamp can then be used to determine if the signature is recent. In this case, the narrower the window of time after issue that a signature is treated as valid, the higher security you have against replay denial of service attacks.

3) **Restricting access to the interaction URL**

Limiting the ability to issue a request to a SAML service at a very low level to a set of known parties drastically reduces the risk of a DoS attack. In this case, only attacks originating from within the finite set of known parties are possible, greatly decreasing exposure both to potentially malicious clients and to DoS attacks using compromised machines as zombies.

There are many possible methods of limiting access, such as placing the SAML responder inside a secured intranet and implementing access rules at the router level.

I.7 SAML bindings security considerations

The security considerations in the design of the SAML request-response protocol depend to a large extent on the particular protocol binding that is used. The supported bindings are the SOAP binding, Reverse SOAP Binding (PAOS), HTTP Redirect binding, HTTP Redirect/POST binding and HTTP Artifact binding and SAML URI bindings.

I.7.1 SAML SOAP binding

Since the SAML SOAP binding requires no authentication and has no requirements for either in-transit confidentiality or message integrity, it is open to a wide variety of common attacks. General considerations are discussed separately from considerations related to the SOAP-over-HTTP case.

1) **Eavesdropping**

Threat: Since there is no in-transit confidentiality requirement, it is possible that an eavesdropping party could acquire both the SOAP message containing a request and the SOAP message containing the corresponding response. This acquisition exposes both the nature of the request and the details of the response, possibly including one or more assertions.

Exposure of the details of the request will in some cases weaken the security of the requesting party by revealing details of what kinds of assertions it requires, or from whom those assertions are requested. For example, if an eavesdropper can determine that site *X* is frequently requesting authentication assertions with a given confirmation method from site *Y*, he may be able to use this information to aid in the compromise of site *X*.

Similarly, eavesdropping on a series of authorization queries could create a "map" of resources that are under the control of a given authorization authority.

Additionally, in some cases exposure of the request itself could constitute a violation of privacy. For example, eavesdropping on a query and its response may expose that a given user is active on the querying site, which could be information that should not be divulged in cases such as medical information sites, political sites, and so on. Also the details of any assertions carried in the response may be information that should be kept confidential. This is particularly true for responses containing attribute assertions; if these attributes represent information that should not be available to entities not party to the transaction (credit ratings, medical attributes, and so on), then the risk from eavesdropping is high.

Countermeasures: In cases where any of these risks is a concern, the countermeasure for eavesdropping attacks is to provide some form of in-transit message confidentiality. For SOAP messages, this confidentiality can be enforced either at the SOAP level or at the SOAP transport level (or some level below it).

Adding in-transit confidentiality at the SOAP level means constructing the SOAP message such that, regardless of SOAP transport, no one but the intended party will be able to access the message. The general solution to this problem is likely to be XML Encryption. This Recommendation allows encryption of the SOAP message itself, which eliminates the risk of eavesdropping unless the key used in the encryption has been compromised. Alternatively, deployers can depend on the SOAP transport layer, or a layer beneath it, to provide in-transit confidentiality.

The details of how to provide this confidentiality depend on the specific SOAP transport chosen. Using HTTP over TLS/SSL is one method. Other transports will necessitate other in-transit confidentiality techniques; for example, an SMTP transport might use S/MIME.

In some cases, a layer beneath the SOAP transport might provide the required in-transit confidentiality. For example, if the request-response interaction is carried out over an IPSec tunnel, then adequate in-transit confidentiality may be provided by the tunnel itself.

2) Replay

Threat: There is little vulnerability to replay attacks at the level of the SOAP binding. Replay is more of an issue in the various profiles. The primary concern about replay at the SOAP binding level is the potential for use of replay as a denial-of-service attack method.

Countermeasures: In general, the best way to prevent replay attacks is to prevent the message capture in the first place. Some of the transport-level schemes used to provide in-transit confidentiality will accomplish this goal. For example, if the SAML request-response conversation occurs over SOAP on HTTP/TLS, third parties are prevented from capturing the messages.

Since the potential replayer does not need to understand the message to replay it, schemes such as XML Encryption do not provide protection against replay. If an attacker can capture a SAML request that has been signed by the requester and encrypted to the responder, then the attacker can replay that request at any time without needing to be able to undo the encryption. The SAML request includes information about the issue time of the request, allowing a determination about whether replay is occurring. Alternatively, the unique key of the request (its `ID`) can be used to determine if this is a replay request or not.

Additional threats from the replay attack include cases where a "charge per request" model is in place. Replay could be used to run up large charges on a given account.

Similarly, models where a client is allocated (or purchases) a fixed number of interactions with a system, the replay attack could exhaust these uses unless the issuer is careful to keep track of the unique key of each request.

3) Message insertion

Threat: A fabricated request or response is inserted into the message stream. A false response such as a spurious "yes" reply to an authorization decision query or the return of false attribute information in response to an attribute query may result in inappropriate receiver action.

Countermeasures: The ability to insert a request is not a threat at the SOAP binding level. The threat of inserting a false response can be a denial of service attack, for example returning SOAP Faults for responses, but this attack would become quickly obvious. The more subtle attack of returning fabricated responses is addressed in the SAML protocol, appropriate since according to the SOAP Binding definition each SOAP response must contain a single SAML protocol response unless it contains a fault. The SAML Protocol addresses this with two mechanisms, correlation of responses to requests using the required InResponseTo attribute, making an attack harder since requests must be intercepted to generate responses, and through the support origin authentication, either via signed SAML responses or through a secured transport connection such as SSL/TLS.

4) **Message deletion**

Threat: The message deletion attack would either prevent a request from reaching a responder, or would prevent the response from reaching the requester.

Countermeasures: In either case, the SOAP binding does not address this threat. In general, correlation of request and response messages may deter such an attack, for example use of the InResponseTo attribute in the **StatusResponseType**.

5) **Message modification**

Threat: Message modification is a threat to the SOAP binding in both directions.

Modification of the request to alter the details of the request can result in significantly different results being returned, which in turn can be used by a clever attacker to compromise systems depending on the assertions returned. For example, altering the list of requested attributes in the <Attribute> elements could produce results leading to compromise or rejection of the request by the responder.

Modification of the request to alter the apparent issuer of the request could result in denial of service or incorrect routing of the response. This alteration would need to occur below the SAML level and is thus out of scope.

Modification of the response to alter the details of the assertions therein could result in vast degrees of compromise. The simple examples of altering details of an authentication or an authorization decision could lead to very serious security breaches.

Countermeasures: In order to address these potential threats, a system that guarantees in-transit message integrity must be used. The SAML protocol and the SOAP binding neither require nor forbid the deployment of systems that guarantee in-transit message integrity, but due to this large threat, it is highly recommended that such a system be used. At the SOAP binding level, this can be accomplished by digitally signing requests and responses with a system such as XML Signature.

If messages are digitally signed then the recipient has a guarantee that the message has not been altered in transit, unless the key used has been compromised.

The goal of in-transit message integrity can also be accomplished at a lower level by using a SOAP transport that provides the property of guaranteed integrity, or is based on a protocol that provides such a property. SOAP over HTTP over TLS/SSL is a transport that would provide such a guarantee.

Encryption alone does not provide this protection, as even if the intercepted message could not be altered *per se*, it could be replaced with a newly created one.

6) **Man-in-the-middle**

Threat: The SOAP binding is susceptible to man-in-the-middle (MITM) attacks. In order to prevent malicious entities from operating as a man in the middle (with all the perils discussed in both the eavesdropping and message modification sections), some sort of bilateral authentication is required.

Countermeasures: A bilateral authentication system would allow both parties to determine that what they are seeing in a conversation actually came from the other party to the conversation.

At the SOAP binding level, this goal could also be accomplished by digitally signing both requests and responses. This method does not prevent an eavesdropper from sitting in the middle and forwarding both ways, but he is prevented from altering the conversation in any way without being detected.

Since many applications of SOAP do not use sessions, this sort of authentication of author (as opposed to authentication of sender) may need to be combined with information from the transport layer to confirm that the sender and the author are the same party in order to prevent a weaker form of "MITM as eavesdropper".

Another implementation would depend on a SOAP transport that provides, or is implemented on a lower layer that provides, bilateral authentication. The example of this is again SOAP over HTTP over TLS/SSL with both server- and client-side certificates required.

Additionally, the validity interval of the assertions returned functions as an adjustment on the degree of risk from MITM attacks. The shorter the valid window of the assertion, the less damage can be done if it is intercepted.

7) **Use of SOAP over HTTP**

Since the SOAP binding requires that conformant applications support HTTP over TLS/SSL with a number of different bilateral authentication methods such as Basic over server-side SSL and certificate-backed authentication over server-side SSL, these methods are always available to mitigate threats in cases where other lower-level systems are not available and the above listed attacks are considered significant threats.

This does not mean that use of HTTP over TLS with some form of bilateral authentication is mandatory. If an acceptable level of protection from the various risks can be arrived at through other means (for example, by an IPSec tunnel), full TLS with certificates is not required. However, in the majority of cases for SOAP over HTTP, using HTTP over TLS with bilateral authentication will be the appropriate choice.

The HTTP authentication RFC (IETF RFC 2617) describes possible attacks in the HTTP environment when basic or message-digest authentication schemes are used.

Note however, that the use of transport-level security (such as the SSL or TLS protocols under HTTP) only provides confidentiality and/or integrity and/or authentication for "one hop". For models where there may be intermediaries, or the assertions in question need to live over more than one hop, the use of HTTP with TLS/SSL does not provide adequate security.

I.7.2 Web browser single sign on (SSO) profiles

User authentication at the source site is explicitly out of scope, as are issues related to this source site authentication. The key notion is that the source system entity must be able to ascertain that the authenticated client system entity that it is interacting with is the same as the one in the next interaction step. One way to accomplish this is for these initial steps to be performed using TLS as a session layer underneath the protocol being used for this initial interaction (likely HTTP).

I.7.2.1 SSO profile

1) **Eavesdropping**

Threat: The possibility of eavesdropping exists in all web browser cases.

Countermeasures: In cases where confidentiality is required (bearing in mind that any assertion that is not sent securely, along with the requests associated with it, is available to the malicious eavesdropper), HTTP traffic needs to take place over a transport that ensures confidentiality. HTTP over TLS/SSL and the IP Security Protocol meet this requirement.

2) **Theft of the user authentication information**

Threat: In the case where the subject authenticates to the source site by revealing reusable authentication information, for example, in the form of a password, theft of the authentication information will enable an adversary to impersonate the subject.

Countermeasures: In order to avoid this problem, the connection between the subject's browser and the source site must implement a confidentiality safeguard. In addition, steps must be taken by either the subject or the destination site to ensure that the source site is genuinely the expected and trusted source site before revealing the authentication information. Using HTTP over TLS can be used to address this concern.

3) **Theft of the bearer token**

Threat: In the case where the authentication assertion contains the assertion bearer's authentication protocol identifier, theft of the artifact will enable an adversary to impersonate the subject.

Countermeasures: Each of the following methods decreases the likelihood of this happening:

The destination site implements a confidentiality safeguard on its connection with the subject's browser.

The subject or destination site ensures (out of band) that the source site implements a confidentiality safeguard on its connection with the subject's browser.

The destination site verifies that the subject's browser was directly redirected by a source site that directly authenticated the subject.

The source site refuses to respond to more than one request for an assertion corresponding to the same assertion ID.

If the assertion contains a condition element of type **AudienceRestrictionType** that identifies a specific domain, then the destination site verifies that it is a member of that domain.

The connection between the destination site and the source site, over which the assertion ID is passed, is implemented with a confidentiality safeguard.

The destination site, in its communication with the source site, over which the assertion ID is passed, must verify that the source site is genuinely the expected and trusted source site.

4) **Replay**

The possibility of a replay attack exists for this set of profiles. A replay attack can be used either to attempt to deny service or to retrieve information fraudulently. The specific countermeasures depend on which specific binding is used and are discussed above.

5) **Message insertion**

Message insertion attacks are discussed in I.7.1.

6) **Message deletion**

Threat: Deleting a message during any step of the interactions between the browser, SAML assertion issuer, and SAML assertion consumer will cause the interaction to fail. It results in a denial of some service but does not increase the exposure of any information.

Countermeasures: Use of an integrity protected transport channel addresses the threat of message deletion when no intermediaries are present.

7) **Message modification**

Threat: The possibility of alteration of the messages in the stream exists for this set of profiles. Some potential undesirable results are as follows:

Alteration of the initial request can result in rejection at the SAML issuer, or creation of an artifact targeted at a different resource than the one requested.

Alteration of the artifact can result in denial of service at the SAML consumer.

Alteration of the assertions themselves while in transit could result in all kinds of bad results (if they are unsigned) or denial of service (if they are signed and the consumer rejects them).

Countermeasures: To avoid message modification, the traffic needs to be transported by means of a system that guarantees message integrity from endpoint to endpoint.

For the web browser-based profiles, the recommended method of providing message integrity in transit is the use of HTTP over TLS/SSL with a cipher suite that provides data integrity checking.

8) **Man-in-the-middle**

Threat: Man-in-the-middle attacks are particularly pernicious for this set of profiles. The MITM can relay requests, capture the returned assertion (or artifact), and relay back a false one. Then the original user cannot access the resource in question, but the MITM can do so using the captured resource.

Countermeasures: Preventing this threat requires a number of countermeasures. First, using a system that provides strong bilateral authentication will make it much more difficult for a MITM to insert himself into the conversation.

However the possibility still exists of a MITM who is purely acting as a bidirectional port forwarder, and eavesdropping on the information with the intent to capture the returned assertion or handler (and possibly alter the final return to the requester). Putting a confidentiality system in place will prevent eavesdropping. Putting a data integrity system in place will prevent alteration of the message during port forwarding.

For this set of profiles, all the requirements of strong bilateral session authentication, confidentiality, and data integrity can be met by the use of HTTP over TLS/SSL if the TLS/SSL layer uses an appropriate cipher suite (strong enough encryption to provide confidentiality, and supporting data integrity) and requires X.509 v3 certificates for authentication.

9) **Impersonation without reauthentication**

Threat: Rogue user attempts to impersonate currently logged-in legitimate Principal and thereby gain access to protected resources.

Once a Principal is successfully logged into an identity provider, subsequent `<AuthnRequest>` messages from different service providers concerning that Principal will not necessarily cause the Principal to be reauthenticated. Principals must, however, be authenticated unless the identity provider can determine that an `<AuthnRequest>` is associated not only with the Principal's identity, but also with a validly authenticated identity provider session for that Principal.

Countermeasures: In implementations where this threat is a concern, identity providers must maintain state information concerning active sessions, and must validate the correspondence between an `<AuthnRequest>` and an active session before issuing a `<Response>` without first authenticating the Principal. Cookies posted by identity providers may be used to support this validation process, though Liberty does not mandate a cookie-based approach.

I.7.2.2 Enhanced client and proxy profile

1) **Man-in-the-middle**

   Threat: Intercept AuthnRequest and Response SOAP messages, allowing subsequent Principal impersonation.

   A spurious system entity can interject itself as a man-in-the-middle (MITM) between the enhanced client and a legitimate service provider, where it acts in the service provider role in interactions with the enhanced client and in the enhanced client role in interactions with the legitimate service provider. In this way, as a first step, the MITM is able to intercept the service provider's AuthnRequest and substitute any URL of its choosing for the responseConsumerServiceURL value in the PAOS header block before forwarding the AuthnRequest on to the enhanced client. Typically, the MITM will insert a URL value that points back to itself. Then, if the enhanced client subsequently receives a Response from the identity provider and subsequently sends the contained Response to the responseConsumerServiceURL received from the MITM, the MITM will be able to masquerade as the Principal at the legitimate service provider.

   Countermeasures: The identity provider specifies to the enhanced client the address to which the enhanced client must send the Response. The responseConsumerServiceURL in the PAOS header is only used for error responses from the enhanced client – as specified in the profile.

2) **Denial of service**

   Threat: Change an AuthnRequest SOAP request so that it cannot be processed, such as by changing the PAOS header block service attribute value to an unknown value or by changing the ECP header block ProviderID or IDPList to cause the request to fail.

   Countermeasures: Provide integrity protection for the SOAP message, by using SOAP Message Security or SSL/TLS.

I.7.2.3 Identity provider discovery profile

Threat: Cookie poisoning attack, where parameters within the cookie are modified, to cause discovery of a fraudulent identity provider for example.

Countermeasures: The specific mechanism of using a common domain limits the feasibility of this threat.

I.7.2.4 Single logout profile

Threat: Passive attacker can collect a Principal's name identifier.

During the initial steps, a passive attacker can collect the <LogoutRequest> information when it is issued in the redirect. Exposing these data poses a privacy threat.

Countermeasures: All exchanges should be conducted over a secure transport such as SSL or TLS.

Threat: Unsigned <LogoutRequest> message.

An Unsigned <LogoutRequest> could be injected by a spurious system entity thus denying service to the Principal. Assuming that the NameID can be deduced or derived, then it is conceivable that the user agent could be directed to deliver a fabricated <LogoutRequest> message.

Countermeasures: Sign the <LogoutRequest> message. The identity provider can also verify the identity of a Principal in the absence of a signed request.

I.7.2.5 Name Identifier management profiles

Threat: Allow system entities to correlate information or otherwise inappropriately expose identity information, harming privacy.

Countermeasures: IdP must take care to use different name identifiers with different service providers for same principal. The IdP should encrypt the name identifier it returns to the service provider, allowing subsequent interactions to use an opaque identifier.

I.7.2.6 Attribute profiles

Threats related to bindings associated with attribute profiles are discussed above. No additional profile-specific threats are known.

Appendix II

Registration of MIME media type application/samlassertion+xml

This appendix contains the registration of SAML Application/Assertion MIME media type.

MIME media type name

– `application`

MIME subtype name

– `samlassertion+xml`

Required parameters

– None.

Optional parameters

– `charset`

– Same as `charset` parameter of `application/xml` as in IETF RFC 3923.

Encoding considerations

– Same as for application/xml as in IETF RFC 3923.

Security considerations

The `samlassertion+xml`-typed objects do not contain executable content. However, SAML assertions are XML-based objects. As such, they have all of the general security considerations presented in IETF RFC 3923 clause 10, as well as additional ones, since they are explicit security objects. For example, `samlassertion+xml`-typed objects will often contain data that may identify or pertain to a natural person, and may be used as a basis for sessions and access control decisions.

To counter potential issues, `samlassertion+xml`-typed objects contain data that should be signed appropriately by the sender. Any such signature must be verified by the recipient of the data – both as a valid signature, and as being the signature of the sender. Issuers of `samlassertion+xml`-typed objects containing SAML assertions may also encrypt all, or portions of, the assertions.

In addition, SAML profiles and protocol bindings specify use of secure channels as appropriate.

SAML Version 2 (this Recommendation) incorporates various privacy-protection techniques in its design. For example: opaque handles, specific to interactions between specific system entities, may be assigned to subjects. The handles are mappable to wider-context identifiers (e.g., email addresses, account identifiers, etc.) by only the specific parties.

Interoperability considerations

SAML assertions are explicitly versioned. Relying parties should ensure that they observe assertion version information and behave accordingly.

Published specification

SAML Version 2 (this Recommendation) explicitly specifies use of the `application/samlassertion+xml` MIME media type. However, it is conceivable that non-SAML assertions (i.e., SAMLv1 and/or SAMLv1.1) might in practice be conveyed using SAML bindings.

Applications which use this media type

Potentially any application implementing SAML, as well as those applications implementing specifications based on SAML.

Additional information

Magic number(s)

In general, the same as for application/xml. In particular, the XML root element of the returned object will have a namespace-qualified name with:

- a local name of: `Assertion`

- a namespace URI of: one of the version-specific SAML assertion XML namespace URIs, as defined by the appropriate version-specific SAML "core" Recommendation.

With SAML specifically, the root element of the returned object may be either `<saml:Assertion>` or `<saml:EncryptedAssertion>`, where "saml" represents any XML namespace prefix that maps to the SAML assertion namespace URI:

`urn:oasis:names:tc:SAML:2.0:assertion`

File extension(s)

None

Macintosh file type code(s)

None

Person & email address to contact for further information

This registration was made on behalf of the OASIS Security Services Technical Committee (SSTC).

Intended usage

Common

Appendix III

Registration of MIME media type application/samlmetadata+xml

This appendix defines a MIME media type - `application/samlmetadata+xml` - for use with the XML serialization of Security Assertion Markup Language metadata.

1) **MIME media type name**

   – `application`

2) **MIME subtype name**

   – `samlmetadata+xml`

3) **Required parameters**

   – None.

4) **Optional parameters**

   – `charset`

   – Same as `charset` parameter of application/xml (see IETF RFC 3023).

5) **Encoding considerations**

   – Same as for `application/xml` in IETF RFC 3023.

6) **Security considerations**

   The `samlmetadata+xml` typed objects do not contain executable content. However, these objects are XML-based, and thus they have all of the general security considerations presented in IETF RFC 3023, clause 10.

   To counter potential issues, the publisher may sign `samlmetadata+xml` typed objects. Any such signature should be verified by the recipient of the data – both as a valid signature, and as being the signature of the publisher.

7) **Interoperability considerations**

   SAML metadata explicitly supports identifying the protocols and versions supported by the identified entities. For example, an identity provider entity can be denoted as supporting SAML v2.0 and other protocols if they are unambiguously identifiable via URI. This protocol support information is conveyed via the `protocolSupportEnumeration` attribute of metadata objects of the **RoleDescriptorType**.

8) **Published Recommendation**

   SAML Metadata explicitly specifies use of the `application/samlmetadata+xml` MIME media type.

Applications which use this media type:

Potentially any application implementing SAML v2.0, as well as those applications implementing specifications based on SAML.

9) Additional information

1) Magic number(s)

In general, the same as for `application/xml` in IETF RFC 3023. In particular, the XML root element of the returned object will have a namespace-qualified name with:

– a local name of: `EntityDescriptor`, or `AffiliationDescriptor`, or `EntitiesDescriptor`;

– a namespace URI of: `urn:oasis:names:tc:SAML:2.0:metadata` (the SAMLv2.0 metadata namespace).

10) File extension(s)

None.

11) Macintosh file type code(s)

None.

12) Person and email address to contact for further information

This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC).

13) Intended usage

Common.

Appendix IV

Use of SSL

Some implementation of SAML might support the use of SSL 3.0 in addition or as an alternative to TLS 1.0. Implementations that use SSL 3.0 should ensure that the overall security of the implementation is consistent with the restrictions placed on the use of ciphers in TLS. For example, the requirement to use cipher suite TLS_RSA_WITH_3DES_EDE_CBC_SHA translate to the use of SSL_RSA_WITH_3DES_EDE_CBC_SHA cipher suite. FIPS SSL-capable implementations use the FIPS cipher suite corresponding to the SSL SSL_RSA_WITH_3DES_EDE_CBC_SHA cipher suite.

TLS implementations of the Web SSO profile of SAML that support TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher suite will use the SSL_RSA_WITH_3DES_EDE_CBC_SHA cipher suite.

Appendix V

SAML Schema Authentication Context

This appendix contains SAML Authentication Context Schema for SSL certificate (sslcert).

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient"
  finalDefault="extension"
  blockDefault="substitution"
  version="2.0">

  <xs:redefine schemaLocation="saml-schema-authn-context-types-2.0.xsd">

    <xs:annotation>
      <xs:documentation>
```

```
        Class identifier: urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient
        Document identifier: saml-schema-authn-context-sslcert-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New authentication_u99 context class schema for SAML V2.0.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType name="AuthnContextDeclarationBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnContextDeclarationBaseType">
          <xs:sequence>
            <xs:element ref="Identification" minOccurs="0"/>
            <xs:element ref="TechnicalProtection" minOccurs="0"/>
            <xs:element ref="OperationalProtection" minOccurs="0"/>
            <xs:element ref="AuthnMethod"/>
            <xs:element ref="GoverningAgreements" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="ID" type="xs:ID" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthnMethodBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthnMethodBaseType">
          <xs:sequence>
            <xs:element ref="PrincipalAuthenticationMechanism"/>
            <xs:element ref="Authenticator"/>
            <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PrincipalAuthenticationMechanismType">
      <xs:complexContent>
        <xs:restriction base="PrincipalAuthenticationMechanismType">
          <xs:sequence>
            <xs:element ref="RestrictedPassword"/>
          </xs:sequence>
          <xs:attribute name="preauth" type="xs:integer" use="optional"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorBaseType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorBaseType">
          <xs:sequence>
            <xs:element ref="DigSig"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PublicKeyType">
      <xs:complexContent>
        <xs:restriction base="PublicKeyType">
          <xs:attribute name="keyValidation" type="xs:anyURI"
fixed="urn:oasis:names:tc:SAML:2.0:ac:classes:X509"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AuthenticatorTransportProtocolType">
      <xs:complexContent>
        <xs:restriction base="AuthenticatorTransportProtocolType">
```

```
          <xs:sequence>
            <xs:choice>
              <xs:element ref="SSL"/>
              <xs:element ref="WTLS"/>
            </xs:choice>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>

  </xs:redefine>

</xs:schema>
```

Appendix VI

Authentication Context types XML Schema

This appendix lists the complete Authentication Context types XML Schema and the Authentication Context XML schema itself, used for the validation of individual generalized declarations. The types schema has no target namespace itself, and is then included in Appendix V.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  version="2.0">

  <xs:annotation>
    <xs:documentation>
      Document identifier: saml-schema-authn-context-types-2.0
      Location: http://docs.oasis-open.org/security/saml/v2.0/
      Revision history:
          V2.0 (March, 2005):
          New core authentication context schema types for SAML V2.0.
    </xs:documentation>
  </xs:annotation>

  <xs:element name="AuthenticationContextDeclaration"
type="AuthnContextDeclarationBaseType">
    <xs:annotation>
      <xs:documentation>
        A particular assertion on an identity
        provider's part with respect to the authentication
        context associated with an authentication assertion.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="Identification" type="IdentificationType">
    <xs:annotation>
      <xs:documentation>
        Refers to those characteristics that describe the
        processes and mechanisms
        the Authentication Authority uses to initially create
        an association between a Principal
        and the identity (or name) by which the Principal will
        be known
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="PhysicalVerification">
    <xs:annotation>
      <xs:documentation>
        This element indicates that identification has been
        performed in a physical
        face-to-face meeting with the principal and not in an
        online manner.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:attribute name="credentialLevel">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="primary"/>
            <xs:enumeration value="secondary"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="WrittenConsent" type="ExtensionOnlyType"/>

  <xs:element name="TechnicalProtection" type="TechnicalProtectionBaseType">
    <xs:annotation>
      <xs:documentation>
        Refers to those characterstics that describe how the
        'secret' (the knowledge or possession
        of which allows the Principal to authenticate to the
        Authentication Authority) is kept secure
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="SecretKeyProtection" type="SecretKeyProtectionType">
    <xs:annotation>
      <xs:documentation>
        This element indicates the types and strengths of
        facilities
        of a UA used to protect a shared secret key from
        unauthorized access and/or use.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="PrivateKeyProtection" type="PrivateKeyProtectionType">
    <xs:annotation>
      <xs:documentation>
        This element indicates the types and strengths of
        facilities
        of a UA used to protect a private key from
        unauthorized access and/or use.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="KeyActivation" type="KeyActivationType">
    <xs:annotation>
      <xs:documentation>The actions that must be performed
        before the private key can be used. </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="KeySharing" type="KeySharingType">
    <xs:annotation>
      <xs:documentation>Whether or not the private key is shared
        with the certificate authority.</xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="KeyStorage" type="KeyStorageType">
    <xs:annotation>
      <xs:documentation>
```

```
        In which medium is the key stored.
        memory - the key is stored in memory.
        smartcard - the key is stored in a smartcard.
        token - the key is stored in a hardware token.
        MobileDevice - the key is stored in a mobile device.
        MobileAuthCard - the key is stored in a mobile
        authentication card.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="SubscriberLineNumber" type="ExtensionOnlyType"/>
  <xs:element name="UserSuffix" type="ExtensionOnlyType"/>

  <xs:element name="Password" type="PasswordType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that a password (or passphrase)
        has been used to
        authenticate the Principal to a remote system.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="ActivationPin" type="ActivationPinType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that a Pin (Personal
        Identification Number) has been used to authenticate the Principal
        to some local system in order to activate a key.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="Token" type="TokenType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that a hardware or software
        token is used
        as a method of identifying the Principal.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="TimeSyncToken" type="TimeSyncTokenType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that a time synchronization
        token is used to identify the Principal. hardware -
        the time synchonization
        token has been implemented in hardware. software - the
        time synchronization
        token has been implemented in software. SeedLength -
        the length, in bits, of the
        random seed used in the time synchronization token.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="Smartcard" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that a smartcard is used to
        identify the Principal.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="Length" type="LengthType">
    <xs:annotation>
      <xs:documentation>
```

```
        This element indicates the minimum and/or maximum
        ASCII length of the password which is enforced (by the UA or the
        IdP). In other words, this is the minimum and/or maximum number of
        ASCII characters required to represent a valid password.
        min - the minimum number of ASCII characters required
        in a valid password, as enforced by the UA or the IdP.
        max - the maximum number of ASCII characters required
        in a valid password, as enforced by the UA or the IdP.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="ActivationLimit" type="ActivationLimitType">
    <xs:annotation>
      <xs:documentation>
        This element indicates the length of time for which an
        PIN-based authentication is valid.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="Generation">
    <xs:annotation>
      <xs:documentation>
        Indicates whether the password was chosen by the
        Principal or auto-supplied by the Authentication Authority.
        principalchosen - the Principal is allowed to choose
        the value of the password. This is true even if
        the initial password is chosen at random by the UA or
        the IdP and the Principal is then free to change
        the password.
        automatic - the password is chosen by the UA or the
        IdP to be cryptographically strong in some sense,
        or to satisfy certain password rules, and that the
        Principal is not free to change it or to choose a new password.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType>
      <xs:attribute name="mechanism" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="principalchosen"/>
            <xs:enumeration value="automatic"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="AuthnMethod" type="AuthnMethodBaseType">
    <xs:annotation>
      <xs:documentation>
        Refers to those characteristics that define the
        mechanisms by which the Principal authenticates to the
        Authentication Authority.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="PrincipalAuthenticationMechanism"
type="PrincipalAuthenticationMechanismType">
    <xs:annotation>
      <xs:documentation>
        The method that a Principal employs to perform
        authentication to local system components.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="Authenticator" type="AuthenticatorBaseType">
```

```
    <xs:annotation>
      <xs:documentation>
        The method applied to validate a principal's
        authentication across a network
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="ComplexAuthenticator" type="ComplexAuthenticatorType">
    <xs:annotation>
      <xs:documentation>
        Supports Authenticators with nested combinations of
        additional complexity.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="PreviousSession" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        Indicates that the Principal has been strongly
        authenticated in a previous session during which the IdP has set a
        cookie in the UA. During the present session the Principal has only
        been authenticated by the UA returning the cookie to the IdP.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="ResumeSession" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        Rather like PreviousSession but using stronger
        security. A secret that was established in a previous session with
        the Authentication Authority has been cached by the local system
        and is now re-used (e.g. a Master Secret is used to derive new
        session keys in TLS, WTLS).
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="ZeroKnowledge" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that the Principal has been
        authenticated by a zero knowledge technique as specified in ISO/IEC
        9798-5.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="SharedSecretChallengeResponse"
type="SharedSecretChallengeResponseType"/>

  <xs:complexType name="SharedSecretChallengeResponseType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that the Principal has been
        authenticated by a challenge-response protocol utilizing shared
        secret keys and symmetric cryptography.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="method" type="xs:anyURI" use="optional"/>
  </xs:complexType>

  <xs:element name="DigSig" type="PublicKeyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that the Principal has been
        authenticated by a mechanism which involves the Principal computing
        a digital signature over at least challenge data provided
        by the IdP.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="AsymmetricDecryption" type="PublicKeyType">
    <xs:annotation>
      <xs:documentation>
        The local system has a private key but it is used
        in decryption mode, rather than signature mode. For example, the
        Authentication Authority generates a secret and encrypts it using
        the local system's public key: the local system then proves it has
        decrypted the secret.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="AsymmetricKeyAgreement" type="PublicKeyType">
    <xs:annotation>
      <xs:documentation>
        The local system has a private key and uses it for
        shared secret key agreement with the Authentication Authority
        (e.g., via Diffie Helman).
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:complexType name="PublicKeyType">
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="keyValidation" use="optional"/>
  </xs:complexType>

  <xs:element name="IPAddress" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that the Principal has been
        authenticated through connection from a particular IP address.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="SharedSecretDynamicPlaintext" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        The local system and Authentication Authority
        share a secret key. The local system uses this to encrypt a
        randomised string to pass to the Authentication Authority.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="AuthenticatorTransportProtocol"
type="AuthenticatorTransportProtocolType">
    <xs:annotation>
      <xs:documentation>
        The protocol across which Authenticator information is
        transferred to an Authentication Authority verifier.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="HTTP" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that the Authenticator has been
        transmitted using bare HTTP utilizing no additional security
        protocols.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="IPSec" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that the Authenticator has been
        transmitted using a transport mechanism protected by an IPSEC session.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="WTLS" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that the Authenticator has been
        transmitted using a transport mechanism protected by a WTLS session.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="MobileNetworkNoEncryption" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that the Authenticator has been
        transmitted solely across a mobile network using no additional
        security mechanism.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="MobileNetworkRadioEncryption" type="ExtensionOnlyType"/>
  <xs:element name="MobileNetworkEndToEndEncryption" type="ExtensionOnlyType"/>

  <xs:element name="SSL" type="ExtensionOnlyType">
    <xs:annotation>
      <xs:documentation>
        This element indicates that the Authenticator has been
        transmitted using a transport mechanism protected by an SSL or TLS
        session.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="PSTN" type="ExtensionOnlyType"/>
  <xs:element name="ISDN" type="ExtensionOnlyType"/>
  <xs:element name="ADSL" type="ExtensionOnlyType"/>

  <xs:element name="OperationalProtection" type="OperationalProtectionType">
    <xs:annotation>
      <xs:documentation>
        Refers to those characteristics that describe
        procedural security controls employed by the Authentication Authority.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="SecurityAudit" type="SecurityAuditType"/>
  <xs:element name="SwitchAudit" type="ExtensionOnlyType"/>
  <xs:element name="DeactivationCallCenter" type="ExtensionOnlyType"/>

  <xs:element name="GoverningAgreements" type="GoverningAgreementsType">
    <xs:annotation>
      <xs:documentation>
        Provides a mechanism for linking to external (likely
        human readable) documents in which additional business agreements,
        (e.g., liability constraints, obligations, etc.) can be placed.
      </xs:documentation>
    </xs:annotation>
  </xs:element>
```

```
<xs:element name="GoverningAgreementRef" type="GoverningAgreementRefType"/>

<xs:simpleType name="nymType">
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="anonymity"/>
    <xs:enumeration value="verinymity"/>
    <xs:enumeration value="pseudonymity"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="AuthnContextDeclarationBaseType">
  <xs:sequence>
    <xs:element ref="Identification" minOccurs="0"/>
    <xs:element ref="TechnicalProtection" minOccurs="0"/>
    <xs:element ref="OperationalProtection" minOccurs="0"/>
    <xs:element ref="AuthnMethod" minOccurs="0"/>
    <xs:element ref="GoverningAgreements" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="ID" type="xs:ID" use="optional"/>
</xs:complexType>

<xs:complexType name="IdentificationType">
  <xs:sequence>
    <xs:element ref="PhysicalVerification" minOccurs="0"/>
    <xs:element ref="WrittenConsent" minOccurs="0"/>
    <xs:element ref="GoverningAgreements" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="nym" type="nymType">
    <xs:annotation>
      <xs:documentation>
        This attribute indicates whether or not the
        Identification mechanisms allow the actions of the Principal to
        be linked to an actual end user.
      </xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>

<xs:complexType name="TechnicalProtectionBaseType">
  <xs:sequence>
    <xs:choice minOccurs="0">
      <xs:element ref="PrivateKeyProtection"/>
      <xs:element ref="SecretKeyProtection"/>
    </xs:choice>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="OperationalProtectionType">
  <xs:sequence>
    <xs:element ref="SecurityAudit" minOccurs="0"/>
    <xs:element ref="DeactivationCallCenter" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="AuthnMethodBaseType">
  <xs:sequence>
    <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
    <xs:element ref="Authenticator" minOccurs="0"/>
    <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="GoverningAgreementsType">
  <xs:sequence>
    <xs:element ref="GoverningAgreementRef" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="GoverningAgreementRefType">
  <xs:attribute name="governingAgreementRef" type="xs:anyURI" use="required"/>
</xs:complexType>

<xs:complexType name="PrincipalAuthenticationMechanismType">
  <xs:sequence>
    <xs:element ref="Password" minOccurs="0"/>
    <xs:element ref="RestrictedPassword" minOccurs="0"/>
    <xs:element ref="Token" minOccurs="0"/>
    <xs:element ref="Smartcard" minOccurs="0"/>
    <xs:element ref="ActivationPin" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="preauth" type="xs:integer" use="optional"/>
</xs:complexType>

<xs:group name="AuthenticatorChoiceGroup">
  <xs:choice>
    <xs:element ref="PreviousSession"/>
    <xs:element ref="ResumeSession"/>
    <xs:element ref="DigSig"/>
    <xs:element ref="Password"/>
    <xs:element ref="RestrictedPassword"/>
    <xs:element ref="ZeroKnowledge"/>
    <xs:element ref="SharedSecretChallengeResponse"/>
    <xs:element ref="SharedSecretDynamicPlaintext"/>
    <xs:element ref="IPAddress"/>
    <xs:element ref="AsymmetricDecryption"/>
    <xs:element ref="AsymmetricKeyAgreement"/>
    <xs:element ref="SubscriberLineNumber"/>
    <xs:element ref="UserSuffix"/>
    <xs:element ref="ComplexAuthenticator"/>
  </xs:choice>
</xs:group>

<xs:group name="AuthenticatorSequenceGroup">
  <xs:sequence>
    <xs:element ref="PreviousSession" minOccurs="0"/>
    <xs:element ref="ResumeSession" minOccurs="0"/>
    <xs:element ref="DigSig" minOccurs="0"/>
    <xs:element ref="Password" minOccurs="0"/>
    <xs:element ref="RestrictedPassword" minOccurs="0"/>
    <xs:element ref="ZeroKnowledge" minOccurs="0"/>
    <xs:element ref="SharedSecretChallengeResponse" minOccurs="0"/>
    <xs:element ref="SharedSecretDynamicPlaintext" minOccurs="0"/>
    <xs:element ref="IPAddress" minOccurs="0"/>
    <xs:element ref="AsymmetricDecryption" minOccurs="0"/>
    <xs:element ref="AsymmetricKeyAgreement" minOccurs="0"/>
    <xs:element ref="SubscriberLineNumber" minOccurs="0"/>
    <xs:element ref="UserSuffix" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:group>

<xs:complexType name="AuthenticatorBaseType">
  <xs:sequence>
    <xs:group ref="AuthenticatorChoiceGroup"/>
    <xs:group ref="AuthenticatorSequenceGroup"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ComplexAuthenticatorType">
  <xs:sequence>
    <xs:group ref="AuthenticatorChoiceGroup"/>
    <xs:group ref="AuthenticatorSequenceGroup"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="AuthenticatorTransportProtocolType">
  <xs:sequence>
    <xs:choice minOccurs="0">
      <xs:element ref="HTTP"/>
      <xs:element ref="SSL"/>
      <xs:element ref="MobileNetworkNoEncryption"/>
      <xs:element ref="MobileNetworkRadioEncryption"/>
      <xs:element ref="MobileNetworkEndToEndEncryption"/>
      <xs:element ref="WTLS"/>
      <xs:element ref="IPSec"/>
      <xs:element ref="PSTN"/>
      <xs:element ref="ISDN"/>
      <xs:element ref="ADSL"/>
    </xs:choice>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="KeyActivationType">
  <xs:sequence>
    <xs:element ref="ActivationPin" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="KeySharingType">
  <xs:attribute name="sharing" type="xs:boolean" use="required"/>
</xs:complexType>

<xs:complexType name="PrivateKeyProtectionType">
  <xs:sequence>
    <xs:element ref="KeyActivation" minOccurs="0"/>
    <xs:element ref="KeyStorage" minOccurs="0"/>
    <xs:element ref="KeySharing" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="PasswordType">
  <xs:sequence>
    <xs:element ref="Length" minOccurs="0"/>
    <xs:element ref="Alphabet" minOccurs="0"/>
    <xs:element ref="Generation" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="ExternalVerification" type="xs:anyURI" use="optional"/>
</xs:complexType>

<xs:element name="RestrictedPassword" type="RestrictedPasswordType"/>

<xs:complexType name="RestrictedPasswordType">
  <xs:complexContent>
    <xs:restriction base="PasswordType">
      <xs:sequence>
        <xs:element name="Length" type="RestrictedLengthType" minOccurs="1"/>
        <xs:element ref="Generation" minOccurs="0"/>
        <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="ExternalVerification" type="xs:anyURI"
use="optional"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="RestrictedLengthType">
  <xs:complexContent>
    <xs:restriction base="LengthType">
      <xs:attribute name="min" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="3"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="max" type="xs:integer" use="optional"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="ActivationPinType">
  <xs:sequence>
    <xs:element ref="Length" minOccurs="0"/>
    <xs:element ref="Alphabet" minOccurs="0"/>
    <xs:element ref="Generation" minOccurs="0"/>
    <xs:element ref="ActivationLimit" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="Alphabet" type="AlphabetType"/>
<xs:complexType name="AlphabetType">
  <xs:attribute name="requiredChars" type="xs:string" use="required"/>
  <xs:attribute name="excludedChars" type="xs:string" use="optional"/>
  <xs:attribute name="case" type="xs:string" use="optional"/>
</xs:complexType>

<xs:complexType name="TokenType">
  <xs:sequence>
    <xs:element ref="TimeSyncToken"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="DeviceTypeType">
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="hardware"/>
    <xs:enumeration value="software"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="booleanType">
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="true"/>
    <xs:enumeration value="false"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="TimeSyncTokenType">
  <xs:attribute name="DeviceType" type="DeviceTypeType" use="required"/>
  <xs:attribute name="SeedLength" type="xs:integer" use="required"/>
  <xs:attribute name="DeviceInHand" type="booleanType" use="required"/>
</xs:complexType>

<xs:complexType name="ActivationLimitType">
  <xs:choice>
    <xs:element ref="ActivationLimitDuration"/>
    <xs:element ref="ActivationLimitUsages"/>
    <xs:element ref="ActivationLimitSession"/>
  </xs:choice>
</xs:complexType>

<xs:element name="ActivationLimitDuration" type="ActivationLimitDurationType">
  <xs:annotation>
    <xs:documentation>
      This element indicates that the Key Activation Limit is
      defined as a specific duration of time.
    </xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="ActivationLimitUsages" type="ActivationLimitUsagesType">
  <xs:annotation>
```

```
        <xs:documentation>
          This element indicates that the Key Activation Limit is
          defined as a number of usages.
        </xs:documentation>
      </xs:annotation>
    </xs:element>

    <xs:element name="ActivationLimitSession" type="ActivationLimitSessionType">
      <xs:annotation>
        <xs:documentation>
          This element indicates that the Key Activation Limit is
          the session.
        </xs:documentation>
      </xs:annotation>
    </xs:element>

    <xs:complexType name="ActivationLimitDurationType">
      <xs:attribute name="duration" type="xs:duration" use="required"/>
    </xs:complexType>

    <xs:complexType name="ActivationLimitUsagesType">
      <xs:attribute name="number" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:complexType name="ActivationLimitSessionType"/>

    <xs:complexType name="LengthType">
      <xs:attribute name="min" type="xs:integer" use="required"/>
      <xs:attribute name="max" type="xs:integer" use="optional"/>
    </xs:complexType>

    <xs:simpleType name="mediumType">
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="memory"/>
        <xs:enumeration value="smartcard"/>
        <xs:enumeration value="token"/>
        <xs:enumeration value="MobileDevice"/>
        <xs:enumeration value="MobileAuthCard"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="KeyStorageType">
      <xs:attribute name="medium" type="mediumType" use="required"/>
    </xs:complexType>

    <xs:complexType name="SecretKeyProtectionType">
      <xs:sequence>
        <xs:element ref="KeyActivation" minOccurs="0"/>
        <xs:element ref="KeyStorage" minOccurs="0"/>
        <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SecurityAuditType">
      <xs:sequence>
        <xs:element ref="SwitchAudit" minOccurs="0"/>
        <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ExtensionOnlyType">
      <xs:sequence>
        <xs:element ref="Extension" minOccurs="0"  maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>

    <xs:element name="Extension" type="ExtensionType"/>

    <xs:complexType name="ExtensionType">
      <xs:sequence>
        <xs:any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>

  </xs:schema>


  <?xml version="1.0" encoding="UTF-8"?>
  <xs:schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:ac"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:oasis:names:tc:SAML:2.0:ac"
    blockDefault="substitution"
    version="2.0">

    <xs:annotation>
      <xs:documentation>
        Document identifier: saml-schema-authn-context-2.0
        Location: http://docs.oasis-open.org/security/saml/v2.0/
        Revision history:
          V2.0 (March, 2005):
            New core authentication context schema for SAML V2.0.
            This is just an include of all types from the schema
            referred to in the include statement below.
      </xs:documentation>
    </xs:annotation>

    <xs:include schemaLocation="saml-schema-authn-context-types-2.0.xsd"/>

  </xs:schema>
```

NOTE – The use of SSL is presented in Appendix IV.

Appendix VII

SAML DCE PAC attribute profile

This appendix discusses SAML binding profile for distributed computing environment (DCE), privilege attribute certificates (PAC) (see opensource DCE).

VII.1 DCE PAC attribute profile

The DCE PAC attribute profile defines the expression of DCE PAC information as SAML attribute names and values. It is used to standardize a mapping between the primary information that makes up a DCE principal's identity and a set of SAML attributes. This profile builds on the UUID attribute profile defined in 11.4.9.3.

1) **Required information**

   - **Identification**: `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE` (this is also the target namespace assigned in the corresponding DCE PAC attribute profile schema in Annex A)
   - **Contact information**: security-services-comment@lists.oasis-open.org
   - **Description**: Given below.
   - **Updates**: None.

2) **PAC description**

   A DCE PAC is an extensible structure that can carry arbitrary DCE registry attributes, but a core set of information is common across principals and makes up the bulk of a DCE identity:

   - The principal's DCE "realm" or "cell".
   - The principal's unique identifier.
   - The principal's primary DCE local group membership.
   - The principal's set of DCE local group memberships (multi-valued).

- The principal's set of DCE foreign group memberships (multi-valued).

The primary value(s) of each of these attributes is a UUID.

3) **SAML attribute naming**

This profile defines a mapping of specific DCE information into SAML attributes, and thus defines actual specific attribute names, rather than a naming convention.

For all attributes defined by this profile, the `NameFormat` XML attribute in `<Attribute>` elements must have the value `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

For purposes of human readability, there may also be a requirement for some applications to carry an optional string name together with the URI. The optional XML attribute `FriendlyName` may be used for this purpose.

4) **Attribute name comparison**

Two `<Attribute>` elements refer to the same SAML attribute if and only if their Name XML attribute values are equal in the sense of ITU-T Rec. X.667. The `FriendlyName` attribute plays no role in the comparison.

5) **Profile-specific XML attributes**

No additional XML attributes are defined for use with the `<Attribute>` element.

6) **SAML attribute values**

The primary value(s) of each of the attributes defined by this profile is a UUID. The URN syntax described in 11.4.9.3 is used to represent such values.

However, additional information associated with the UUID value is permitted by this profile, consisting of a friendly, human-readable string, and an additional UUID representing a DCE cell or realm. The additional information is carried in the `<AttributeValue>` element in `FriendlyName` and Realm XML attributes defined in the XML namespace `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE`. This is not the same as the `FriendlyName` XML attribute defined in clause 8, although it has the same basic purpose.

The following schema listing shows how the profile-specific XML attributes and complex type are used in an `xsi:type` (Annex A):

```
<schema
targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"
    xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <annotation>
        <documentation>
            Document identifier: saml-schema-dce-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
            V2.0 (March, 2005):
                Custom schema for DCE attribute profile, first published in
SAML 2.0.
        </documentation>
    </annotation>
    <complexType name="DCEValueType">
        <simpleContent>
            <extension base="anyURI">
                <attribute ref="dce:Realm" use="optional"/>
                <attribute ref="dce:FriendlyName" use="optional"/>
            </extension>
        </simpleContent>
    </complexType>
    <attribute name="Realm" type="anyURI"/>
    <attribute name="FriendlyName" type="string"/>
</schema>
```

7) **Attribute definitions**

The following are the set of SAML attributes defined by this profile. In each case, an `xsi:type` XML attribute may be included in the `<AttributeValue>` element, but must have the value

dce:DCEValueType, where the dce prefix is arbitrary and must be bound to the XML namespace `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE`.

Such use of `xsi:type` will require validating attribute consumers to include the extension schema defined by this profile.

a) **Realm**

This single-valued attribute represents the SAML assertion subject's DCE realm or cell.

Name: `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm`

The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion subject's DCE realm/cell, with an optional profile-specific `FriendlyName` XML attribute containing the realm's string name.

b) **Principal**

This single-valued attribute represents the SAML assertion subject's DCE principal identity.

Name: `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal`

The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion subject's DCE principal identity, with an optional profile-specific `FriendlyName` XML attribute containing the principal's string name.

The profile-specific Realm XML attribute may be included and must contain a UUID in URN form identifying the SAML assertion subject's DCE realm/cell.

c) **Primary group**

This single-valued attribute represents the SAML assertion subject's primary DCE group membership.

Name: `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group`

The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion subject's primary DCE group, with an optional profile-specific `FriendlyName` XML attribute containing the group's string name.

The profile-specific Realm XML attribute may be included and must contain a UUID in URN form identifying the SAML assertion subject's DCE realm/cell.

d) **Groups**

This multi-valued attribute represents the SAML assertion subject's DCE local group memberships.

Name: `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups`

Each `<AttributeValue>` element `contains` a UUID in URN form identifying a DCE group membership of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute containing the group's string name.

The profile-specific Realm XML attribute may be included and must contain a UUID in URN form identifying the SAML assertion subject's DCE realm/cell.

e) **Foreign groups**

This multi-valued attribute represents the SAML assertion subject's DCE foreign group memberships.

Name: `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups`

Each `<AttributeValue>` element contains a UUID in URN form identifying a DCE foreign group membership of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute containing the group's string name.

The profile-specific realm XML attribute must be included and must contain a UUID in URN form identifying the DCE realm/cell of the foreign group.

VII.2 SAML schema dce

This is the SAML authentication context schema for distributed computing environment (DCE).

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"
    xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <annotation>
        <documentation>
            Document identifier: saml-schema-dce-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
            V2.0 (March, 2005):
                Custom schema for DCE attribute profile, first published in SAML 2.0.
        </documentation>
    </annotation>
    <complexType name="DCEValueType">
        <simpleContent>
            <extension base="anyURI">
                <attribute ref="dce:Realm" use="optional"/>
                <attribute ref="dce:FriendlyName" use="optional"/>
            </extension>
        </simpleContent>
    </complexType>
    <attribute name="Realm" type="anyURI"/>
    <attribute name="FriendlyName" type="string"/>
</schema>
```

VII.3 Example

The following is an example of the transformation of PAC data into SAML attributes belonging to a DCE principal named "jdoe" in realm "example.com", a member of the "cubicle-dwellers" and "underpaid" local groups and an "engineers" foreign group.

```
<saml:Assertion
xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE" ...>
  <saml:Issuer>...</saml:Issuer>
  <saml:Subject>...</saml:Subject>
  <saml:AttributeStatement>
  <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"
       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm">
    <saml:AttributeValue xsi:type="dce:DCEValueType"
dce:FriendlyName="example.com">
    urn:uuid:003c6cc1-9ff8-10f9-990f-004005b13a2b
    </saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"
       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal">
    <saml:AttributeValue xsi:type="dce:DCEValueType"
dce:FriendlyName="jdoe">
    urn:uuid:00305ed1-a1bd-10f9-a2d0-004005b13a2b
    </saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"
       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-
group">
    <saml:AttributeValue xsi:type="dce:DCEValueType"
       dce:FriendlyName="cubicle-dwellers">
    urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b
    </saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"
       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups">
    <saml:AttributeValue xsi:type="dce:DCEValueType"
       dce:FriendlyName="cubicle-dwellers">
    urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b
    </saml:AttributeValue>
    <saml:AttributeValue xsi:type="dce:DCEValueType"
dce:FriendlyName="underpaid">
    urn:uuid:006a5a91-a2b7-10f9-824d-004005b13a2b
    </saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"
       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-
groups">
    <saml:AttributeValue xsi:type="dce:DCEValueType"
dce:FriendlyName="engineers"
       dce:Realm="urn:uuid:00583221-a35f-10f9-8b6e-004005b13a2b">
    urn:uuid:00099cf1-a355-10f9-9e95-004005b13a2b
    </saml:AttributeValue>
  </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

Appendix VIII

OASIS clarifications of SAML

This appendix adds reviews that were done on SAML v2.0 within OASIS. The OASIS SAML Group decided to publish these clarification comments as a separate document (see OASIS:2006 PE). These clarifications are Non-Normative and were not incorporated in version 2.0 of OASIS SAML. In this Recommendation, these reviews are listed in this appendix to ensure that SAML implementers are aware of the discussions that took place after the publication of OASIS SAML v2.0 as an OASIS standard.

VIII.1 Potential errata: PE14

Description: Allowcreate needs more clear definition

Applicability in the Recommendation:

Refer to the appropriate Notes in 8.2.4.1 and 8.2.6. Furthermore, in 8.2.6.3, a clarification of the second paragraph in the subclause is provided below:

If the `<Terminate>` element is included in the request, the requesting provider is indicating that (in the case of a service provider) it will no longer accept assertions from the identity provider or (in the case of an identity provider) it will no longer issue assertions to the service provider about the principal.

If the receiving provider is maintaining state associated with the name identifier, such as the value of the identifier itself (in the case of a pair-wise identifier), an `SPProvidedID` value, the sender's consent to the identifier's creation/use, etc., then the receiver can perform any maintenance with the knowledge that the relationship represented by the name identifier has been terminated.

Any subsequent operations performed by the receiver on behalf of the sender regarding the principal (for example, a subsequent `<AuthnRequest>`) should be carried out in a manner consistent with the absence of any previous state.

Termination is potentially the cleanup step for any state management behaviour triggered by the use of the `AllowCreate` attribute in the Authentication Request protocol in 8.2.4. Deployments that do not make use of that attribute are likely to avoid the use of the `<Terminate>` element or would treat it as a purely advisory matter.

Note that in most cases (a notable exception being the rules surrounding the `SPProvidedID` attribute), there are no requirements on either identity providers or service providers regarding the creation or use of persistent state. Therefore, no explicit behaviour is mandated when the `<Terminate>` element is 450 received. However, if persistent state is present pertaining to the use of an identifier (such as if an `SPProvidedID` attribute was attached), the `<Terminate>` element provides a clear indication that this state should be deleted (or marked as obsolete in some fashion).

VIII.2 Potential errata: PE26

Description: SSO Profile needs clarifications

Applicability in the Recommendation: The following subclauses are clarified as below:

11.4.1.4.2 <Response> Usage

If the identity provider wishes to return an error, it must not include any assertions in the `<Response>` message. Otherwise, if the request is successful (or if the response is not associated with a request), the `<Response>` element must conform to the following:

- If the response is unsigned, the `<Issuer>` element may be omitted, but if present (or if the response is signed) it must contain the unique identifier of the issuing identity provider; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.
- It must contain at least one `<Assertion>`. Each assertion's `<Issuer>` element must contain the unique identifier of the responding identity provider; the `Format` attribute must be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`. Note that this profile assumes a single responding identity provider, and all assertions in a response must be issued by the same entity.
- If multiple assertions are included, then each assertion's `<Subject>` element must refer to the same principal. It is allowable for the content of the `<Subject>` elements to differ (e.g., using different `<NameID>` or alternative `<SubjectConfirmation>` elements).
- Any assertion issued for consumption using this profile must contain a `<Subject>` element with at least one `<SubjectConfirmation>` element containing a `Method` of `urn:oasis:names:tc:SAML:2.0:cm:bearer`. Such an assertion is termed a bearer assertion. Bearer assertions may contain additional `<SubjectConfirmation>` elements.
- Assertions without a bearer `<SubjectConfirmation>` may also be included; processing of additional assertions or `<SubjectConfirmation>` elements is outside the scope of this profile.
- At least one bearer `<SubjectConfirmation>` element must contain a `<SubjectConfirmationData>` element that itself must contain a `Recipient` attribute containing the service provider's assertion consumer service URL and a `NotOnOrAfter` attribute that limits the window during which the assertion can be delivered. It may also contain an `Address` attribute limiting the client address from which the assertion can be delivered. It must not contain a `NotBefore` attribute. If the containing message is in response to an `<AuthnRequest>`, then the `InResponseTo` attribute must match the request's `ID`.
- The set of one or more bearer assertions must contain at least one `<AuthnStatement>` that reflects the authentication of the principal to the identity provider. Multiple `<AuthnStatement>` elements may be included, but the semantics of multiple statements is not defined by this profile.
- If the identity provider supports the Single Logout profile, defined in 11.4.1.4.5, any authentication statements must include a `SessionIndex` attribute to enable per-session logout requests by the service provider.
- Other statements may be included in the bearer assertion(s) at the discretion of the identity provider. In particular, `<AttributeStatement>` elements may be included. The `<AuthnRequest>` may contain an `AttributeConsumingServiceIndex` XML attribute referencing in formation about desired or required attributes in clause 9. The identity provider may ignore this, or send other attributes at its discretion.
- Each bearer assertion must contain an `<AudienceRestriction>` including the service provider's unique identifier as an `<Audience>`.
- Other conditions (and other `<Audience>` elements) may be included as requested by the service provider or at the discretion of the identity provider. (Of course, all such conditions must be understood by and accepted by the service provider in order for the assertion to be considered valid.)
- The identity provider is not obligated to honour the requested set of `<Conditions>` in the `<AuthnRequest>`, if any.

11.4.1.4.3 <Response> message processing rules

Regardless of the SAML binding used, the service provider must do the following:

- Verify any signatures present on the assertion(s) or the response.
- Verify that the `Recipient` attribute in the bearer `<SubjectConfirmationData>` matches the assertion consumer service URL to which the `<Response>` or artifact was delivered.
- Verify that the `NotOnOrAfter` attribute in the bearer `<SubjectConfirmationData>` has not passed, subject to allowable clock skew between the providers.
- Verify that the `InResponseTo` attribute in the bearer `<SubjectConfirmationData>` equals the `ID` of its original `<AuthnRequest>` message, unless the response is unsolicited in which case the attribute must not be present.
- Verify that any assertions relied upon are valid in other respects. Note that while multiple bearer `<SubjectConfirmation>` elements may be present, the successful evaluation of a single such element in accordance with this profile is sufficient to confirm an assertion. However, each assertion, if more than one is present, must be evaluated independently.
- If the bearer `<SubjectConfirmationData>` includes an `Address` attribute, the service provider may check the user agent's client address against it.
- Any assertion which is not valid, or whose subject confirmation requirements cannot be met should be discarded and should not be used to establish a security context for the principal.
- If an `<AuthnStatement>` used to establish a security context for the principal contains a `SessionNotOnOrAfter` attribute, the security context should be discarded once this time is reached, unless the service provider re-establishes the principal's identity by repeating the use of this profile. Note that if multiple `<AuthnStatement>` elements are present, the `SessionNotOnOrAfter` value closest to the present time should be honoured.

11.4.1.4.4 POST-specific processing rules

If the HTTP POST binding is used to deliver the `<Response>`, each assertion must be protected by a digital signature. This can be accomplished by signing each individual `<Assertion>` element or by signing the `<Response>` element.

The service provider must ensure that bearer assertions are not replayed, by maintaining the set of used `ID` values for the length of time for which the assertion would be considered valid based on the `NotOnOrAfter` attribute in the `<SubjectConfirmationData>`.

BIBLIOGRAPHY

- **FIPS-197** (2001), *Advanced Encryption Standard* (AES).
- **IETF RFC 1738** (1994), *Uniform Resource Locators (URL).*
- **IETF RFC 2256** (1997), *A Summary of the X.500 (96) User Schema for use with LDAPv3*.
- **IETF RFC 2279** (1998), *UTF-8, a transformation format of ISO 10646*.
- **IETF RFC 2743** (2000), *Generic Security Service Application Program Interface Version 2, Update 1*.
- **DCE**, *Distributed Computing Environment (DCE)*, Open Source. See http://www.opengroup.org/dce.
- **OASIS Authentication Context 2.0**, *Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0*, 15 March 2005, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Bindings 1**, *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML),* 5 November 2002, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Bindings 1.1**, *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML),* 22 September 2003, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Bindings 2.0**, *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*, 15 March 2005, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Conformance 2.0**, *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.00*, 15 March 2005, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Glossary 2.0**, *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*, 15 March 2005, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Metadata 2.0**, *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*, 15 March 2005, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Errata Document 24**, *Revision 24 draft of the non-normative SAML V2.0 Errata document*, 27 February 2006, http://www.oasis-open.org/committees/download.php/16935/sstc-saml-errata-2.0-draft-24.pdf.
- **OASIS Protocol 1.0**, *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML),* 5 November 2002, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Protocol 1.1**, *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML),* 22 September 2003, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Protocol 2.0**, *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*, 15 March 2005, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS SAML 1.0**, *Security Assertion Markup Language (SAML) Version 1.0 Specification Set*, 5 November 2002, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS SAML 1.1**, *Security Assertion Markup Language (SAML) Version 1.1 Specification Set*, 22 September 2003, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Security 1**, *Security Considerations for the OASIS Security Assertion Markup Language (SAML),* 5 November 2002, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Security 1.1**, *Security Considerations for the OASIS Security Assertion Markup Language (SAML),* 22 September 2003, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS Security 2.0**, *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*, 15 March 2005, http://www.oasis-open.org/apps/org/workgroup/security/.
- **OASIS XACML1-1**, *eXtensible Access Control Markup Language (XACML) V1.1*, 24 July 2003, http://www.oasis-open.org/apps/org/workgroup/xacml/.
- **OASIS XACML1-1**, *eXtensible Access Control Markup Language (XACML) V1.0*, 18 February 2003, http://www.oasis-open.org/apps/org/workgroup/xacml/.
- **OASIS XACML 2.0**, *eXtensible Access Control Markup Language (XACML) V2.0*, 1 February 2005, http://www.oasis-open.org/apps/org/workgroup/xacml/.
- **SSL3**, *The SSL Protocol Version 3.0*. See http://wp.netscape.com/eng/ssl3/draft302.txt.
- **W3C Character Model** (2004), Working draft, 27 October 2005, *Character Model for the World Wide Web 1.0: Normalization.*

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	**Data networks, open system communications and security**
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2007