National Security Agency

Information Assurance Directorate

Net-Centric Enterprise Services (NCES) Security Assertion Markup Language (SAML) Attribute Profile

30 June 2008

Prepared for the

Defense Information Systems Agency (DISA)

By the

National Security Agency

9800 Savage Road

Fort George G. Meade, MD 20755

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF EXHIBITS

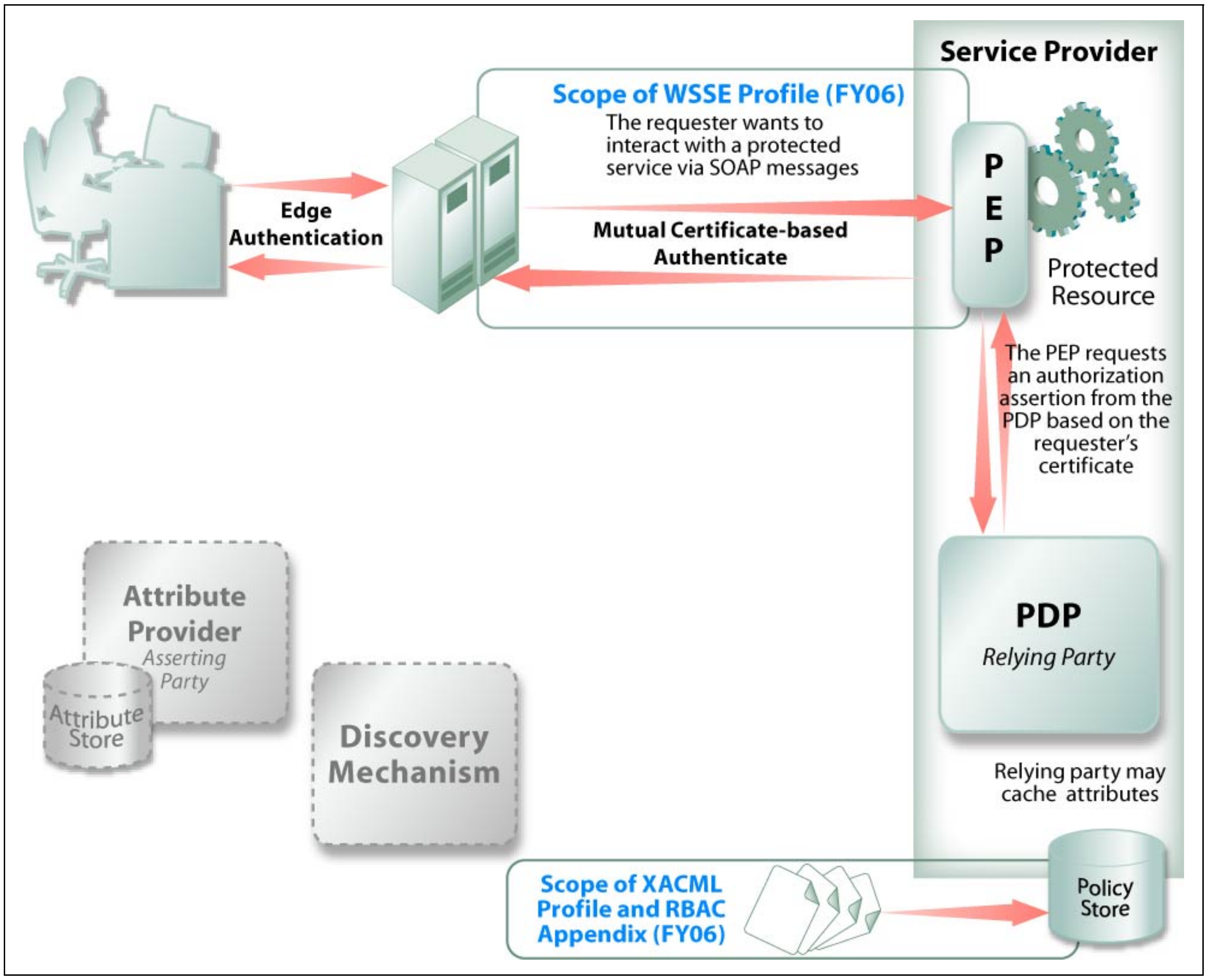
DOCUMENT VERSIONING HISTORY

Contributor	Update	Date
NSA IAD	Updates to NCES SAML Attribute Profile	4/13/2007
NSA IAD	Draft NCES SAML Attribute Profile	7/13/2007
NSA IAD	Final NCES SAML Attribute Profile	6/30/2008

1 SCOPE

This profile has been created to establish a standard means to express policies and attributes within the Security Assertion Markup Language (SAML) construct. The scope of this profile is strictly limited to transactions involving SAML attributes. The SAML standard prescribes transactions involving authentication assertions and policy decision assertions; these are corollary, but separate functions of SAML. Implementations that conform to this profile are also expected to support related functionality that is subject to other information assurance (IA) profiles.

This profile specifies implementation options and functional selections for SAML attribute authority policy definition and location, as well as attribute request and response, such that conformant implementations will satisfy the conformance requirements for SAML-based attribute authority services. Previous Net-Centric Enterprise Services (NCES) can be seen in **Figure 1–1: Previous NCES Profiles**.



Service Provider
Scope of WSSE Profile (FY06)
The requester wants to interact with a protected service via SOAP messages
Edge Authentication
Mutual Certificate-based Authenticate
P E P
Protected Resource
The PEP requests an authorization assertion from the PDP based on the requester's certificate
PDP
Relying Party
Relying party may cache attributes
Attribute Provider
Asserting Party
Attribute Store
Discovery Mechanism
Scope of XACML Profile and RBAC Appendix (FY06)
Policy Store

Figure 1–1: Previous NCES Profiles

The scope of the SAML Attribute profile is depicted in **Figure 1–2: Scope of the NCES SAML Attribute Profile**.

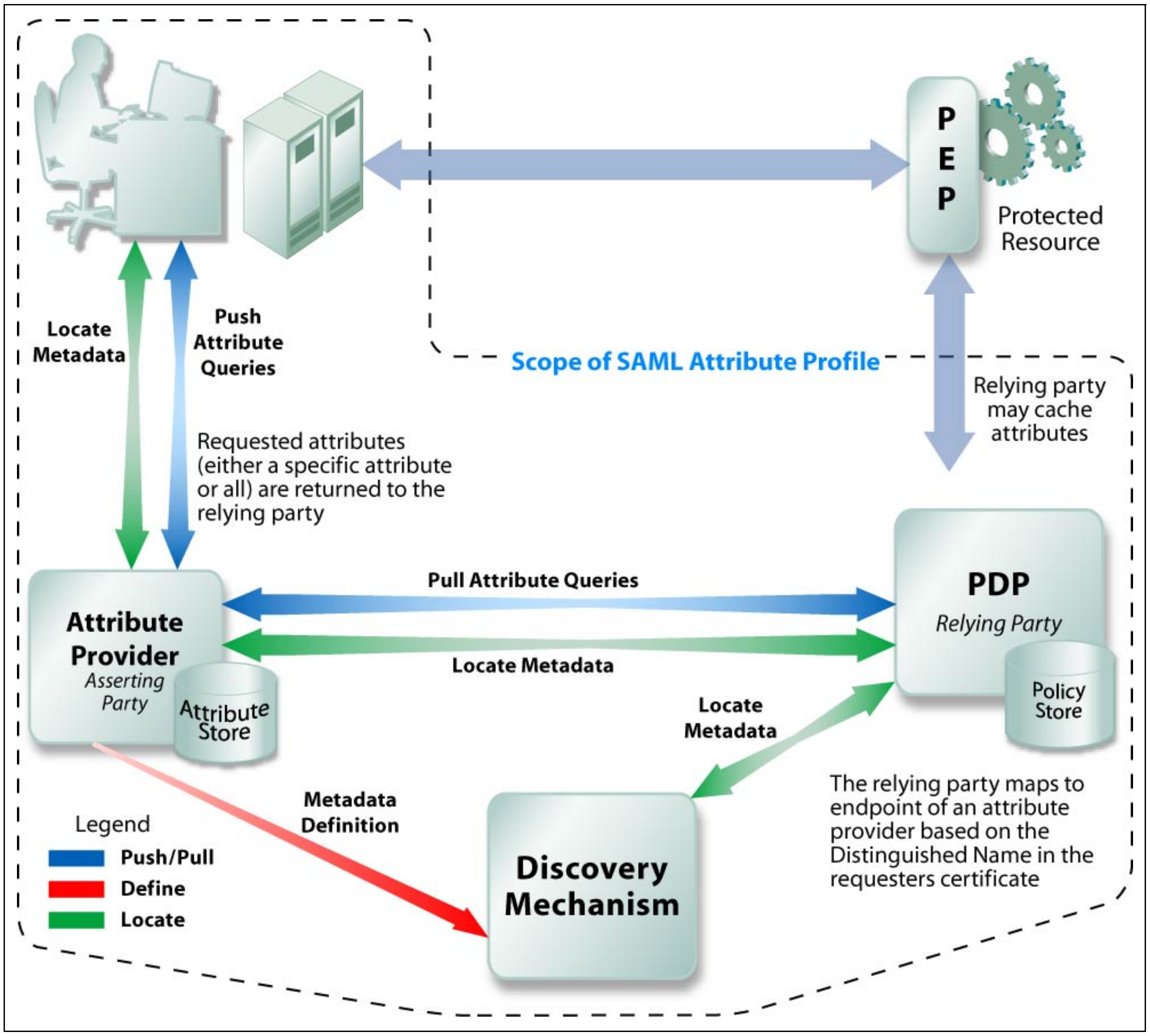

P
E
P
Protected Resource
Locate Metadata
Push Attribute Queries
Scope of SAML Attribute Profile
Relying party may cache attributes
Requested attributes (either a specific attribute or all) are returned to the relying party
Pull Attribute Queries
Attribute Provider
Asserting Party
Attribute Store
Locate Metadata
PDP
Relying Party
Policy Store
Locate Metadata
Metadata Definition
The relying party maps to endpoint of an attribute provider based on the Distinguished Name in the requesters certificate
Discovery Mechanism
Legend
Push/Pull
Define
Locate

Figure 1–2: Scope of the NCES SAML Attribute Profile

1.1 NCES IA Profiles Taxonomy

This specification is part of a collection of profiles to augment existing web services (WS) standards to meet NCES IA requirements. The objectives of these profiles are to:

- Refine WS standards requirements to improve interoperability.
- Identify known gaps in the standards without necessarily taking actions on these gaps.
- Address known WS security vulnerabilities where possible through interface profiles.
- Harmonize standards profiles with existing security architecture efforts, in particular NCES.

All of the NCES IA profiles are in alignment with the taxonomy utilized in the development of industry standard profiles. The top level taxonomy of major interface types is defined by the following three categories:

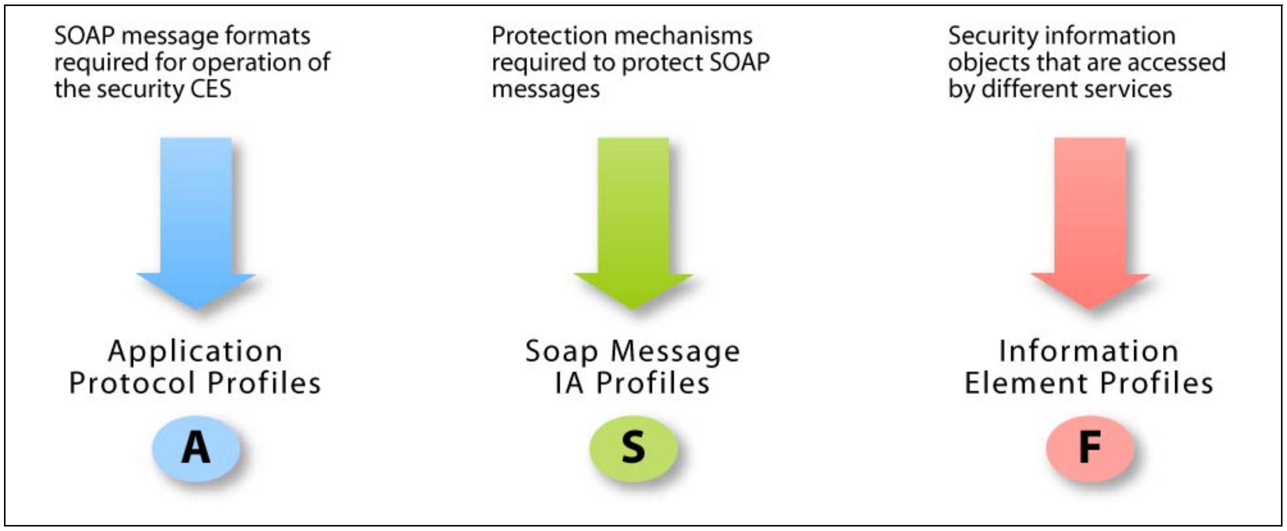

SOAP message formats required for operation of the security CES
Protection mechanisms required to protect SOAP messages
Security information objects that are accessed by different services
Application Protocol Profiles
Soap Message IA Profiles
Information Element Profiles
A
S
F

Figure 1–3: Top Level Taxonomy for Profile Interface

Specifically, this profile is part of the series of SOAP message IA Profiles (i.e., S-profiles) that are described in the IA profiles taxonomy. The relationship between this profile [(SAML-P) Attribute Profile] to other S-profiles is illustrated in **Figure 1–4**. Its position within the taxonomy defines its scope: the SAML Protocol transactions that carry SAML Attributes. These protocols are at their core defined by the [SAML1Core] and [SAML2Core] references. In addition to this profile, **S2**, NSA-Information Assurance Directorate (IAD) has also developed Profiles **S0** and **F21**. Future profiles are also under consideration.

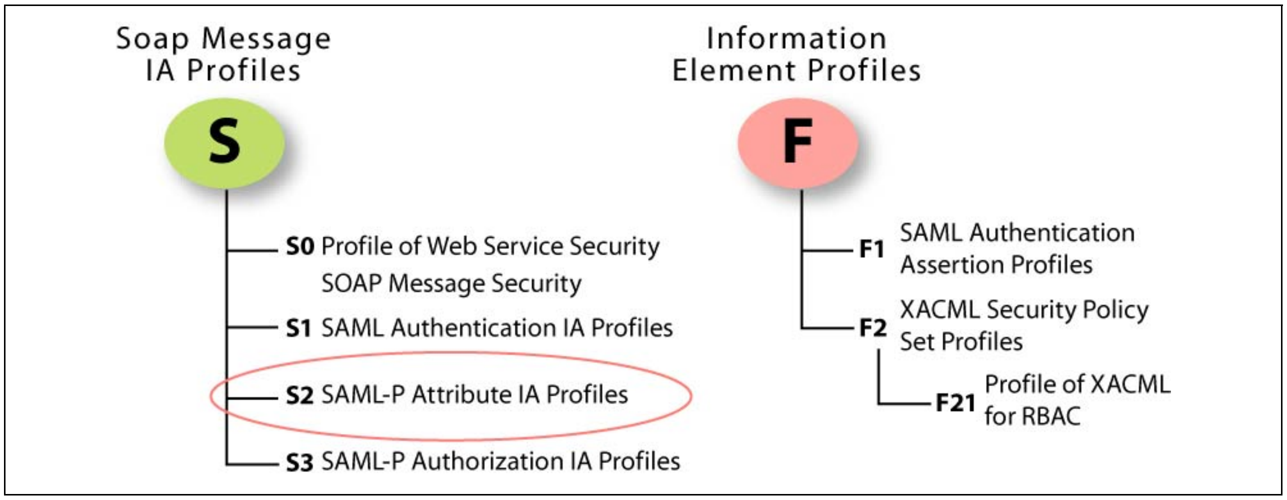

Soap Message IA Profiles
S
S0 Profile of Web Service Security SOAP Message Security
S1 SAML Authentication IA Profiles
S2 SAML-P Attribute IA Profiles
S3 SAML-P Authorization IA Profiles
Information Element Profiles
F
F1 SAML Authentication Assertion Profiles
F2 XACML Security Policy Set Profiles
F21 Profile of XACML for RBAC

Figure 1–4: Relationship of the SAML Attribute Profile to other Profiles

1.2 Dependencies on Other Profiles

This section contains information related to the dependencies between this profile and other profiles in the NCES IA profiles taxonomy. As discussed in the security considerations section, this profile is dependent upon the following profiles to remain in congruence with previously agreed-upon rules for policy compliant message exchange.

- NCES Profile of Web Service Security: SOAP Message Security (WSSE): This profile defines the collective requirements for SOAP Message Security to support digital signatures, encryption, and tokens within the context of the NCES IA subsystem. It supports the definition of rules on signing and encrypting messages. Implementations conforming to the SAML Attribute Profile are expected to conform to this IA profile.
- NCES Profile of XACML: This profile covers the collective requirements for embedded eXtensible Access Control Markup Language (XACML) policy objects used to support a Role-Based Access Control (RBAC) function within the context of the NCES IA subsystem. The profile defined by this document was created with the expectation that implementations of it would not be in conflict with implementations of the NCES profile of XACML.

1.3 Development Methodology

This section defines the process by which the SAML Attribute Profile was developed. The first step was to evaluate publicly available (and ratified) standards relating to the exchange of attributes in accordance with the architectural assumptions made by NCES. The second step was to analyze documentation relating to the SAML 2.0 (and SAML 1.1) standards and to identify relevant information flow patterns and use cases. Finally, the rule sets for the appropriate standards were examined to identify any areas of potential conflict or those requiring further clarification.

The following primary source documents were evaluated for the information flow patterns/use cases:

- Office of the Director of National Intelligence, Chief Information Officer, Intelligence Community Enterprise Architecture, Service-Oriented Architecture, Security Reference Architecture [ODNI-SOASRA].
- *Net-Centric Enterprise Services (NCES) Security Core Enterprise Services (CES) Architecture* [NCESarch].
- Web Services Profile of XACML (WS-XACML) Version 1.0, Working Draft 8, 12 December 2006 [WS-XACML].

Each of these documents articulates transactional information flows predicated on transmission of attributes—in particular, attributes exchanged in accordance with SAML specifications. Therefore, use cases that defined authorization models were extracted from the primary source documents and normalized. Then, use cases relevant to the SAML Attribute Profile were identified. Documents from NCES, the Organization for the Advancement of Structured Information Standards (OASIS), World Wide Web Consortium (W3C), National Institute of Standards and Technology (NIST) and Internet Engineering Task Force (IETF) standards were evaluated as the basic building blocks for the profile. A full listing of documents is contained in **Section 9: References**. Some of these documents include:

- NCES Service Security Design and Interface Specifications.
- [SAMLv2.0], the collection of documents that make up Version 2.0 of the SAML profile.
- [SAMLv1.1], the collection of documents that make up Version 1.1 of the SAML profile.
- SAML Attribute Self-Query Profile for X.509 Subjects.
- SAML Attribute Query Profile for X.509 Subjects.

Upon further analysis of these use cases, certain standard message exchange patterns emerged. NSA-IAD examined the individual information flows and examined the normative requirements for articulation within this profile.

2 OVERVIEW

2.1 Goals

This profile has been created to establish a standard means to discover and query information relating to attribute authorities and their attributes. Specifically, it builds on current SAML profiles to define an “on the wire” interface to exchange attribute assertions when X.509 certificates are used for authentication. This profile is concerned with the publication of metadata specific to attribute authorities within the enterprise, the need to facilitate the use of the metadata related to attribute authorities, and the request/response messages exchanged between an attribute service and its consumers.

This profile employs the use of several diagrammatic conventions. It uses conceptual visual representations of authorization models, message transactions, and messages themselves, as well as eXtensible Markup Language (XML) code snippets of the content delivered within the messages. The next section depicts the relationship of these diagrammatic conventions.

2.2 Naming Conventions

The keywords "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this specification are to be interpreted as described in IETF Request for Comment (RFC) 2119.

Any XML fragments are illustrative only and non-normative unless otherwise indicated.

```
Example code listings appear like this. Within example code listings
the following typographical conventions are followed:
<ns:QualifiedElement>, attributes, attribute values, element values,
<document type statements>, comments.
```

`URI listings appear like this`.

This specification uses the following typographical conventions in text: `<UnqualifiedElement>`, `<ns:QualifiedElement>`, `Attribute`, ***Datatype***, OtherKeyword.

XPath is a W3C standard language for finding information in an XML document. XPath is used to navigate through elements and attributes in an XML document and is a key component in XSLT (eXtensible Stylesheet Language Transformations). XPath uses path expressions to select nodes or node-sets in an XML document. These path expressions look very similar to expressions you see when you work with a traditional computer file system. An example of the use of XPath would be ***element:attribute/@attributename***. This profile assumes a cursory knowledge of the rules and syntax that constitute an XPath expression.

2.3 Terminology

In general, the Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0 [SAML-Gloss] should be used as the reference for any terms and definitions that appear in this document that relate to the SAML language, authorization and authentication. However, there are some terms that either do not exist in the SAML Glossary, or have been expanded upon or redefined in some manner by another relevant document. Such terms are included in this section. When using the terms in this section, the reader should compare the provided definition to the definition that exists in the SAML Glossary, **Appendix B: Glossary of Terms**.

Furthermore, terms from the NCES architecture context documentation that are relevant to the guiding scenario of this profile are also included in this section. The source for each term within this section is cited.

Asserting Party - A mission entity that issues assertions. [NCESarch]

Attribute Assertion - A SAML assertion that contains security context information related to the attributes for an entity. The attribute assertion contains a SAML attribute statement that indicates details such as the issuer of the assertion (the attribute authority), the date/time of assertion, the subject of the assertion (corresponding to the subject from the request), and the attributes associated with the subject. The attribute assertion is the response to an attribute assertion request. [NCESarch]

Attribute Based Access Control (ABAC) - A policy model that allows for access control policy applicability and the associated rules that govern access, to be formulated based on an extensible notion of subject, resource, and other types of attributes. ABAC encompasses the following three tenets:

- An extensible notion of subject attributes encompassing identifiers, groups, roles, and any number of additional subject attribute types.
- The use of attributes in policy rules where attributes are compared with fixed values or with each other, in accordance with the appropriate security business logic(s).
- The use of resource attributes when specifying the applicability of a policy.

[ODNI-SOASRA], [NCESarch]

Attribute Consumer - An attribute consumer is a special category of SAML requesters that utilize the SAML protocol to request attributes about a subject on behalf of themselves as relying parties, on behalf of themselves as subjects, or on behalf of other entities acting as relying parties.

Attribute Information Service (AIS) - A service that provides attribute values that describe subjects (either human users or systems) for the purpose of enabling all IC organizations and partners to make access control decisions related to their data. [ODNI-SOASRA]

Mission Entity - An entity that is operationally responsible for performing a mission function or operationally in need of having a mission function performed. Mission entities may include entities such as individuals (humans), organizational units, Community of Interests (COIs), and programs of record. [NCESarch]

Principal - A principal is a system entity that has an identity, that may be authenticated, that is capable of making decisions, and to whom actions performed within the enterprise may be attributed. A principal may refer to human entities such as an individual user, an organization, or a legal entity; depending on the context, it may also refer to non-human system entities such as a web service provider. NOTE: This document makes a distinction between principals and Identities. A principal may have multiple local identities in different security domains. For example, a user principal can have a work account called “JDoe” in his employer’s network and also a personal account called “John Doe” issued by his Internet Service Provider (ISP). [NCESarch]

Service Provider - A mission entity that performs a mission function for another mission entity. NOTE: A service provider may be a consumer of other service providers. [NCESarch]

2.4 Namespaces

The SAML assertion schema defines the attribute assertion format. The SAML protocol schema defines an attribute query used for requesting instances of attribute assertions, and a response that contains the requested instances. Systems using XACML MAY use instances of these SAML elements, transmit and store SAML attributes. Systems using XACML MAY use the SAML attribute query protocol to request instances of SAML Attributes. To be used in an XACML request context, the SAML attribute SHALL be mapped to an XACML attribute.
The following table indicates all referenced XML namespaces within this document as well as the prefix used to indicate each specific namespace.

Table 2–1: Referenced XML Namespaces

Prefix	URI	Description
`xml`	`http://www.w3.org/TR/REC-xml`	XML Definition (for xml:lang)
`xs`	`http://www.w3.org/2001/XMLSchema`	XML Schema Definition
`ds`	`http://www.w3.org/2000//09/xmldsig#`	XML Digital Signature Definition
`saml`	`urn:oasis:names:tc:SAML:1.0:assertion`	SAML v1.1 Assertion Definition
`samlp`	`urn:oasis:names:tc:SAML:1.0:protocol`	SAML v1.1 Protocol Definition
`saml2`	`urn:oasis:names:tc:SAML:2.0:assertion`	SAML v2.0 Assertion Definition
`samlp2`	`urn:oasis:names:tc:SAML:2.0:protocol`	SAML v2.0 Protocol Definition
`md`	`urn:oasis:names:tc:SAML:2.0:metadata`	SAML v2.0 Metadata Descriptor
`xacml`	`urn:oasis:names:tc:xacml:2.0:policy:schema:os`	XACML v2.0 policy
`xacmlc`	`urn:oasis:names:tc:xacml:2.0:context`	XACML v2.0 context
`xacml-saml`	`urn:oasis:names:tc:xacml:2.0:saml:assertion:schema:os`	SAML 2.0 Profile of XACML 2.0 Assertion Extension
`xacml-samlp`	`urn:oasis:names:tc:xacml:2.0:saml:protocol:schema:os`	SAML 2.0 Profile of XACML 2.0 Protocol Extension

2.5 General Description

This profile was created to provide guidance on the representation of SAML attribute- related transactions. It articulates the rules for publishing attribute authority information; discovering metadata regarding an attribute authority as well as metadata regarding attributes; and for the request and response of SAML attributes.

2.5.1 Relationship to SAML

The SAML standard provides a standard means of describing and exchanging interoperable security credentials that assist in identifying key subjects in transactions between business or mission partners, regardless of the authorization product. Partners exchange these security credentials in the form of SAML assertions. The SAML standard defines the following kinds of assertions:

- Authentication assertions.
- Attribute assertions.
- Authorization assertions.

This profile provides additional guidance relating to attribute assertions and the adoption of multiple SAML profiles (as published by OASIS) to define the standard attribute exchanges within the NCES architecture context. In addition, it provides guidance on the distribution and discovery of metadata about attribute authorities which expose attributes for consumption. This profile seeks to articulate the means for defining the SAML attribute request and response within the NCES security services architecture.

2.5.2 Relationship to XACML

XACML provides a general purpose mechanism for expressing an organization's access control policies. At its core, XACML is an access-control policy language that allows standard specification of rules about who can do what, and when. XACML provides for fine-grained control of activities based on common types of criteria, such as entity attributes, authentication mechanisms, and the protocol employed. XACML defines a vocabulary for expressing these policies as XML constructs. Each policy defines individual rules as the basic unit of management. Each rule evaluates some combination of characteristics of the requester, the requested resource, the desired action and the current environment. This profile defines how to rely upon SAML request and response protocols and assertion formats as the attribute transmissions that will be used by Policy Decision Points (PDP) for input to access control decisions. Therefore, XACML information flows and requirements dictate certain requirements for the expression of attribute requests and attribute assertions so that such required information is available to the XACML-compliant PDP.

2.5.3 Relationship to the SAML 2.0 Profile of XACML

Section 8.5 of the SAML 2.0 Profile of XACML [SAML2Prof] defines how to use SAML 2.0 to protect, transport, and request XACML schema instances and other information needed by an XACML implementation. The SAML profile of XACML provides a means to map XACML attributes to SAML attributes. It includes a conceptual model of information flows between nodes of an authorization model; a few of the flows relate to the kinds of information flows that are captured within the authorization models discussed in **Section 3.2: Authorization Models**. The profile within this document has incorporated those flows into its approach. It has also levied requirements on attribute messages based on the need to interoperate with the XACML policy language NCES is expected to use.

2.5.4 Relationship to WS-Security

Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) [WSS: SOAP Message Security V1.0] provides a message-integrity and message-confidentiality extension to SOAP messaging. The core WS-Security standard is an extensible protocol for defining security tokens within SOAP messages. This profile defines the use of SAML assertions as an attached security token in the ***wsse:Security*** header block defined by the WSS: SOAP Message Security specification (and as constrained by the WSSE profile [WSS: SAML Token Profile]).

2.5.5 Relationship to WS-Trust

[WS-Trust] is a standard language for extending WS-Security for security token exchange between parties seeking to establish interoperable trust relationships across heterogeneous security infrastructures. [WS-Trust] works in conjunction with [WS-SecurityPolicy] to establish agreements about the nature of the security token exercised in a subsequent SOAP transaction. The language defines a request/response protocol by which SOAP entities can make requests of a trusted authority to map and exchange disparate security tokens. The result should be such that a SOAP client and service endpoints are guaranteed to receive only relevant and recognizable security tokens. Although this is crucial in establishing the underlying trust relationship between SOAP actors, it remains out of scope for the considerations of this profile**.** The assumption is that this interchange occurs prior to the exchange of attributes and any comments upon its usage would require an architectural decision to be made.

3 PROFILE SCENARIO

3.1 Assumptions

The following assumptions in the following subsections shape the requirements of this profile.

3.1.1 Establishment of Trust

The need to establish relationships of trust between mission entities and system entities is assumed. Authorization models described in this profile are dependent upon the entities described having previously established relationships. This profile establishes and defines the underlying trust between mission entities, system entities, and organizational entities within architectures that use the information flow patterns listed within this document. This can be established through enterprise governance or regulatory requirement. Its definition can be accomplished through engineering and/or legal means. However, trust cannot be established through automated means. This profile assumes that the establishment of some degree of trust precedes the message transactions that take place. Message transactions described within this profile support the security requirements of those previously established trust mechanisms. A series of prerequisite steps must take place prior to an authorization decision exchange that is reliant upon attribute information. The technical mechanism underlying trust may rely upon the exchange of root certificates to validate signature chains. Typically, this occurs out of band and is required to validate any information exchanged between two parties.

3.1.2 Attribute Provisioning

The ways that attributes are identified, collected, and established for subjects is out of the scope of this profile, which assumes only that asserting parties provide attributes about subjects and resources.

3.1.3 Definition of Authoritative Attribute Services

The degree to which an asserting party is considered by entities involved in an authorization to be authoritative and how this is done is out of the scope of this profile.

How an attribute service may be declared to be an authoritative attribute service is out of the scope of this profile.

3.1.4 Attribute Binding

Binding of attributes to the identity of the attribute provider is important for the integrity of the attributes and is a function carried out by the attribute authority. However, the means and mechanisms by which this is performed are out of the scope of this profile.

3.1.5 Representational State Transfer (REST)-based Services

Use of Hypertext Transfer Protocol (HTTP) POST, GET, PUT, DELETE or other technical mechanisms to distribute attribute information are out of the scope of this profile. The NCES Security Architecture context is built as a set of SOAP Web services. Therefore, no assumptions are made regarding the technical feasibility of REST-based implementations of attribute-based transactions that are described within this profile.

3.1.6 Decision Logic for Attribute Precedence in Policy Evaluation

This profile is entirely focused on transactional processes related to the collection of SAML attributes for the purpose of authorization decisions. No implied assumptions or statements are made concerning the need for decision logic within the actual authorization decision itself. Questions about precedence (i.e., which policy rule sets are evaluated, in what order, or with what weighting) are outside the scope of this profile.

3.1.7 Mechanisms for Attribute Service Discovery

While the publishing and discovery of related SAML 2.0 Metadata document Uniform Resource Locators (URLs) can be facilitated through the use of Dynamic Delegation Discovery System (DDDS) mechanisms such as querying Naming Authority Pointer (NAPTR) resource records, those mechanisms are not currently considered within the NCES architecture. They are reserved for future considerations. This profile assumes that the discovery of attribute services will rely upon the mechanisms provided by NCES Service Discovery *to the greatest extent possible*.

3.2 Authorization Models

An authorization model is a high-level conceptual description of the transactional process for adjudicating access control within an environment. It is a conceptual framework by which system entities can be identified and categorized as nodes. By identifying nodes and categorizing them, system architects and product vendors can maintain common understanding of the functional responsibilities and necessary capabilities that each category of system node requires.

The system entities that are identified by the authorization models discussed within this profile are the following:

- Consumer (or Service Consumer).
- Service Provider (or protected resources).
- Policy Enforcement Point (PEP).
- Policy Decision Point (PDP).
- Attribute Provider (AP).
- Policy Store.

These are the entities within the environment that are the important actors during the authorization process. Entities give, receive, and/or act upon information from other entities. Authorization models seek to define the patterns by which authorization information is distributed from entity to entity.

This profile discusses several influential authorization models that currently exist so as to define the functional responsibilities of each of these entities. (Specifically, this profile focuses on the functional responsibilities of entities that are involved with SAML attribute transactions.) This section describes three sets of authorization models that are currently being explored within three important enterprise realms. These realms are the DoD environment, the intelligence community environment, and the commercial world.

This profile provides means for discovery and retrieval of attributes and the requisite location and exchange of metadata on attribute services within the context of the NCES Security Services architecture. Within this architectural context, there are several sets of authorization models which have implications for the exchange of attributes and attribute metadata. In addition to NCES Security Services Authorization models, this profile considers the authorization interaction patterns contained within ODNI IC SOA Security Reference Architecture [ODNI-SOASRA] so that the requirements contained herein may support the eventual federation of these two security architectures. Furthermore, this profile considers the draft WS-XACML use cases which—although they are still in draft—are the initial authorization information flows published by the XACML technical committee. They have been considered with the intent to strengthen the relationship and maintain consistency between industry and defense approaches to authorization.

The following subsections describe these three sets of authorization models and elaborate on them.

3.2.1 NCES Security Services Authorization Models

The NCES Service Security Design Specification [NCESspec] was designed to accommodate different approaches for an ABAC implementation. The NCES Service Security Design Specification introduces and discusses four ABAC authorization models that represent the primary use cases in which attributes are requested, retrieved, and exchanged in support of attribute-based authorization policy decisions. These models reflect the different permutations that can occur during the ABAC authorization processes between the Service Consumer, Service Provider, Attribute Authority, PEP, PDP, and Policy Store. The primary differences between these models occur in the interactions and flow between the PDP, the PEP, the Service Consumer, and the Service Provider. In addition, a PEP should be capability of supporting multiple authorization models simultaneously.

The depictions of the following authorization models are logical representations and only illustrate a single AP, PEP, PDP, and Policy Store for the purpose of simplicity. However, there is no limitation on the number of components that can exist in an actual implementation. Although the NCES Security Service Design Specification only discusses four archetypical ABAC authorization models, there are a number of additional permutations to these basic models (through expansion, scaling, and hierarchical positioning). These additional permutations are not discussed directly within the NCES Security Service Design Specification. The remainder of this section will present the four basic NCES ABAC authorization models. Each authorization model is represented by a diagram and an ordered series of transactions describing the information transmitted between nodes. The description of an activity wherein a node “trusts” the information provided by another, conveys the need to validate that information. Within the NCES CES Security Service infrastructure, this is in part provided by the Certificate Validation Service (CVS). For the purpose of describing attribute transactions, this profile makes no requirement that the usage of the CVS be mandatory. The use of the word “trust” in this manner seeks to convey the need to address, through policy or otherwise outstanding requirements that attribute information have a means to be trusted.

3.2.1.1 ABAC - Direct Retrieval Authorization Model

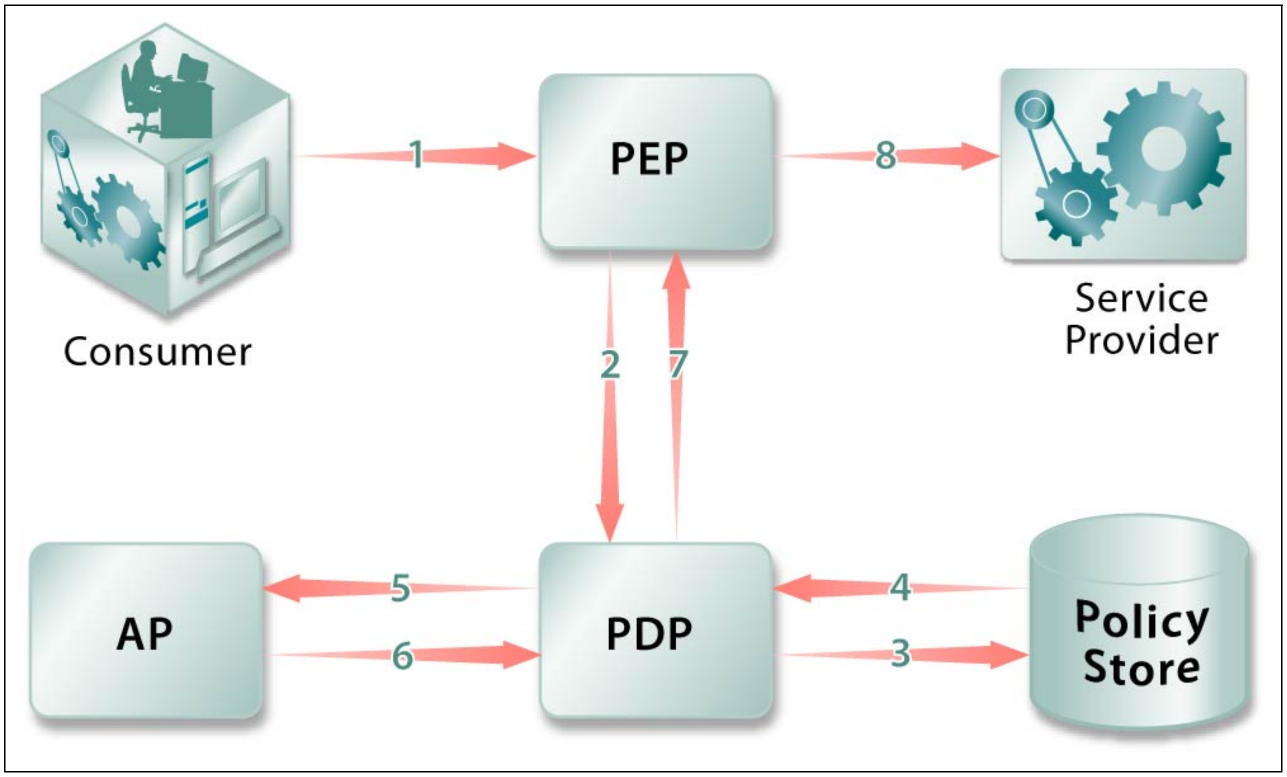


Figure 3–5: ABAC - Direct Retrieval Authorization Model

This model represents a traditional access control process whereby there is close association of the functional nodes performing the step-by-step access control system functions. The majority of ABAC implementations will take this form. As discussed in **Section 3.3** of this profile, this model can be categorized as having a "direct retrieval" transaction pattern, because the functional node performing the access control decisions asks for attributes directly from the AP.

Table 3–2: ABAC - Direct Retrieval Authorization Model - Details

Step	Description
1	A Consumer sends a request to a Service Provider that is managed by a PEP.
2	The PEP requests an authorization decision from the PDP.
3	The PDP requests authorization policies from the Policy Store.
4	The Policy Store returns a policy assertion to the PDP.
5	The PDP requests attributes from the AP, based on the service consumer's identity.
6	The AP returns attribute assertions with the relevant service consumer attributes to the PDP.
7	The PDP makes a decision based on the service consumer's attributes and the policies and returns a decision assertion to the PEP. • The PDP trusts the attributes returned from the AP. • The PDP trusts the policies returned from the Policy Store.
8	The PEP either grants or denies access to the service provider, based on the decision assertion. • The PEP trusts the decision assertion generated by the PDP.

3.2.1.2 ABAC - Pre-Retrieval Authorization Model

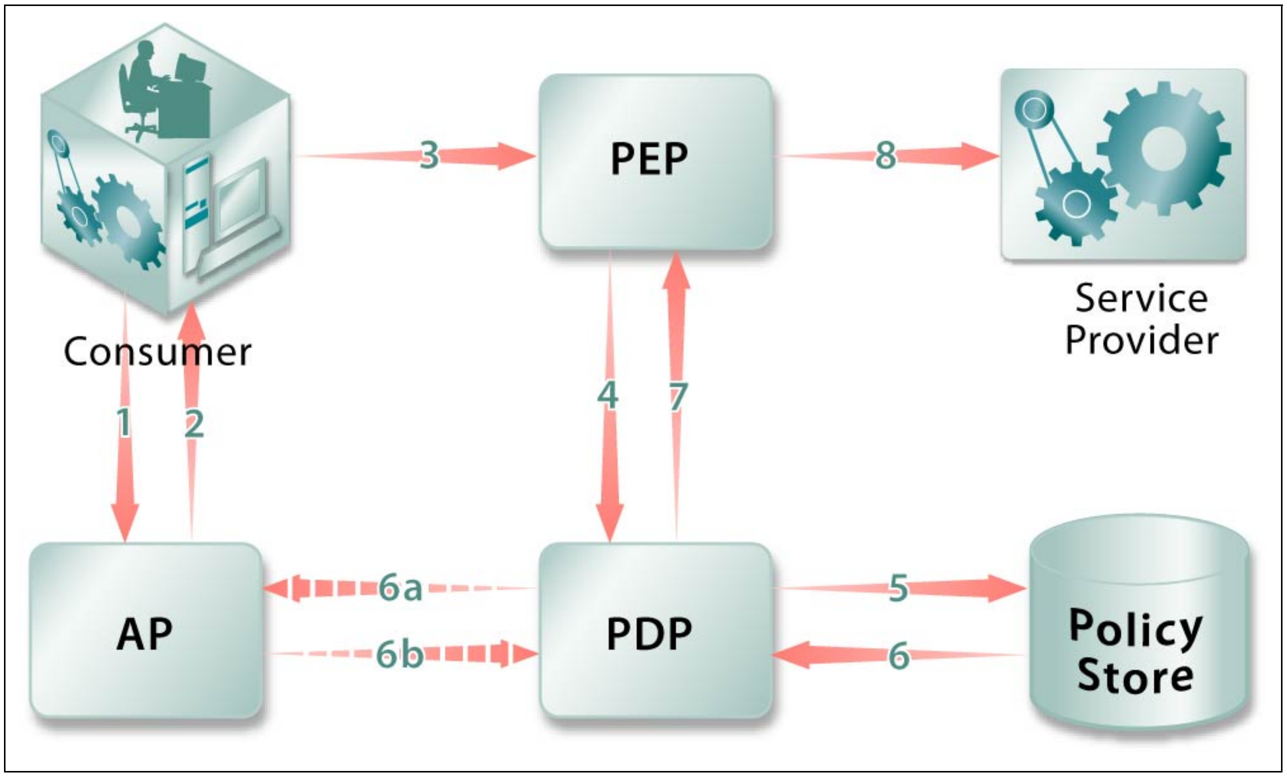


Figure 3–6: ABAC - Pre-Retrieval Authorization Model

This model represents a less traditional access control process. In it, there is some disassociation between all the access control steps and the functional nodes that traditionally perform access control. Specifically, it is the service consumer and not the PDP that pre-retrieves attributes necessary for the access control decision to be made. In a fully Net-Centric enterprise, where attributes of principals are centrally regulated, but de-centrally managed, the majority of ABAC implementations will support this model. As discussed in **Section 3.3** of this profile, this model can be categorized as having a “pre-retrieval” transaction pattern, because the attributes are retrieved prior to the transmission of a service request to a policy decision point.

Table 3–3: ABAC - Pre-Retrieval Authorization Model - Details

Step	Description
1	A service consumer requests a principal's attributes from an AP necessary for the service request.
2	The AP returns a signed attribute assertion to the service consumer.
3	The service consumer sends the attribute assertion and the service provider request to the PEP.
4	The PEP requests an authorization decision from the PDP, including the principal's attributes in the request. The PDP trusts the attribute assertion.
5	The PDP requests authorization policies from the Policy Store.
6	The Policy Store returns a policy. The PDP trusts the policies returned from the Policy Store. Step 6a: PDP validates attributes and requests additional attributes from the AP, if necessary. Step 6b: The AP returns an attribute assertion to the PDP, if necessary. The PDP trusts the assertions returned from the AP.
7	The PDP makes a decision based on the principal's attributes and the policies, and returns a decision assertion to the PEP.
8	The PEP either grants or denies access to the service provider, based on the decision assertion. • The PEP trusts the decision assertion generated by the PDP.

3.2.1.3 ABAC - Permit Authorization Model

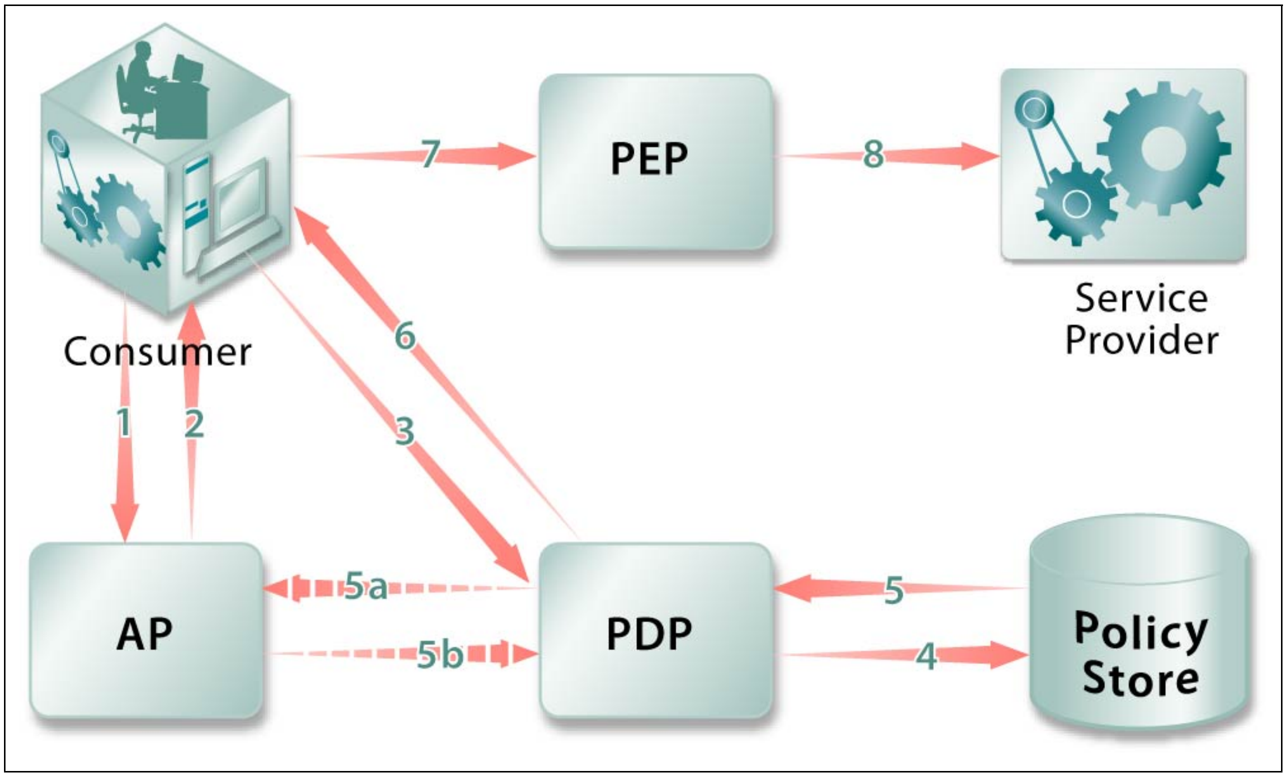


Figure 3–7: ABAC - Permit Authorization Model

This model, like the previous model, represents a less traditional access control process. Not only is the service consumer retrieving the attributes necessary for the access control decision to be made, but when the access control decision is made, it is sent to the consumer prior to any contact with the PEP that protects the resource. This model can be described as a "permit model" in that the decision assertion can be provided and treated in ways similar to a permit. In system environments where a PEP is functionally constrained by throughput or latency requirements, this model may be beneficial. Other instances in which it may be employed are when the PEP and protected resource are removed from the security infrastructure that manages the attributes and the access control policy. As discussed in **Section 3.3** of this profile, this model can be categorized as having a "pre-retrieval" transaction pattern, because the attributes are retrieved prior to the transmission of a service request to a policy decision point.

Table 3–4: ABAC - Permit Authorization Model - Details

Step	Description
1	A service consumer requests attributes from an AP necessary for the service request.
2	The AP returns a signed attribute assertion to the service consumer.
3	The service consumer sends the attribute assertion to the PDP along with the service request. • The PDP trusts the attribute assertions signed by the AP, which were provided by the service consumer.
4	The PDP requests authorization policies from the Policy Store.
5	The Policy Store returns a policy to the PDP. Step 5a: PDP may choose to validate attributes with the AP and request additional attributes if necessary. Step 5b: The AP returns an attribute assertion to the PDP, if necessary.
6	The PDP makes a decision based on the service consumer's attributes and the policies, and returns a signed decision assertion to the service consumer. • The PDP trusts the attributes returned from the AP. The PDP validates that the attribute assertion was not manipulated by the service consumer. • The PDP trusts the policies returned from the Policy Store.
7	The service consumer sends the decision assertion and the service provider request to the PEP.
8	The PEP either grants or denies access to the service provider, based on the decision assertion. • The PEP validates the decision assertion generated by the PDP.

3.2.1.4 ABAC – Permit Direct Retrieval Authorization Model

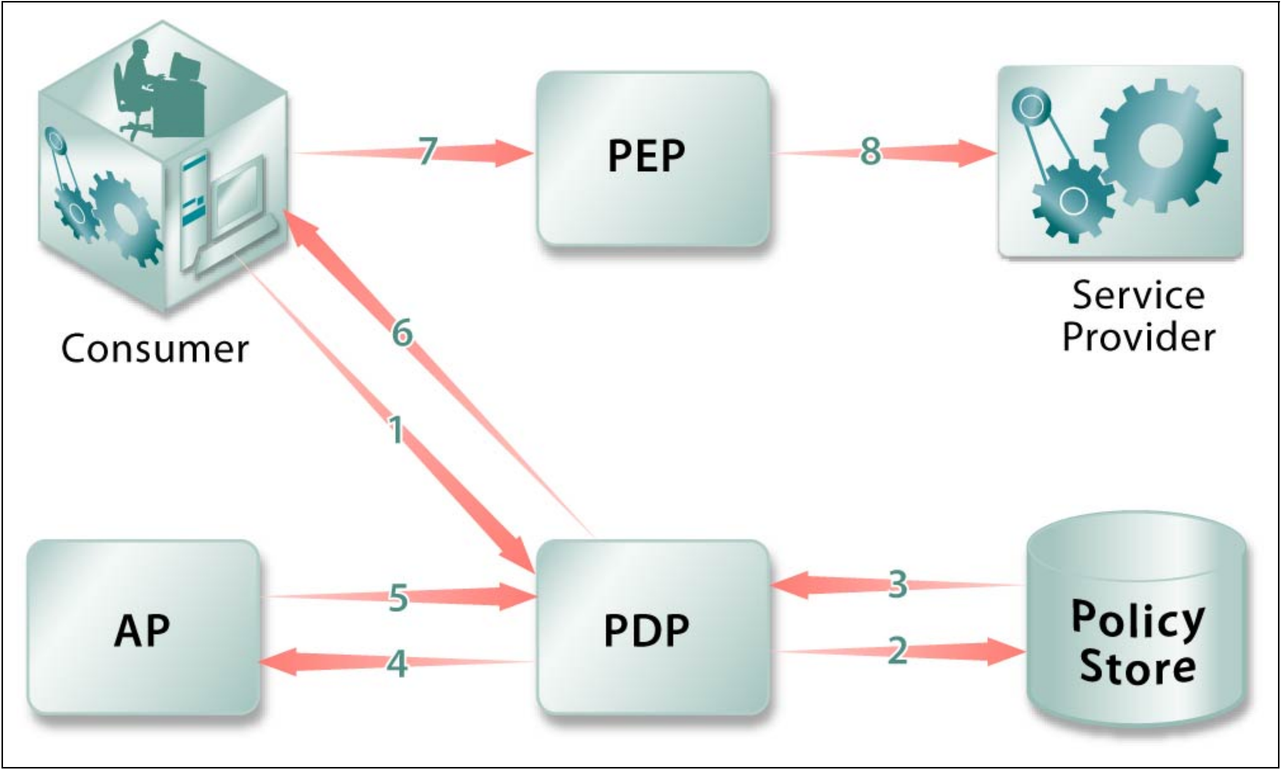


Figure 3–8: ABAC - Permit Direct Retrieval Authorization Model

This model, like the previous model, represents a “permit” model in that the decision assertion is given to the Consumer in advance of the utilization of the service. As with the Permit Model, this model may be beneficial in system environments where a PEP is functionally constrained by throughput or latency requirements. Other instances in which it may be employed are when the PEP and protected resource are removed from the security infrastructure that manages the attributes and the access control policy. The difference between this model and the previous one, however, is that it represents a “direct retrieval” transaction pattern, because the PDP asks for attributes directly from the AP. This pattern may be employed when the consumer as well as the PEP and protected resource are both removed from the security infrastructure, perhaps for operational reasons.

Table 3–5: ABAC - Permit Direct Retrieval Authorization Model - Details

Step	Description
1	A service consumer requests an authorization decision from the PDP.
2	The PDP requests authorization policies from the Policy Store.
3	The Policy Store returns applicable policies to the PDP.
4	The PDP requests attributes from the AP based on the service consumer's identity.
5	The AP returns attribute assertions with the relevant service consumer attributes to the PDP.
6	The PDP makes a decision based on the service consumer's attributes and the policies, and returns a signed decision assertion to the service consumer. • The PDP trusts the attributes returned from the AP. • The PDP trusts the policies returned from the Policy Store.
7	The service consumer sends the decision assertion and the service provider request to the PEP.
8	The PEP either grants or denies access to the service provider, based on the decision assertion. • The PEP validates the decision assertion generated by the PDP.

3.2.2 DNI IC SOA Authorization Models

The ODNI IC SOA Security Reference Architecture [ODNI-SOASRA] presents 13 ABAC Authorization Models - 6 of the 13 are classified as basic and foundational, and 7 are classified as advanced implementations. Of the 6 basic authorization models, 4 are similar to those described in the NCES Security Service Design Specification previously mentioned. In addition, ODNI SOA Security Reference Architecture includes 2 authorization models in which the Policy Store interacts directly with the Policy Enforcement Point. While these models are not included in the NCES Service Security Design Specification, the NCES Security Service architecture does not preclude support for these additional models.

The advanced authorization models describe how to enhance the 6 basic models to perform asymmetric and hierarchical policy enforcement. These models primarily impact access control policy and authorization decision flow rather than attribute information flow. However, for completeness, this profile accommodates the attribute information flow required by these additional authorization models.

The DNI authorization models are not articulated in the text of this profile at this time, but are available from the Office of the Director of National Intelligence, Chief Information Officer, and Intelligence Community Enterprise Architecture Service.

3.2.3 WS-XACML Authorization Models

In addition to the authorization models treated in the [NCESspec] and [ODNI-SOASRA] documents, this profile also recognizes 2 additional authorization models from analysis of the WS-XACML [WS-XACML] profile's **Section 3.2.3** use cases. While there are 9 distinct use case descriptions in that document, the models contained in this section result from a distillation of those transactional models into 2 basic WS-XACML use cases.

Unlike the NCES and the ODNI authorization models, the WS-XACML models have a focus on industry and commercial transactions. In particular, the WS-XACML draft standard focuses on the exchange of privacy attributes as well as the definition of policies relating to privacy and obligations. Because of this influence, the use cases it describes delve into trust in a different way than the NCES and ODNI use cases. There is an emphasis on the definition and delivery of policies to a policy store and the establishment of relationships of trust in these use cases. Although the means of trust are out of the scope of this profile, each of the use cases includes an articulation of the necessary trust relationships with the respective "Establish Trust" sections.

Additionally, entries within the descriptive tables that are in a larger, bold font are within the scope of this profile.

3.2.3.1 WS-XACML Use Case 1

Use Case 1 is characterized by three distinct phases: **Establish Trust, Define Policy,** and **Authorization**. The remainder of this section describes these three phases as well as the individual steps within each phase. **Figure 3–9** is a representation of all three phases.

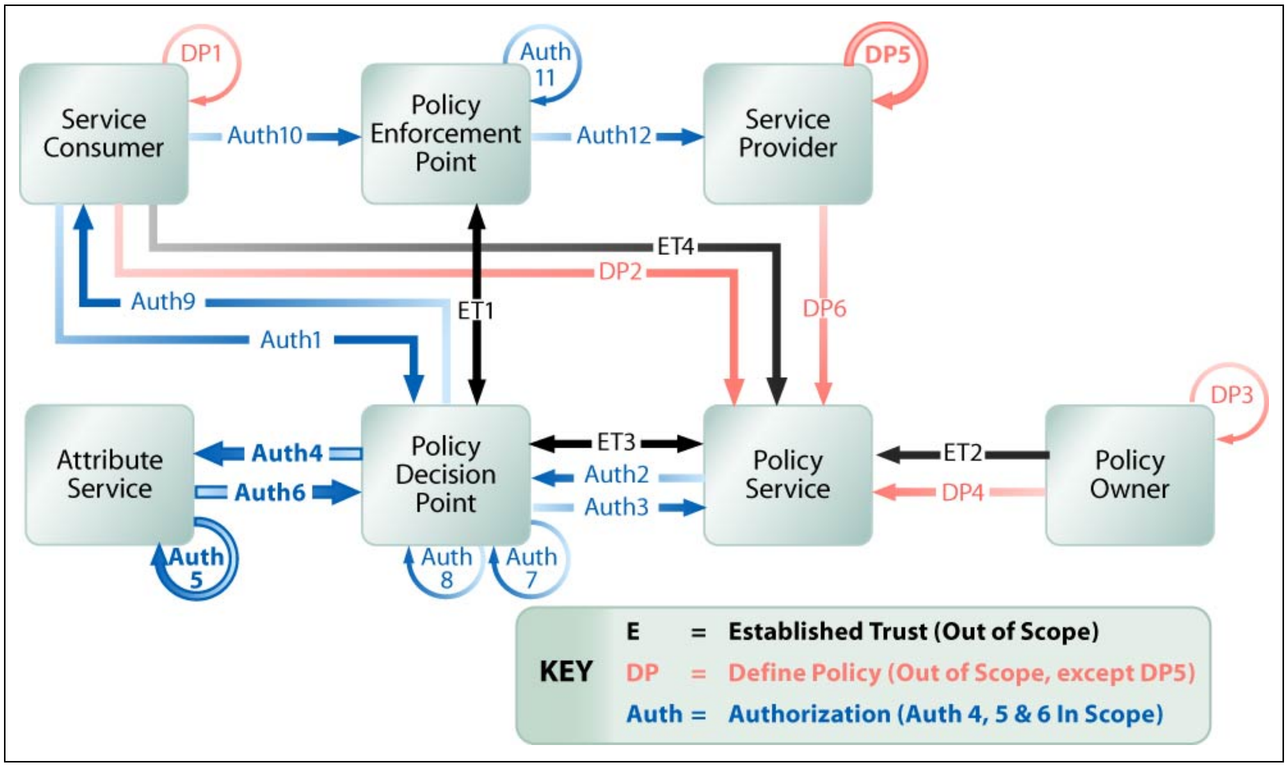


Figure 3–9: WS-XACML Use Case 1 - In-Scope and Out-of-Scope Transactions

Establish Trust: This series of prerequisite steps as shown in **Figure 3–10** describes the various trust relationships necessary before an authorization decision exchange according to WS-XACML and which is reliant upon exchange of attribute information according to the requirements of the SAML attribute profile.

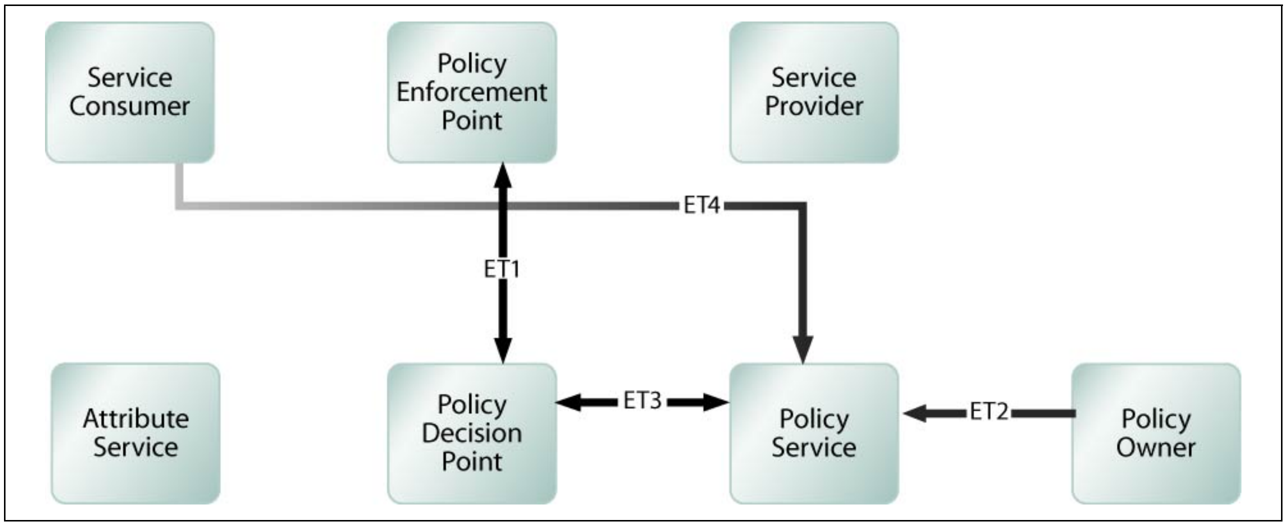


Figure 3–10: WS-XACML Use Case 1 - Establish Trust

The roots of trust may rely upon the exchange of root certificates to validate signature chains, or on some other method. Typically this occurs out of band and is required to validate any information exchanged between two parties. If digital signatures are used to convey validity information, the key used to verify the signature MUST be accessible. The trust transactions are described in the following table.

Table 3–6: WS-XACML Use Case 1 - Establish Trust (Details)

Step	Description
ET 1	The PEP must trust the decisions made by the PDP.
ET 2	The Policy Owner must trust the Policy Service or Policy Store to select the right policies to apply to a given authorization request.
ET 3	The PDP must establish trust with the Policy Service or Policy Store to supply the correct policies.
ET 4	The Service Consumer must trust the Policy Service or Policy Store to retrieve the appropriate client policies (e.g., privacy, capabilities) for each request.

Define Policy: All participants who seek to impose policy on a given interaction must publish their policy before the evaluation of any combination of policies within the context of an authorization decision. They must specify under what circumstances their policy applies, and determine how to make it accessible to any PDPs which much evaluate the policy. These transactions are depicted in **Figure 3–11**.

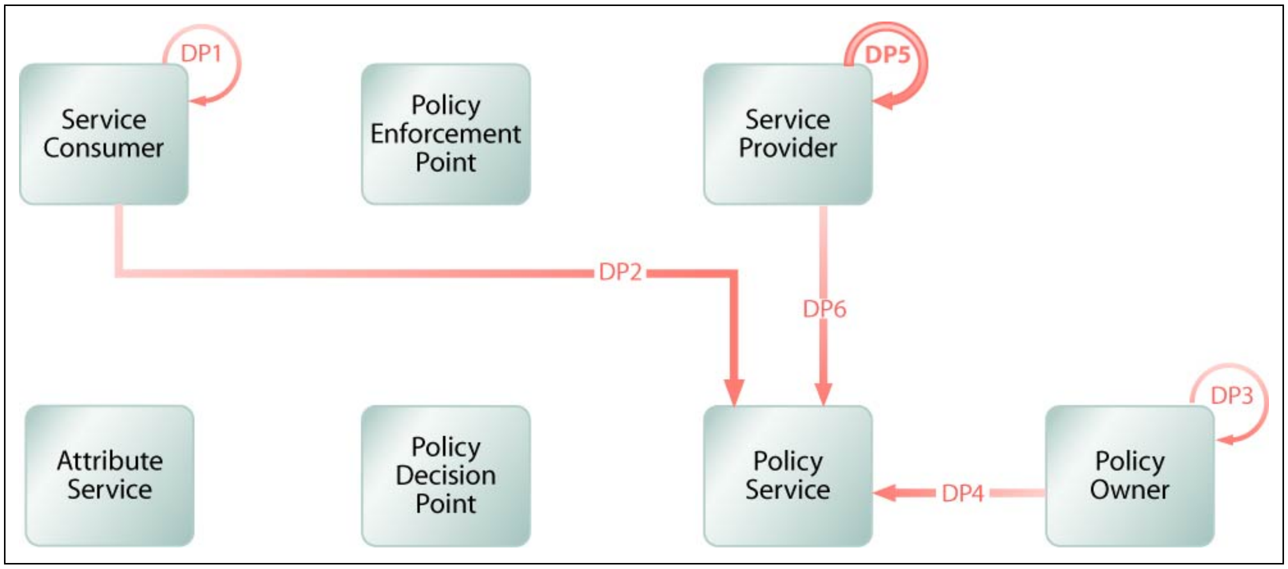


Figure 3–11: WS-XACML Use Case 1 - Define Policy

These transactions are further articulated in the following table. For each policy that is created, transmission of the policy to the policy store upon creation or update is required by the Policy Store. Because the NCES profiles publish their service provider access policies to the policy store as XACML-based SAML attributes, the step depicted by DP5 is within the scope of this profile. All other steps are not within the scope of this profile.

Table 3–7: WS-XACML Use Case 1 - Define Policy (Details)

Step	Description
DP 1	Service Consumer expresses interaction requirements (e.g., privacy and capabilities) as policy.
DP 2	Service Consumer publishes capabilities/requirements to a Policy Service or Policy Store so they are available to a PDP.
DP 3	Policy Owner of the security domain expresses policy requirements for the security domain as XACML policies.
DP 4	Policy Owner publishes policies to a trusted Policy Service or Policy Store.
DP 5	Service Provider expresses access policy requirements for the service as XACML policies.
DP 6	Service Provider publishes policies to a trusted Policy Service or Policy Store.

Authorization: This series of steps is the context for the request and return of attributes that will be used to evaluate policy to reach an access control decision. At this point, this authorization model is similar to the NCES Permit Direct Retrieval Authorization Model discussed in **Section 3.2.1.4**. These transactions are depicted in **Figure 3–12**.

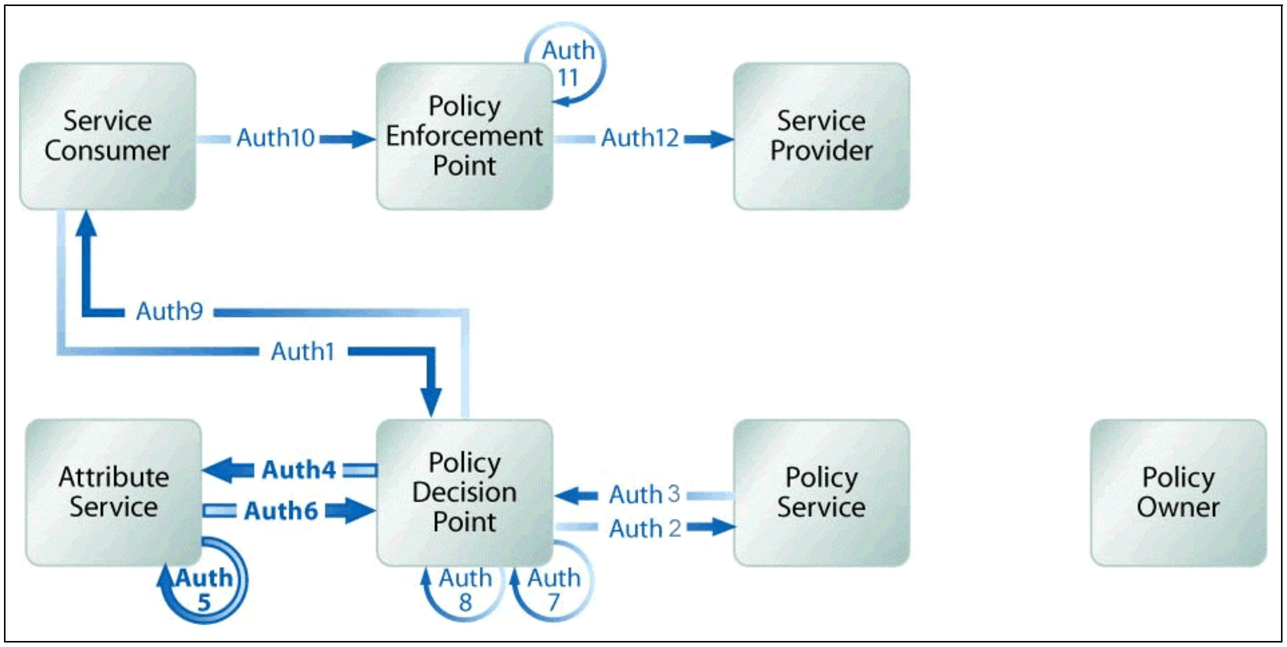


Figure 3–12: WS-XACML Use Case 1 - Authorization

Table 3–8: WS-XACML Use Case 1 - Authorization (Details)

Step	Description
Auth1	Service Consumer requests an authorization token from a PDP to access a service.
Auth 2	PDP retrieves policies applicable to the access request from the Policy Service or Policy Store.
Auth 3	Policy Service or Policy Store returns applicable policies to PDP for evaluation.
Auth 4	PDP requests attributes from Attribute Service (AS) relevant to policy evaluation.
Auth 5	AS determines appropriate attributes and values to return to PDP.
Auth 6	AS returns attributes and values to PDP.
Auth 7	PDP evaluates policy according to selected attributes.
Auth 8	PDP builds and signs authorization token.
Auth 9	PDP returns authorization token to requesting Service Consumer.
Auth 10	Service Consumer presents authorization token to PEP with service request.
Auth 11	Authorization token received by PEP and evaluated for legitimacy.
Auth 12	PEP allows requested access to Service Provider.

3.2.3.2 WS-XACML Use Case 2

Use Case 2 is characterized by three distinct phases: **Establish Trust, Retrieve Attributes,** and **Authorization**. The remainder of this section describes these three phases as well as the individual steps within each phase. **Figure 3–13** is a representation of all three phases.

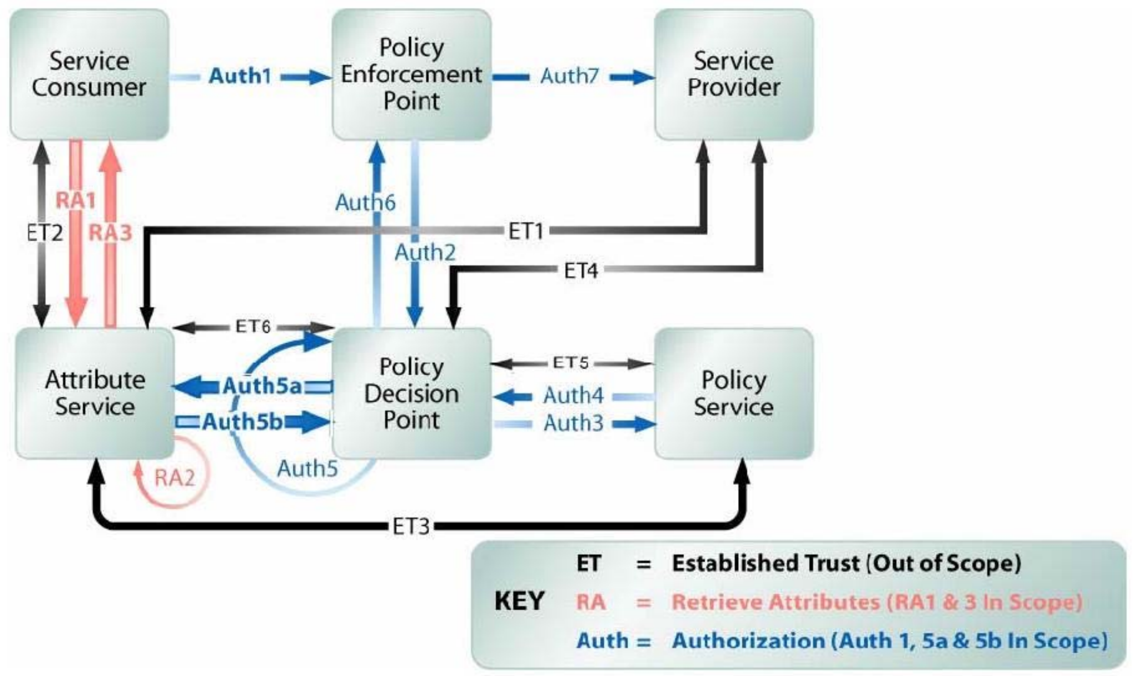


Figure 3–13: WS-XACML Use Case 2 - In-Scope and Out-of-Scope Transactions

Establish Trust: This series of prerequisite steps describes the various pre-acknowledged trust relationships required prior to the authorization decision exchange prescribed by the WS-XACML profile. That same series of steps is similarly reliant upon the exchange of required attribute information prescribed by the SAML attribute profile. For steps ET1 and ET2, the Service Provider and the Service Consumer, respectively, may need to determine from more than one Attribute Service available which one should maintain and serve their attribute information.

Although traditionally the roots of trust may rely on the exchange of root certificates to validate signature chains, other non-specified methods may be used as well. Typically this occurs out of band and is required to validate any information exchanged between two parties. These out-of-band exchanges are depicted in **Figure 3–14**.

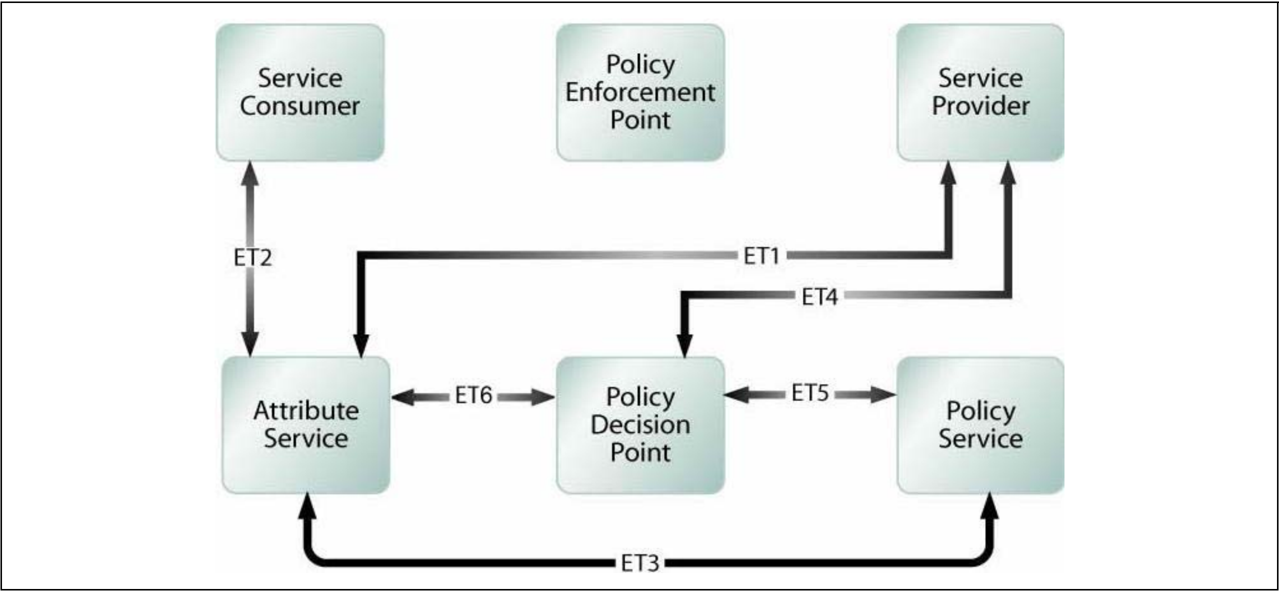


Figure 3–14: WS-XACML Use Case 2 - Establish Trust

If digital signatures are used to convey validity information, the key used to verify the signature must be accessible.

The establishment of trust is described in the following table.

Table 3–9: WS-XACML Use Case 2 - Establish Trust (Details)

Step	Description
ET1	The Service Provider (SP) must trust the Attribute Service (AS) as a steward of attribute information about the SP.
ET 2	The Service Consumer (SC) must trust the AS as a steward of attribute information about the SP.
ET 3	Trust between the AS and Policy Service or Policy Store must be established to ensure correct privacy and release policies are applied to attribute requests.
ET 4	The SP must trust the PDP to reach correct decisions.
ET 5	The PDP must establish trust with the Policy Service or Policy Store to supply the correct policies.
ET 6	The PDP must trust the AS.

Retrieve Attributes: The Service Consumer is responsible for requesting attributes from the appropriate Attribute Service. This fits the “pre-retrieval” category of transaction patterns as discussed in **Section 3** of this profile. The transactions involved are depicted in **Figure 3–15**.

The attribute release policies that are in place during Step RA2 may require the AS to return only a subset of attributes and/or values that may be applied to a given request. The AS may also sign the attributes before returning them to the Service Consumer.

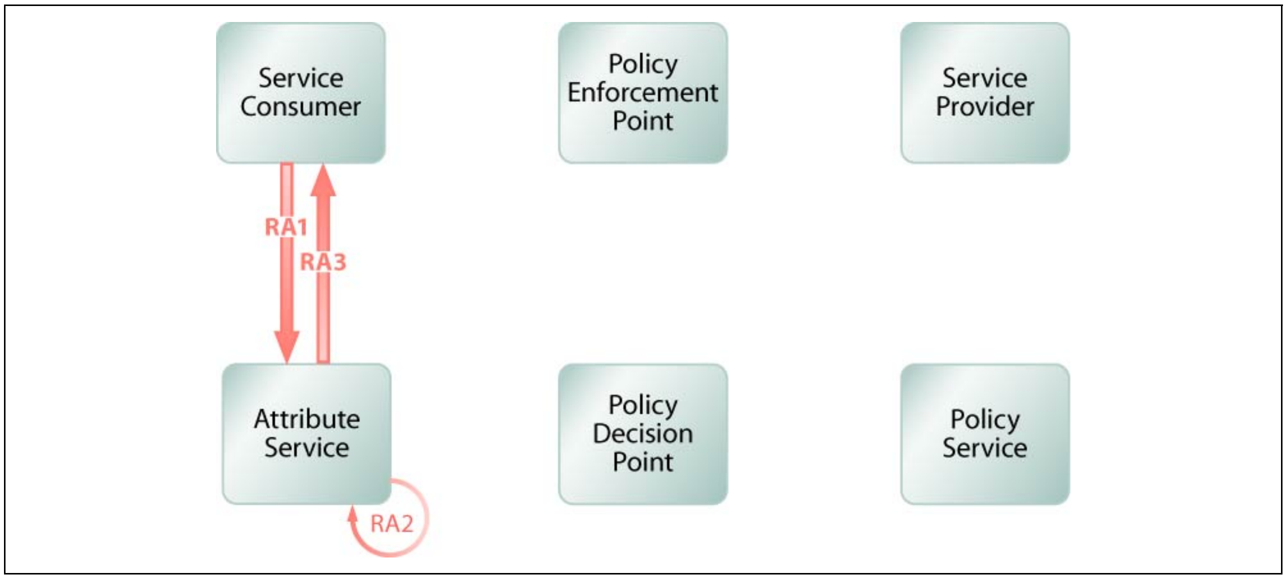


Figure 3–15: WS-XACML Use Case 2 - Retrieve Attributes

Table 3–10: WS-XACML Use Case 2 - Retrieve Attributes (Details)

Step	Description
RA1	Service Consumer requests attributes from Attribute Service.
RA2	AS evaluates authorization of the SC to receive attributes.
RA3	AS returns available and appropriate attributes to SC.

Authorization: This series of steps, as depicted in **Figure 3–16**, is the context for the request and return of attributes that will be used to evaluate policy to reach an access control decision. The first step, Auth1, does not stipulate whether the attributes should be bound to the request and SC through a digital signature, or by whom. WS-XACML introduces a requirement for management of some of the functional nodes. For example, in Auth2, a PEP may need to select from more than one available PDP to render the access control decision. In another example, the PDP may query multiple Policy Services or Policy Stores for applicable data to assist in making a decision.

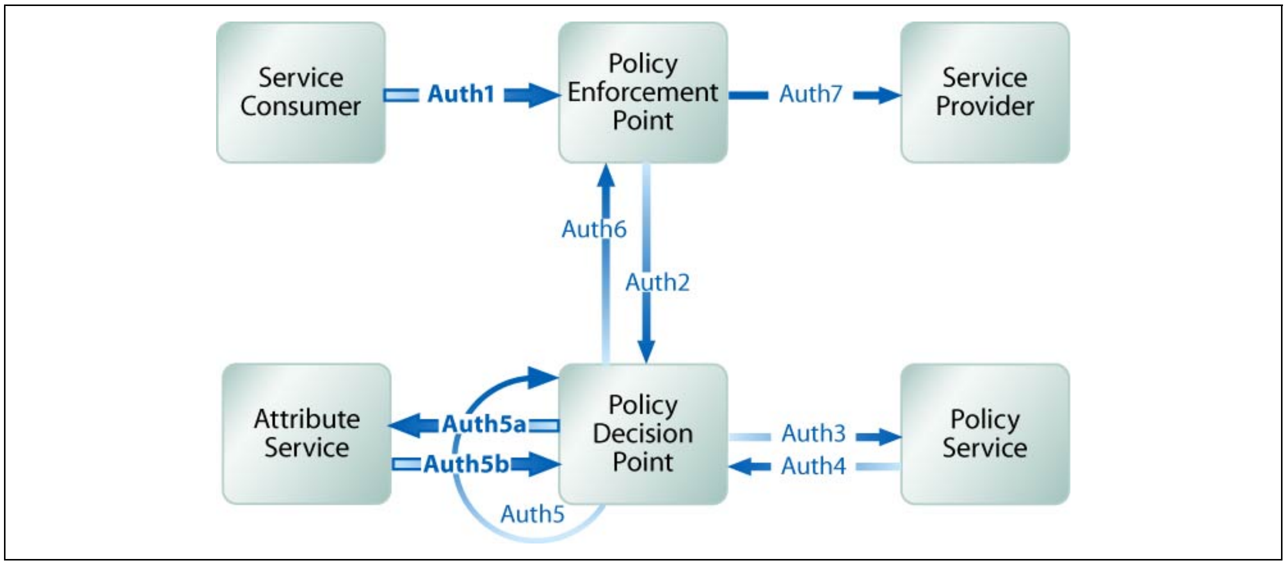


Figure 3–16: WS-XACML Use Case 2 - Authorization

Table 3–11: WS-XACML Use Case 2 - Authorization (Details)

Step	Description
Auth 1	The Service Consumer provides attributes to the PEP as part of the authorization decision request.
Auth 2	The PEP relies on a trusted PDP to evaluate the access control request and render a decision.
Auth 3	The PDP must retrieve the correct policies from the Policy Service or Policy Store to evaluate.
Auth 4	The Policy Service must return the appropriate policies to the PDP to evaluate.
Auth 5	The PDP must evaluate policies to render an authorization decision. Step 5a: If needed for the decision, the PDP can request additional attributes from the AS or from another AS. Step 5b: The AS returns the requested attributes to the PDP.
Auth 6	The PDP returns its decision to the PEP for enforcement.
Auth 7	The PEP permits access to the SP.

3.3 Attribute Transaction Flow

Based on the authorization models discussed in **Section 3.2**, this profile recognizes three categories of attribute retrieval patterns which are helpful in evaluating the attribute transaction flow:

- **Direct retrieval attribute pattern:** attributes required for an authorization decision are retrieved by a direct request from the system entity making the decision.
- **Pre-retrieval attribute pattern:** a service consumer's attributes are retrieved prior to the transmission of a service request to a policy decision point.
- **Tiered retrieval:** attributes required for an authorization decision are retrieved from secondary attribute authorities by an indirect request from the system entity making the decision. The request is brokered through one or more system entities, usually through attribute authorities in a tiered, hierarchical model.

Within the context of the authorization models from [NCESspec], [ODNI-SOASRA], and those from industry represented by the WS-XACML use cases, it is clear that complex attribute transactions will occur in implementations that support interactions between attribute consumers and asserting parties. In addition, operational conditions will influence architectural decisions relating to which authorization model is appropriate. It does not matter which model is adopted by a particular instantiation; certain attribute-related transactions will remain consistent across the models. These attribute transactions are described in this section and are represented in **Figure 3–17**.

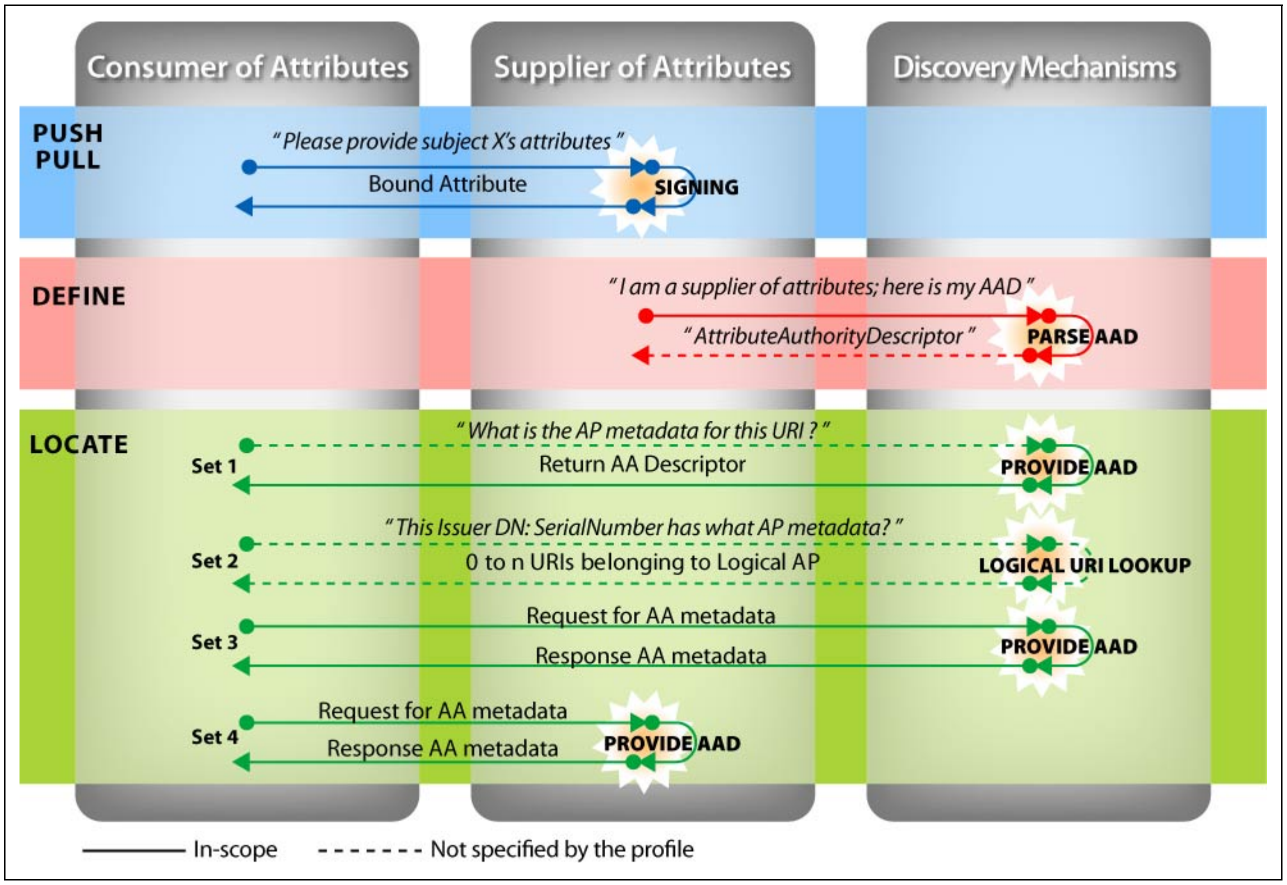


Figure 3–17: SAML Attribute Transaction Flows

3.3.1 Pull/Push Attributes

Attribute authorities maintain attributes about subjects and resources. Within the NCES Security Service Architecture, these authorities expose attributes through an Attribute Service.

The Push/Pull Attribute message transaction is the core of any attribute service that exists within the NCES Security Service Architecture, indeed, throughout all the authorization models. It is the interaction whereby attributes about a subject or a resource that is needed for an authorization decision are acquired by a node other than the attribute service. Whatever node this relying party is, it is considered within **Figure 3–17** to be the "Consumer of Attributes", as it is consuming the service which the attribute service provides. This profile assumes that a consumer of an attribute service will—in most cases be—the PDP that needs the attribute. Such cases correspond to the direct retrieval model. For pre-retrieval patterns, the consumer of the attributes may correlate to the authorization model's service consumer entity. In these pre-retrieval patterns, the consumer may collect their attributes prior to making their service request to the PEP. Additionally, within a tiered-retrieval pattern, the “Consumer of Attributes” may be another attribute service entirely.

The Push/Pull attribute message transaction depicts the request for attributes according to some metadata characteristic (e.g., a specific attribute identifier or all attributes within a given namespace). The two primary variations of this attribute flow are contingent upon the nature of the node acting as the “Consumer of Attributes”. In a “Pull” request, a PDP is the consumer and is more likely to request specifically named attributes (i.e., attributes that are directly referenced in the policy under evaluation). In a “Push” request, the relying party is some other system entity other than the PDP, most likely the consumer seeking to make the request. These non-PDP system entities are more likely to pull a range (e.g., all attributes within a given namespace) of attributes to include within a service request, as they may not have immediate access to the required access control policy. This is especially true in situations such as service chaining.

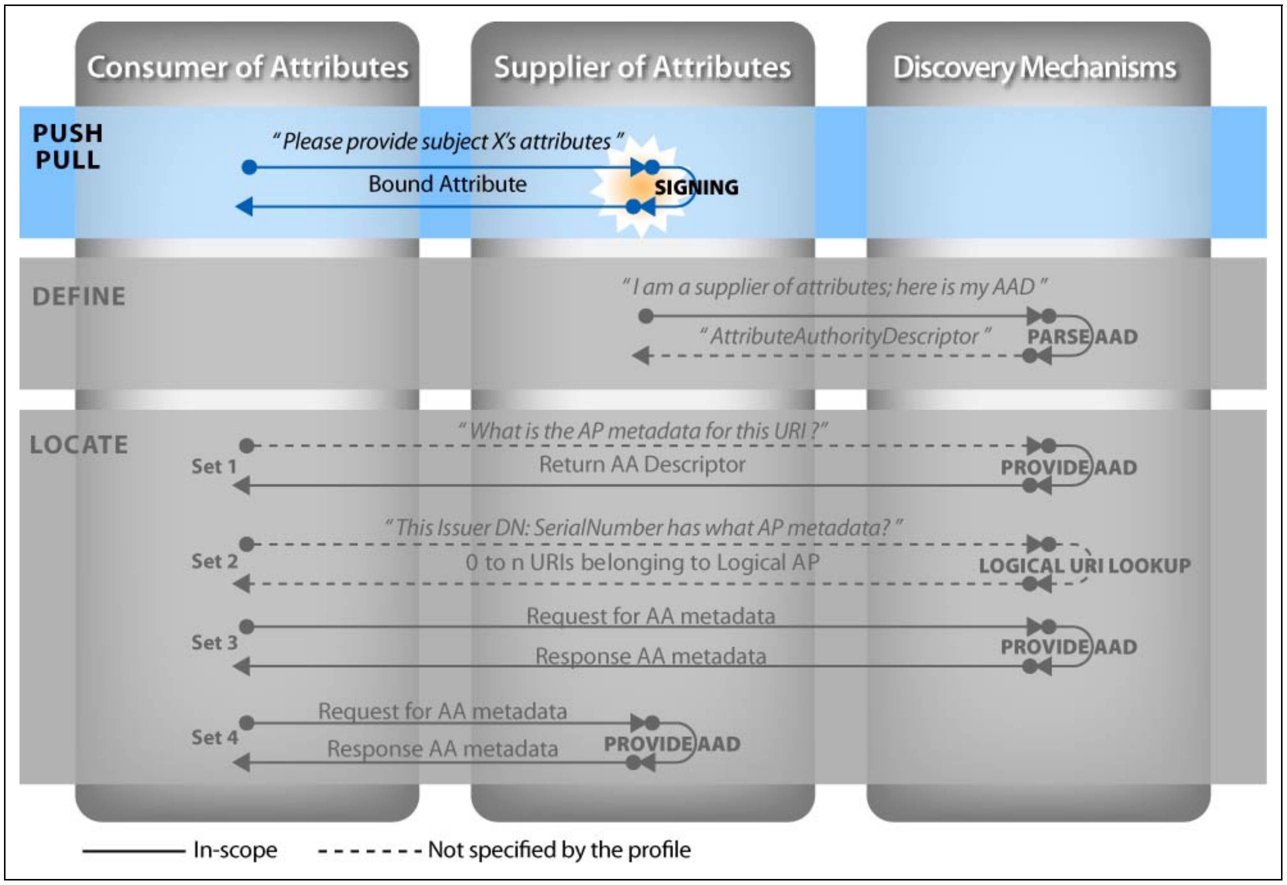


Figure 3–18: Pull and Push Attributes Information Flow

This profile defines a push of attributes as an attribute request/response that occurs prior to the resource request and thus defines the need to “push” attribute assertions with the resource request. The attribute transaction between consumer of attributes and attribute service is the same whether in a push or pull model. In some of the authorization models, the information flow pattern does not assume a specific request for attributes (or a set of attributes) from a PDP. What is depicted in **Figure 3–18** represents the same attribute transaction for push as for pull. What is different is the architectural state of pre-retrieval or direct retrieval. This transaction would be fundamentally similar whether using one of the pre-retrieval models (as in

Section 3.2.3.2, **WS-XACML Use Case 2** or **Section 3.2.1.3**, **ABAC Permit Authorization Model**), or a direct retrieval model (as in **Sections 3.2.1.1**, **ABAC Direct Retrieval Authorization Model**, or **3.2.3.1**, **WS-XACML Use Case 1**).

In order to extract attribute information, this profile recognizes three potential methods to express the subset of desired attributes within an attribute query: namespace, XML Path Language (XPath) or regular expression (REGEX). Care must be taken to ensure that XPath or REGEX expressions are not used to bypass the security posture of the attributes. The following non-normative diagrams represent the structural relationships for Attribute Pull requests in SAML 1.1 and SAML 2.0 as well as the structure of the resulting response. For each structural diagram in this document, the numbers in the upper right-hand corner indicate the minimum and maximum allowable number of instances of the XML element. If there is only one number, that number represents both the minimum and maximum.

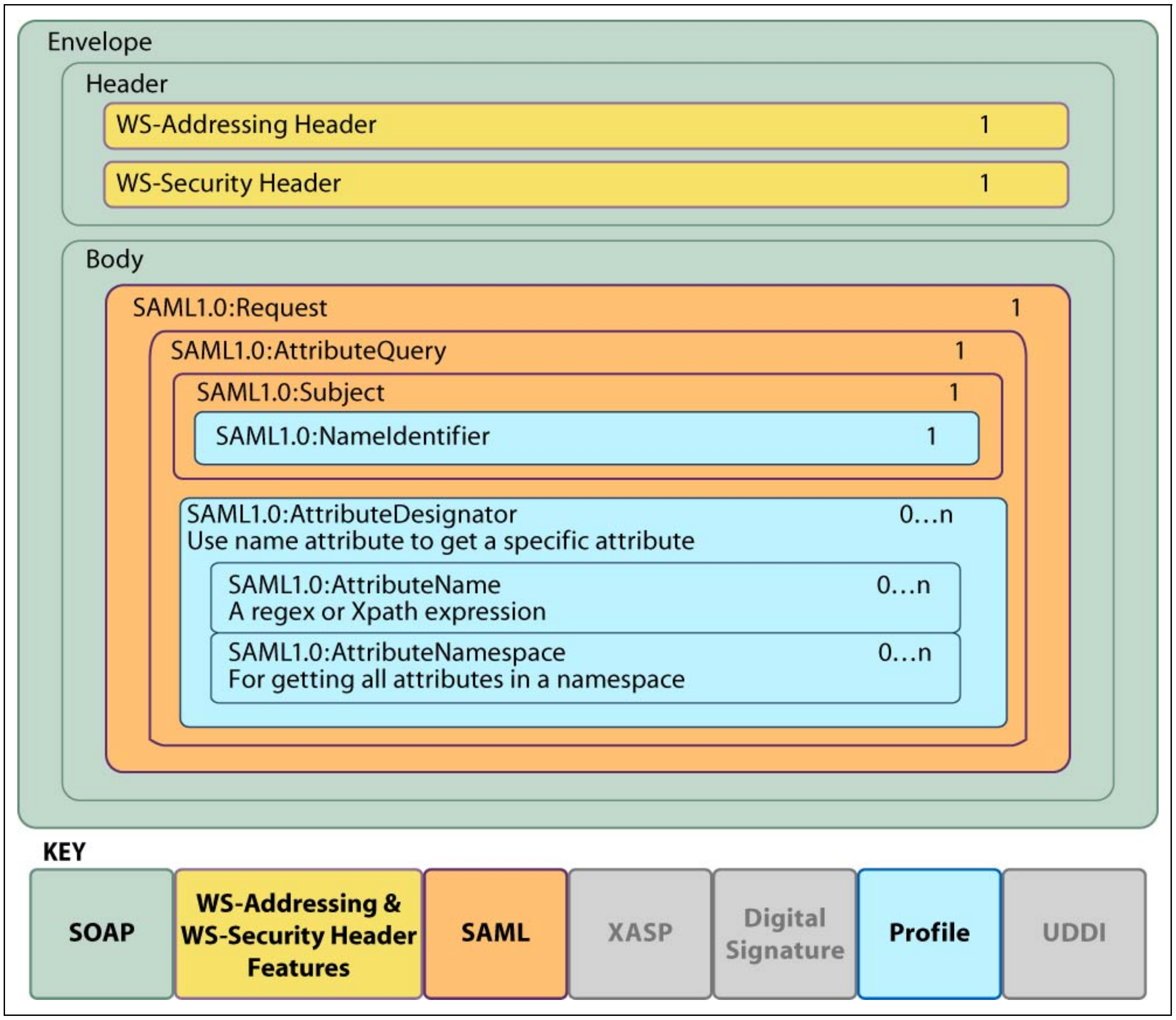

Envelope
Header
WS-Addressing Header 1
WS-Security Header 1
Body
SAML1.0:Request 1
SAML1.0:AttributeQuery 1
SAML1.0:Subject 1
SAML1.0:NameIdentifier 1
SAML1.0:AttributeDesignator 0…n
Use name attribute to get a specific attribute
SAML1.0:AttributeName 0…n
A regex or Xpath expression
SAML1.0:AttributeNamespace 0…n
For getting all attributes in a namespace
KEY
SOAP
WS-Addressing & WS-Security Header Features
SAML
XASP
Digital Signature
Profile
UDDI

Figure 3–19: Pull/Push SAML Version 1.1 Attribute Query

In **Figure 3–19**, the ***saml1:AttributeDesignator*** is the construct within a query for which a range of attributes can be expressed. The following sub-elements can be used to describe the set of attributes to be returned:

- The sub-element ***saml1:AttributeDesignator/@AttributeName*** can be populated with the REGEX expressions or XPath expression that will be evaluated against the requested set of attributes.
- The ***saml1:AttributeDesignator/@AttributeNamespace*** can be populated with the namespace(s) that will be queried for attributes.

If multiple namespaces and/or expressions are desired, or multiple individual attributes are identified, each query expression SHOULD be inserted in a separate ***saml1:AttributeDesignator***. (See **Section 4.4.1**)

Exhibit 3–1: SAML 1.1 Attribute Query by saml1:AttributeDesignator/@AttributeName

```
Line
Ref# Code
27   <samlp:AttributeQuery
28   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol">
29      <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
30
31         <saml:NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
32            format:x509SubjectName" NameQualifier="CN=John
33            Doe,OU=NCES,DC=DISA,DC=mil"/>
34         </saml:Subject>
35
36         <saml:AttributeDesignator
37            xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
38            AttributeName="Citizenship"
39            AttributeNamespace="urn:mil:disa:foo"/>
40
41         <saml:AttributeDesignator
42            xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
43            AttributeName="Clearance"
44            AttributeNamespace="urn:mil:disa:foo"/>
45
46         <saml:AttributeDesignator
47            xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
48            AttributeName="SCIControls"
49            AttributeNamespace="urn:mil:disa:foo"/>
50
51   </samlp:AttributeQuery>
52   <!-- SAML 1.1 query for John Doe's citizenship, clearance, and
53   scicontrols in mil:disa:foo namespace. -->
```

On lines 31- 33, the NameIdentifier is the value of the Subject Domain Name (DN) from the principal's X.509 certificate and a format with the value of `urn:oasis:names:tc:SAML:1.1:nameid-format:x509SubjectName`. The query is looking for all attributes with the names **"Citizenship**,**" "Clearance**,**"** or **"SCIControls"** within the `urn:mil:disa:foo` namespace.

Similarly, it is possible to query by an attribute's namespace. The following example shows a SAML 1.1 Attribute Query by saml1:AttributeDesignator/@AttributeNamespace.

Exhibit 3–2: SAML 1.1 Attribute Query by saml1:AttributeDesignator/@AttributeNamespace

```
Line
Ref# Code
62  <?xml version="1.0" encoding="UTF-8"?>
63  <samlp:AttributeQuery
64  xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol">
65     <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
66        <saml:NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
67           format:x509SubjectName" NameQualifier="CN=John
68           Doe,OU=NCES,DC=DISA,DC=mil"/>
69     </saml:Subject>
70
71  <saml:AttributeDesignator
72     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion" AttributeName="*"
73     AttributeNamespace="urn:mil:disa:foo"/>
74  </samlp:AttributeQuery>
75  <!-- SAML 1.1 query for all of John Doe's attributes in the
76  mil:disa:foo namespace. -->
```

This query is seeking all of the attributes within the `urn:mil:disa:foo` namespace for the principal specified on lines 66-68.

Unlike SAML 1.0 and 1.1, SAML 2.0 specifies the ability to add additional attributes to the attribute element from other namespaces. The format of a SAML 2.0 attribute query is illustrated in the following diagram:

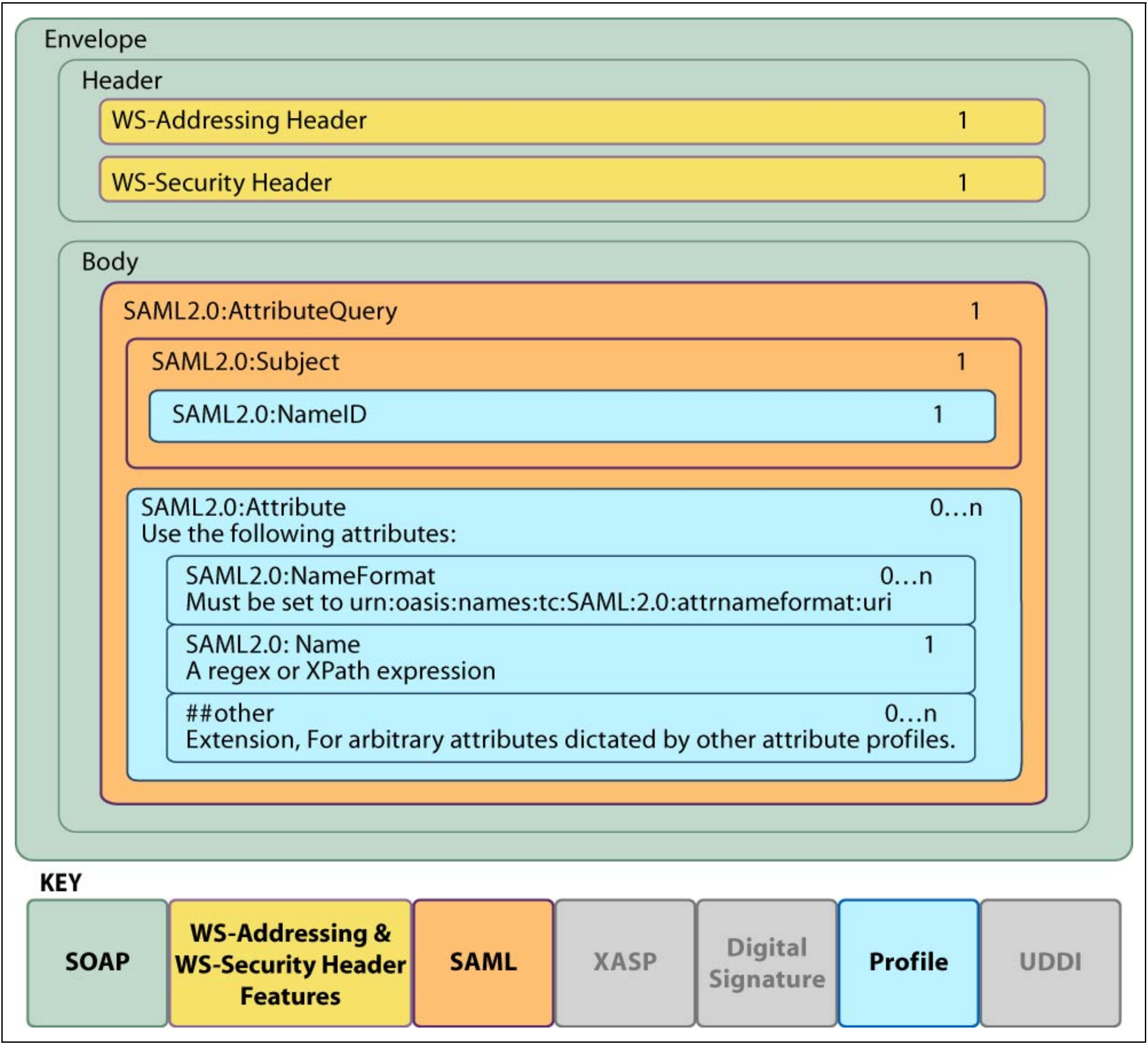

Envelope
Header
WS-Addressing Header 1
WS-Security Header 1
Body
SAML2.0:AttributeQuery 1
SAML2.0:Subject 1
SAML2.0:NameID 1
SAML2.0:Attribute 0…n
Use the following attributes:
SAML2.0:NameFormat 0…n
Must be set to urn:oasis:names:tc:SAML:2.0:attrnameformat:uri
SAML2.0: Name 1
A regex or XPath expression
##other 0…n
Extension, For arbitrary attributes dictated by other attribute profiles.
KEY
SOAP
WS-Addressing & WS-Security Header Features
SAML
XASP
Digital Signature
Profile
UDDI

Figure 3–20: Pull/Push SAML Version 2.0 Attribute Query

In **Figure 3–20**, the ***saml2:Attribute*** is the construct within which a query for a range of attributes can be expressed. The following sub-elements can be used to describe the set of attributes to be returned:

- The ***saml2:Attribute/@NameFormat*** MUST contain the value `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.
- The ***saml2:Attribute/@Name*** can be populated with the identifiers for specific attributes or with a REGEX or XPath expression that SHOULD be evaluated for a range of requested attributes. This value MAY be constructed to query for a fully-qualified URI that is unique to a specific attribute or a range of attributes within a namespace.

In addition, when a SAML 2.0 attribute construct is used, additional data type information SHOULD be included to comply with additional SAML profiles such as [X500] and [XACML]. If multiple namespaces and/or expressions are desired, or multiple attributes identified, then each query expression SHOULD be inserted in a separate ***saml2:Attribute*** construct (Reference **Section 4** for more details).

The following example shows a SAML 2.0 query which uses regular expressions in the SAML2:NameAttribute.

Exhibit 3–3: SAML 2.0 Query with Regular Expressions in the SAML2:NameAttribute

```xml
<?xml version="1.0" encoding="UTF-8"?>
<samlp:AttributeQuery xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
Version="2.0" ID="id-17984263" IssueInstant="2007-08-26T10:01:30.043Z" >
  <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
    <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
      format:x509SubjectName" NameQualifier="CN=John
      Doe,OU=NCES,DC=DISA,DC=mil"/>
    </saml:Subject>

  <saml:Attribute
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    Name="Citizen*"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>

  <saml:Attribute
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    Name="Clearance"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>

  <saml:Attribute
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    Name="SCIControl+"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>

</samlp:AttributeQuery>
<!-- SAML 2.0 query for John Doe's Clearance attribute as well as
attributes that meet the Citizen* and SCIContol+ regular expressions.
-->
```

This query is formatted to look for attributes that have one of the following characteristics:

- Contain an attribute name that begins with the **“Citizen”** prefix. Attributes named **“Citizenship”** or **“Citizen Since”** would fit this criteria.
- Contain an attributed named **“Clearance”**.
- Contain an attribute named **“SCIControl”** plus one additional character, **“SCIControls**,**”** would be a match.

The next example shows a SAML 2.0 query with constraints based on an attribute’s namespace.

Exhibit 3–4: SAML 2.0 Query with Constraints Based on the Attribute’s Namespace

```
Line
Ref# Code
139 <?xml version="1.0" encoding="UTF-8"?>
140 <samlp:AttributeQuery
141 xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" Version="2.0" ID="id-
142 17984263" IssueInstant="2007-08-26T10:01:30.043Z">
143   <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
144     <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
145       format:x509SubjectName" NameQualifier="CN=John
146       Doe,OU=NCES,DC=DISA,DC=mil"/>
147     </saml:Subject>
148
149   <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
150     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
151     Name="urn:mil:disa:foo:*"/>
152   </samlp:AttributeQuery>
153 <!-- SAML 2.0 query for all attributes for John Doe's in the
154 mil:disa:foo namespace. -->
```

The key lines in this example are lines 149-151 where the criteria for the attribute namespace must match the `urn:mil:disa:foo` namespace. The **“*”** within the Name attribute is a wild card value to allow all names to match this query so long as they exist in the `urn:mil:disa:foo` namespace. All attributes within this namespace are returned as a response to this query.

The next example shows a SAML 2.0 query with X500 (LDAP store) attribute information.

Exhibit 3–5: SAML 2.0 Query with X500 (LDAP Store)

```
<?xml version="1.0" encoding="UTF-8"?>
<samlp:AttributeQuery
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" Version="2.0" ID="id-
17984263" IssueInstant="2007-08-26T10:01:30.043Z">
  <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
    <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
      format:x509SubjectName" NameQualifier="CN=John
      Doe,OU=NCES,DC=DISA,DC=mil"/>
  </saml:Subject>

  <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
    FriendlyName="givenName" Name="urn:oid:2.5.4.42"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
    x500:Encoding="LDAP"/>

</samlp:AttributeQuery>
<!-- SAML 2.0 query for John Doe's givenName LDAP attribute identified
by urn:oid:2.5.4.42.-->
```

This query seeks the attribute uniquely identified as `urn:oid:2.5.4.42`, which has the friendly name of "givenName". The use of the namespace `urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500` and the encoding value of LDAP is required by [X500].

The last SAML 2.0 query example shows a query with X500 attribute information as well as SAML and XACML compliant data types.

Exhibit 3–6: X500 Attribute Information and [SAML-XACML] Compliant Data Types

```
Line
Ref# Code
186 <?xml version="1.0" encoding="UTF-8"?>
187 <samlp:AttributeQuery
188 xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" Version="2.0" ID="id-
189 17984263" IssueInstant="2007-08-26T10:01:30.043Z">
190   <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
191     <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
192       format:x509SubjectName" NameQualifier="CN=John
193       Doe,OU=NCES,DC=DISA,DC=mil"/>
194   </saml:Subject>
195
196     <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
197       xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
198       xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:
199       XACML"
200       FriendlyName="givenName" Name="ur:oid:2.5.4.42"
201       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
202       X500:Encoding="LDAP"
203       xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"/>
204
205 </samlp:AttributeQuery>
206 <!-- SAML 2.0 query for John Doe's givenName LDAP attribute →
```

The most interesting lines in this example are lines 196-203 as it is similar to the previous example for the X500 attribute with a friendly name of "givenName". In addition to the X500 attribute information, this construct also includes the appropriate [SAML-XACML] mapping information. On lines 198-199, the xacmlprof namespace is specified to be the SAML 2.0 Profile for XACML. On line 203, the XACML data type is set to **http://www.w3.org/2001/XMLSchema#string**.

The structure of the assertions provided in response to an attribute query is identical to the structure of an attribute assertion that is pushed (see **Section 4.5.1**).

There are well known compatibilities between the major versions of SAML (i.e., between 1.1 and 2.0). Managing transactions between nodes that utilize incompatible message transaction is out of the scope of this profile.

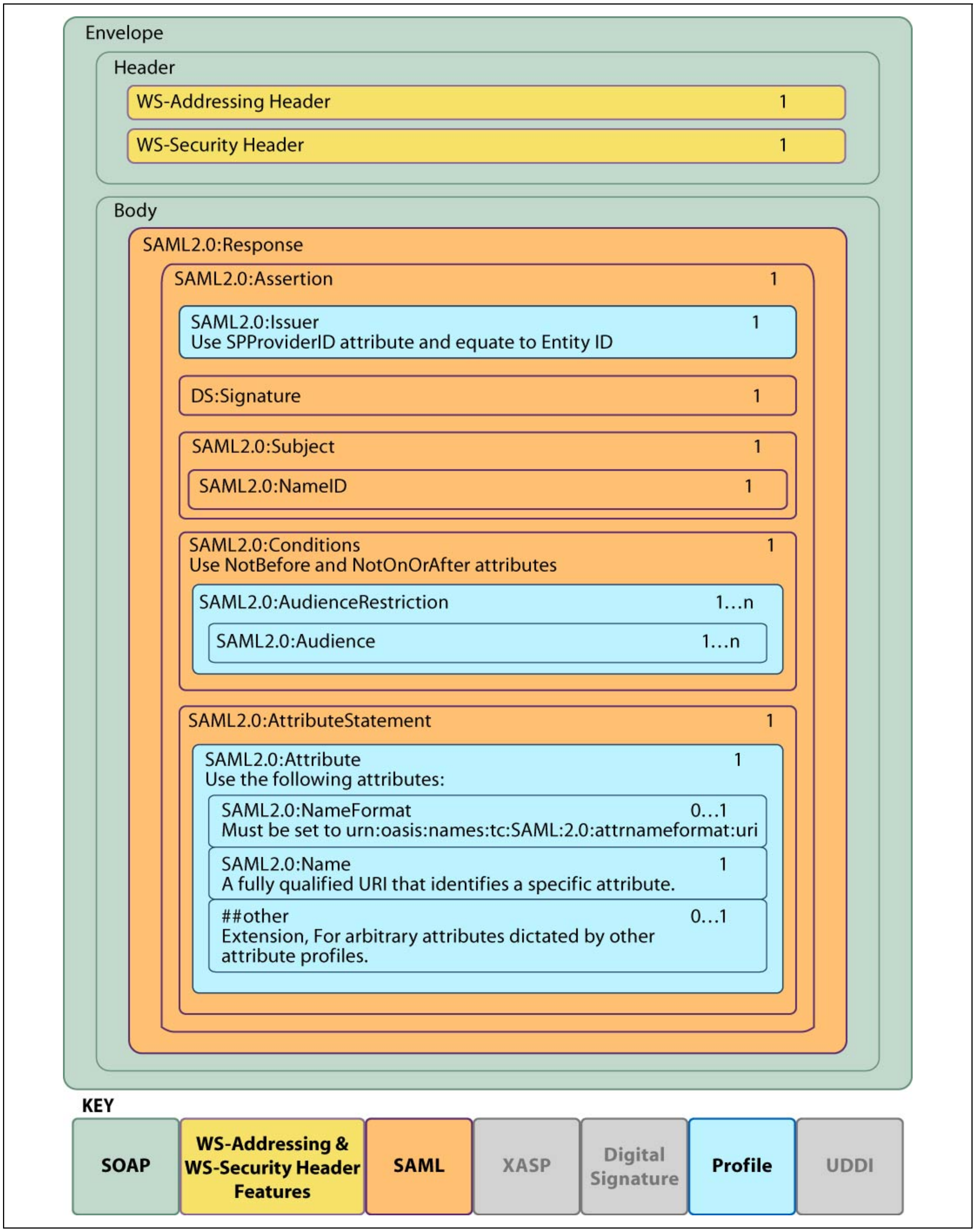

Envelope
Header
WS-Addressing Header 1
WS-Security Header 1
Body
SAML2.0:Response
SAML2.0:Assertion 1
SAML2.0:Issuer 1
Use SPProviderID attribute and equate to Entity ID
DS:Signature 1
SAML2.0:Subject 1
SAML2.0:NameID 1
SAML2.0:Conditions 1
Use NotBefore and NotOnOrAfter attributes
SAML2.0:AudienceRestriction 1…n
SAML2.0:Audience 1…n
SAML2.0:AttributeStatement 1
SAML2.0:Attribute 1
Use the following attributes:
SAML2.0:NameFormat 0…1
Must be set to urn:oasis:names:tc:SAML:2.0:attrnameformat:uri
SAML2.0:Name 1
A fully qualified URI that identifies a specific attribute.
##other 0…1
Extension, For arbitrary attributes dictated by other attribute profiles.
KEY
SOAP
WS-Addressing & WS-Security Header Features
SAML
XASP
Digital Signature
Profile
UDDI

Figure 3–21: Pull/Push Attribute Statement

As illustrated in the following exhibit, the ***saml2:AttributeStatement*** is the construct within which the resulting attributes are expressed. The following sub-elements describe each attribute returned as part of the set:

- The ***saml2:Attribute/@NameFormat*** MUST contain the value `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.
- The ***saml2:Name*** contains the fully qualified URI for the attribute.
- The ***##other attribute(s)*** indicate the appropriate interpretation of the data type of the values.
- The actual values returned.

The following exhibit is an example of a SAML 2.0 Attribute Statement:

Exhibit 3–7: SAML 2.0 Attribute Statement

```
<?xml version="1.0" encoding="UTF-8"?>
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" Version="2.0"
ID="Response-00011520" IssueInstant="2007-08-26T10:01:30.043Z" InResponseTo=”
ID_2d10e285-42d4-4926-984e-fab8ea72d32a”>

<samlp:Status>
   <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</samlp:Status>

<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      ID="A4061B4E-61E9-200F-6115-209A56B8E384"
      IssueInstant="2007-08-26T10:01:30.043Z" Version="2.0">

      <saml:Issuer
         Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
         SPProvidedID="urn:uniqueidentifier255orlesscharactersdeterminablefromAttr
         ibuteServicecertificateIssuerDNandSerialNumber"/>

      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
         <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm=".../REC-xml-c14n-20010315"/>
            <ds:SignatureMethod Algorithm=".../xmldsig#rsa-sha1"/>
            <ds:Reference URI="#A4061B4E-61E9-200F-6115-209A56B8E384">
               <ds:Transforms>
                  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
               </ds:Transforms>
               <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
               <ds:DigestValue>0pjZ1+TvgPf6uG7o+Yp3l2YdGZ4=</ds:DigestValue>
            </ds:Reference>
         </ds:SignedInfo>

         <ds:SignatureValue>
             CwP3qte8VosbgUnQnF+V6/knZgxRhR33=
         </ds:SignatureValue>

         <ds:KeyInfo>
            <ds:X509Data>
               <ds:X509Certificate>
                  MIICEDCCAX2gAwIBAgIQimXeUAxYJbJMady9vV1bLjAJBgUrDgMCHQUAMBIxEDA
                  OBgNVBAMTB1Rlc3QgQ0EwHhcNMDMwODE1MDcwMDAwWhcNMDUwODE1MDY1OTU5Wj
                  ArMSkwJwYDVQQDEyBBbGljZSBBYXJkdmFyayBPPUFsaWNlIENvcnAgQz1VUzCBn
                  zANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA0nIsmR+aVW2egl5MIfOKy4HuMKkk
                  9AZ/IQuDLVPlhzOfgngjVQCjr8uvmnqtNu8HBupui8LgGth06U9D0CNT5mbmhIA
```

Line Ref#	Code

```
                  ErRADUMIAFsi7LzBarUvNWTqYNEJmcHsAUZdrdcDrkNnG7SzbuJx+GDNiHKVDQg
                  gPBLc1XagW20RMvokCAwEAAaNWMFQwDQYDVR0KBAYwBAMCBkAwQwYDVR0BBDwwO
                  oAQAaVOkaVLLKoFmLN37pC8uqEUMBIxEDAOBgNVBAMTB1Rlc3QgQ0GCEC4MndUX
                  jPG1TZxVKg+HutAwCQYFKw4DAh0FAAOBgQABU91ka7IlkXCfv4Zh2Ohwgg2yObt
                  Y3+6C/BTFGrOEBJDy+DoxJ/NuBF18w3rrrR18xE6jNKYLCQb8zUGk4QOG5Y+HT/
                  QTTFvWkiOLXcpTuhnOhXatr42FoYpDkjx2QWK+J5Q2l/Rgjgc/0ZV8U/kD8UuRk
                  Xp4AZh7QsiX8AcO0w==
               </ds:X509Certificate>
            </ds:X509Data>
         </ds:KeyInfo>
      </ds:Signature>

      <saml:Subject>
         <saml:NameID
               Format="urn:oasis:names:tc:SAML:1.1:nameid-format:x509SubjectName"
         NameQualifier="CN=John Doe,OU=NCES,DC=DISA,DC=mil"/>
      </saml:Subject>

      <saml:Conditions NotBefore="2007-08-26T10:01:30.043Z"
                       NotOnOrAfter="2007-08-6T10:11:30.043Z">
         <saml:AudienceRestriction>
            <saml:Audience>
            urn:uniqueidentifier255orlesscharactersdeterminablefromAttributeService
            rtificateIssuerDNandSerialNumber</saml:Audience>
         </saml:AudienceRestriction>
      </saml:Conditions>

      <saml:AttributeStatement>
      <saml:Attribute
         xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
         Name=" urn:mil:disa:foo:Citizenship"
         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
         xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string">
         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">USA
         </saml:AttributeValue>
      </saml:Attribute>

      <saml:Attribute
      xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
         Name=" urn:mil:disa:foo:Clearance"
         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
         xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string">
         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">TS
         </saml:AttributeValue>
      </saml:Attribute>

   <saml:Attribute
   xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
      Name=" urn:mil:disa:foo:SCIControls"
      NameFormat=" urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string">
      <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">CONTROL A
      </saml:AttributeValue>
   </saml:Attribute>
      </saml:AttributeStatement>
```

```
Line
Ref#  Code
331      </saml:Assertion>
332   </samlp:Response>
333
```

On lines 231-232, the ***samlp:Response/@InResponseTo*** attribute is given to prevent replay attacks. The value of this attribute exactly matches the request identification data (ID). Please see **Section** 4**, Profile Requirements**, for more detail. On lines 244-245, the assertion's issuer has the same SPProvidedID as the SAML metadata EntityID.

On line 251, the digital signature references the assertion which began on line 238.

On lines 284-286, the subject NameID is the value of the subject DN from the principal’s X.509 certificate and a format with the value of `urn:oasis:names:tc:SAML:1.1:nameid-format:x509SubjectName`.

The conditions sited on line 289-290 include the mandatory NotBefore and NotOnOrAfter attributes to prevent replay attacks. In addition, the audience element contained on lines 292-294 includes the same EntityID referenced on line 244 but has no authority over the processing of the assertion.

The attribute statement begins on line 298 and contains three attributes with [SAML-XACML] mapping information. On lines 299-308, the citizenship attribute in the `urn:mil:disa:foo` namespace for John Doe is returned with a value of “USA”. Similarly, on lines 310-319, John Doe's clearance attribute is returned with a value of "TS" and on 321-330 the SCI control attribute with a value of "CONTROL A" is returned.

The following example shows an excerpt from an incorrect attribute statement. The example is incorrect because the assertion’s issuer does not match the attribute authority descriptor’s EntityID.

Exhibit 3–8: Incorrect Attribute Statement

```
Line
Ref#  Code
349   <?xml version="1.0" encoding="UTF-8"?>
350   <samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
351   Version="2.0" ID="Response-00011520" IssueInstant="2007-08-
352   26T10:01:30.043Z" InResponseTo=” ID_2d10e285-42d4-4926-984e-fab8ea72d32a”>
353      <samlp:Status>
354         <samlp:StatusCode
355      Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
356      </samlp:Status>
357
358   <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
359         ID="A4061B4E-61E9-200F-6115-209A56B8E384" IssueInstant="2007-08-
360         26T10:01:30.043Z" Version="2.0">
361         <saml:Issuer Format="urn:oasis:names:tc:SAML:1.1:nameid-
362            format:unspecified"
363            SPProvidedID="https://IdentityProvider.com/SAML/AA/SOAP"/>
```

The error exists on line 363 in the XML previous excerpt. The correct value for SPProvidedID would be `urn:uniqueidentifier255orlesscharactersdeterminablefromAttributeServicecertificateIssuerDNandSerialNumber`.

3.4 Metadata Flow

In addition to the push/pull traffic for actual attribute information, it is necessary to communicate peripheral data about the attributes. These communication flows are described as “metadata flow” because their sole purpose is transporting data about data.

An attribute authority must publish metadata about their service offers to prospective consumers. This information must be made accessible to service consumers via a discovery mechanism (e.g., through registration with a Universal Description Discovery and Integration [UDDI]-compliant registry). Beyond general service taxonomy information, Metadata for the OASIS SAML Working Draft [SAML-MD] defines specific metadata relating to several entities; most relevant to this profile are descriptors for attribute authorities and role descriptors. This metadata allows attribute authorities to communicate location endpoints, keys, attribute availability, and other metadata characteristics relevant to the exchange of attributes.

This profile aims to be flexible with the concept of what constitutes a supplier of attributes, such that consumers can act as suppliers of their own attributes and attribute metadata in keeping with the [WS-XACML] notion of obligation, publication, and distribution. The specific means by which this is done is out of the scope of this profile. Nonetheless, the profile was conceived with the idea that entities can act as both a “Consumer of Attributes” and “Supplier of Attribute” entities as seen in the transaction sequence diagrams within **Section 3.3**.

3.4.1 Defining Attribute Authority Policies

Within the NCES architecture, the discovery mechanism is the NCES Service Discovery Service which relies upon UDDI. This profile RECOMMENDS that NCES Service Discovery Service be considered the primary mechanism for implementing both the registration function and the discovery function.

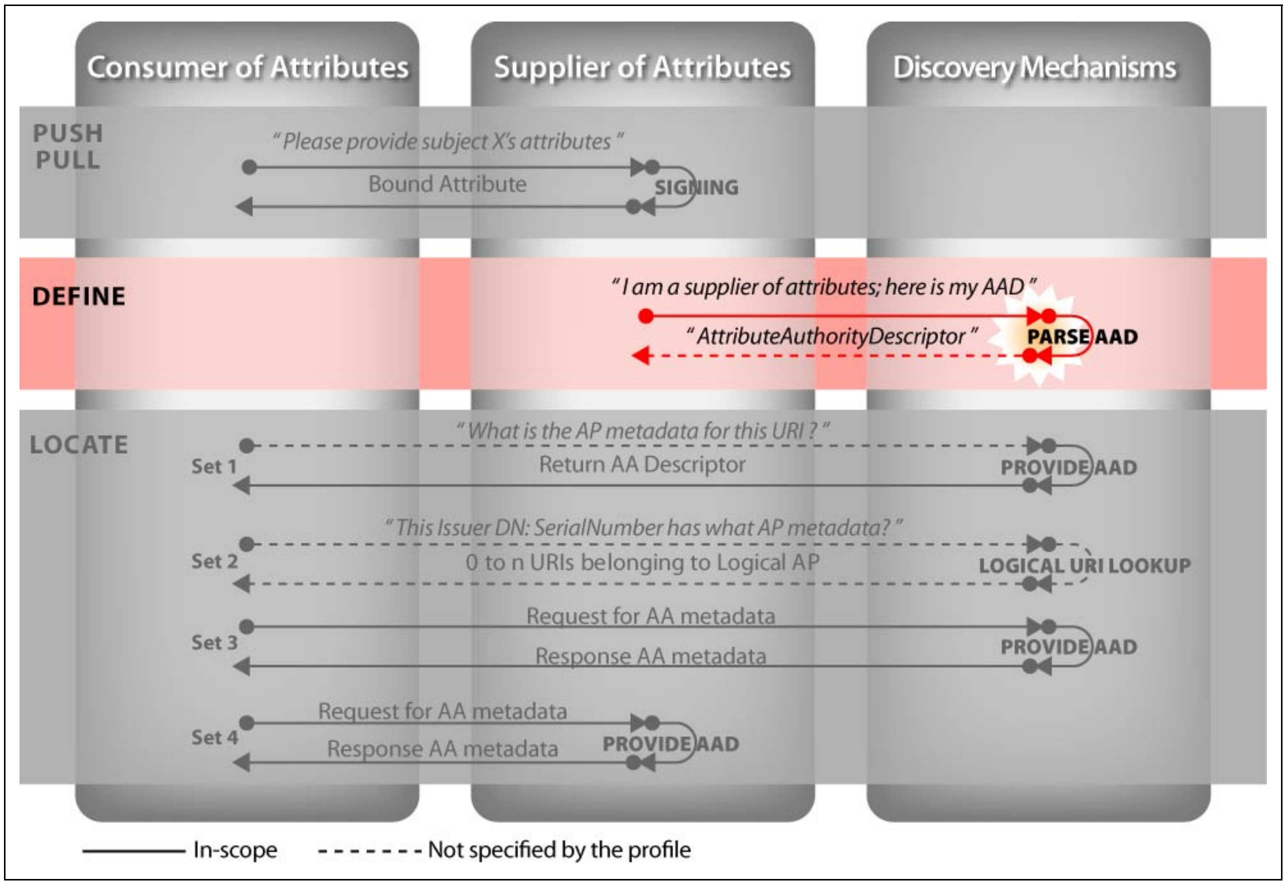


Figure 3–22: SAML Attribute Authority Descriptor Transaction

Within the information flow represented in **Figure 3–23**, the ***md:AttributeAuthorityDescriptor*** is used to illustrate the roles of system entities within the architecture. The use of this descriptor is not meant as a mechanism to convey trust.

The logical composition of a message conveying high level attribute authority metadata is depicted in the following diagram. ***saml1:AttributeNamespace***, ***saml2:Name****,* and ***##other*** are logically depicted in the diagram as a means to provide attribute consumers the greatest degree of flexibility in attribute retrieval and subsequent usage.

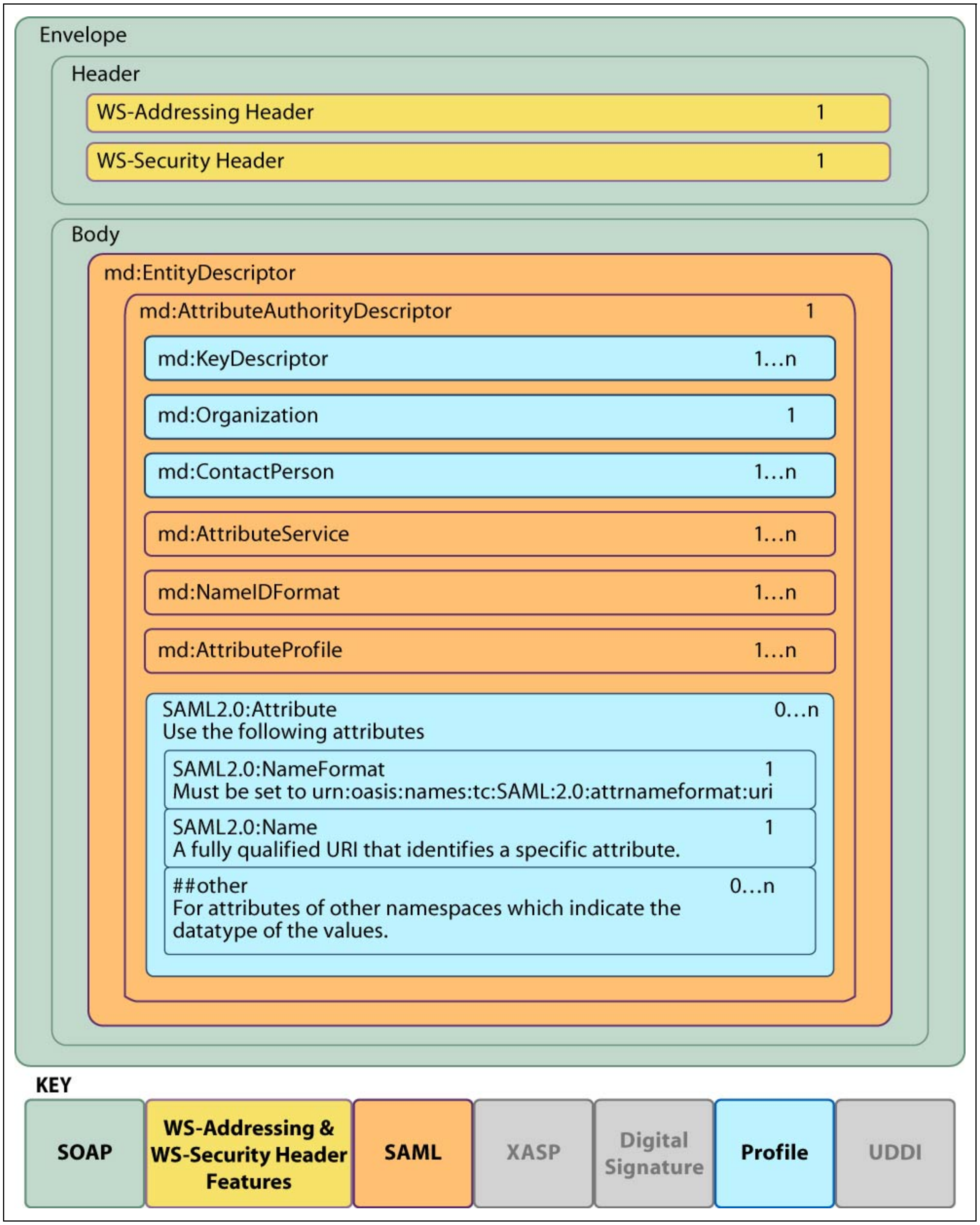


Figure 3–23: Define - Attribute Authority Descriptor

To publicize support for various attribute query mechanisms, this profile describes three distinct expressions of ***saml:Attribute***.

- The ***saml2:NameFormat*** denotes the format of the ***saml2:Name*** attribute.
- The ***saml2:Name*** denotes the ability to request specifically identified attributes.
- The ***##other*** should be used to delineate support for specific data types as part of the attribute values.

The following example illustrates a SAML 2.0 Attribute Authority Descriptor:

Exhibit 3–9: SAML 2.0 Attribute Authority Descriptor – Example 1

```
<?xml version="1.0" encoding="UTF-8"?>
   <md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
      cacheDuration="PT30.000S"
      entityID="urn:uniqueidentifier255orlesscharactersdeterminablefromA
      ttributeServicecertificateIssuerDNandSerialNumber">

   <md:AttributeAuthorityDescriptor
      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
      <md:Extensions/>
      <md:KeyDescriptor>
         <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:X509Data>
               <ds:X509IssuerSerial>
                  <ds:X509IssuerName>
                     cn=Some  Certificate,ou=PKI,ou=DoD,o=U.S.
                     Government,c=US
                  </ds:X509IssuerName>
                  <ds:X509SerialNumber>42</ds:X509SerialNumber>
               </ds:X509IssuerSerial>
            </ds:X509Data>
         </ds:KeyInfo>
      </md:KeyDescriptor>

      <md:Organization>
         <md:OrganizationName xml:lang="en">
            Foo Attribute Service Provider
         </md:OrganizationName>
         <md:OrganizationDisplayName xml:lang="en">
            Foo Attribute Service Provider at Some Location
         </md:OrganizationDisplayName>
         <md:OrganizationURL xml:lang="en">
            htt://www.foo.com
         </md:OrganizationURL>
      </md:Organization>

      <md:ContactPerson contactType="support">
         <md:SurName>Foo Support</md:SurName>
         <md:EmailAddress>foo-support@nsa.gov</md:EmailAddress>
      </md:ContactPerson>
      <md:AttributeService
      Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
      Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
         <md:NameIDFormat>
            urn:oasis:names:tc:SAML:1.1:nameid-format:x509SubjectName
         </md:NameIDFormat>
      <md:AttributeProfile/>

```

Line Ref#	Code

```
      <saml:Attribute
         xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
         Name=" urn:mil:disa:foo:Citizenship"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"

       <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:type="xs:string">USA</saml:AttributeValue>

       </saml:Attribute>

   <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
       xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
       Name="urn:mil:disa:foo:Clearance"
       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
       xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string">

       <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">TS</saml:AttributeValue>
       <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">S</saml:AttributeValue>
       <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">C</saml:AttributeValue>
       <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">U</saml:AttributeValue>
      </saml:Attribute>

     <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
      Name="urn:mil:disa:foo:SCIControls"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string">
         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">CONTROL A</saml:AttributeValue>
         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">CONTROL B</saml:AttributeValue>
         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">CONTROL C</saml:AttributeValue>
         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">CONTROL D</saml:AttributeValue>
         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="xs:string">CONTROL E</saml:AttributeValue>
      </saml:Attribute>

    </md:AttributeAuthorityDescriptor>
</md:EntityDescriptor>
```

The entityID on line 417-418 is the metadata entityID which must be 255 characters or less and derivable from the attribute service certificate's issuer DN and serial number. The attribute authority descriptor beginning on line 420 includes the protocols it supports as well as a key descriptor, organization, and contact information. This particular attribute authority descriptor is provided by the "Foo Attribute Service Provider”. In order to get support, "Foo Support," is the contact person reachable via email at **foo-support@nsa.gov**.

Starting on line 453, the attribute service is described as having three registered attributes called citizenship (lines 461-471), clearance (lines 473-491), and SCI controls (lines 493-513). Citizenship contains one possible value, "USA”. For clearance, the possible values include TS, S, C, and U. For SCI controls, the values CONTROL A, CONTROL B, CONTROL C, CONTROL D, and CONTROL E are allowed. Please note that this attribute service may have more attributes and attribute values available provided that the requestor has the access rights to view them.

The following example shows a snippet of an incorrectly defined attribute authority descriptor. The example is incorrect because the entityID is greater than 255 characters.

Exhibit 3–10: Incorrectly Defined Attribute Authority Descriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
entityID="urn:uniqueidentifier256ormorecharactersdeterminablefromAttrib
uteServicecertificateIssuerDNandSerialNumberssssssssssssssssssssssssssss
"[This is an example of an Entity ID that is too long. Because it
contains more than 255 characters, it will return an error.]>
   <md:AttributeAuthorityDescriptor
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">….
```

3.4.2 Locate Attribute Provider and Attribute Metadata

Within some authentication models, such as those that could be considered pre-retrieval, it is important for consumers of an attribute service to be able to locate the appropriate attribute authority as well as associated metadata. There are special considerations in the case of attribute assertions, as authorization decisions may depend on single or whole sets of attributes. Given the choice of authorization models, it is difficult to predict whether an entity will know how to discover the attributes germane to a particular service. Nor is it feasible to assume that every use case calls for an entity to request all attributes during every exchange, particularly where performance and latency are paramount.

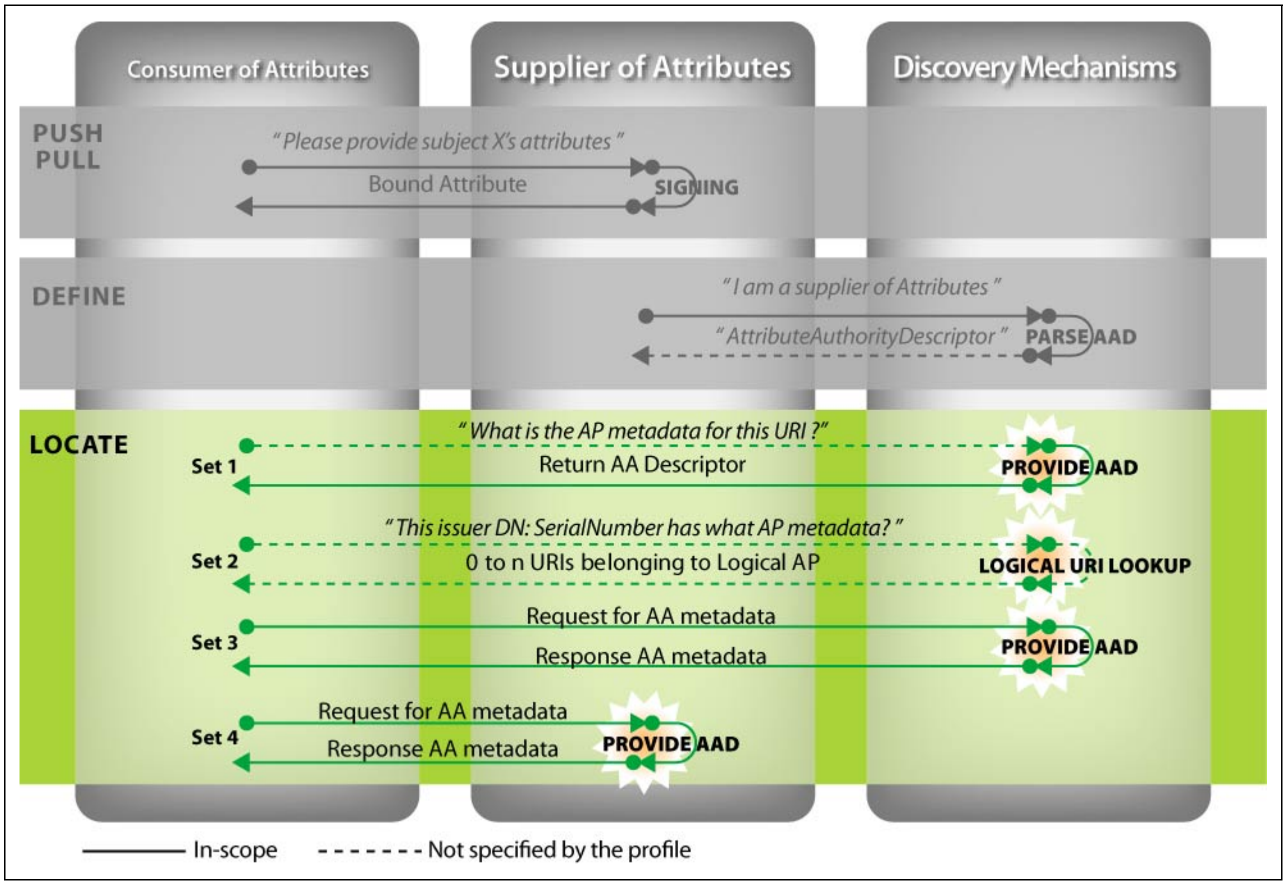


Figure 3–24: Locate Attribute Provider Metadata

From another perspective, different authorization models that are informed by architecture and design constraints also support the need for flexibility. Therefore, this profile applies the SAML metadata query and descriptor as the mechanism to locate the right attribute authority and inquire for metadata about that service.

3.4.2.1 Metadata Query and Response

A consumer of attributes constructs a metadata query to determine the information about the attributes available from a specific attribute authority. This query has a basic structure as shown in **Figure 3–25**.

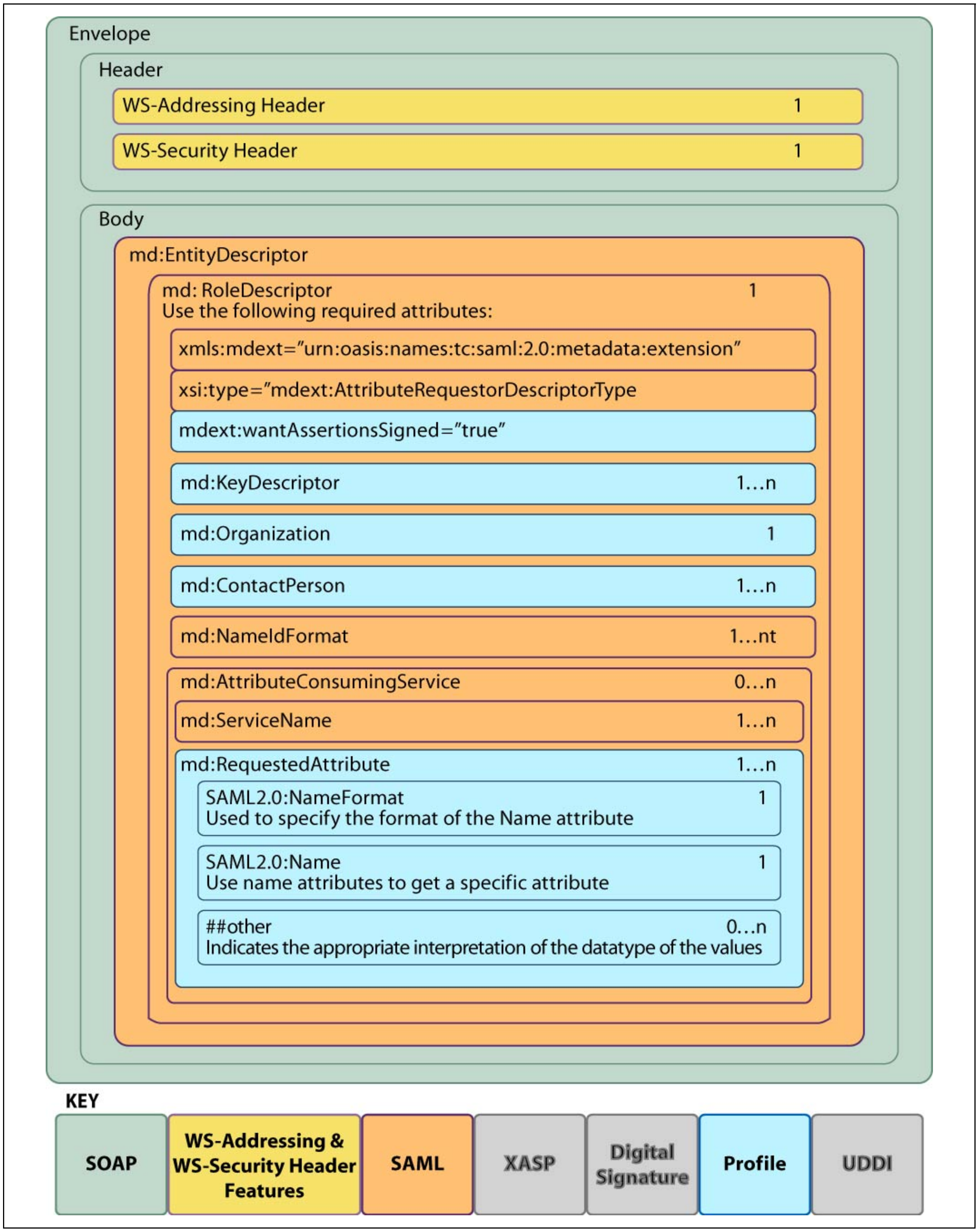

Envelope
Header
WS-Addressing Header 1
WS-Security Header 1
Body
md:EntityDescriptor
md: RoleDescriptor 1
Use the following required attributes:
xmls:mdext="urn:oasis:names:tc:saml:2.0:metadata:extension"
xsi:type="mdext:AttributeRequestorDescriptorType
mdext:wantAssertionsSigned="true"
md:KeyDescriptor 1…n
md:Organization 1
md:ContactPerson 1…n
md:NameIdFormat 1…nt
md:AttributeConsumingService 0…n
md:ServiceName 1…n
md:RequestedAttribute 1…n
SAML2.0:NameFormat 1
Used to specify the format of the Name attribute
SAML2.0:Name 1
Use name attributes to get a specific attribute
##other 0…n
Indicates the appropriate interpretation of the datatype of the values
KEY
SOAP
WS-Addressing & WS-Security Header Features
SAML
XASP
Digital Signature
Profile
UDDI

Figure 3–25: Metadata Query

Within the metadata query, the consumer queries with an entity descriptor which includes a role descriptor of type "AttributeRequesterDescriptorType”. This profile requires that the requestor specify that it wants all assertions to be signed. The following example shows a metadata query from a requestor that supports SAML 1.1 and SAML 2.0:

Exhibit 3–11: SAML Metadata Query

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  cacheDuration="PT30.000S" entityID="AttributeRequesterEntityID">

  <md:RoleDescriptor
  xmlns:mdext="urn:oasis:names:tc:SAML:2.0:metadata:extension"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  mdext:WantAssertionsSigned="true"
  protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol
  urn:oasis:names:tc:SAML:1.1:protocol"
xsi:type="mdext:AttributeRequesterDescriptorType">
  <md:Extensions/>

  <md:KeyDescriptor>
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:X509Data>
        <ds:X509IssuerSerial>
          <ds:X509IssuerName>
          cn=Some Certificate,ou=PKI,ou=DoD,o=U.S. Government,c=US
          </ds:X509IssuerName>
          <ds:X509SerialNumber>42</ds:X509SerialNumber>
        </ds:X509IssuerSerial>
      </ds:X509Data>
    </ds:KeyInfo>
  </md:KeyDescriptor>

  <md:Organization>
      <md:OrganizationName xml:lang="en">
      Goo Service Provider</md:OrganizationName>
      <md:OrganizationDisplayName xml:lang="en">
      Goo Service Provider at Some Location
      </md:OrganizationDisplayName>
      <md:OrganizationURL xml:lang="en">
      htt://www.goo.com
      </md:OrganizationURL>
  </md:Organization>

    <md:ContactPerson contactType="support">
      <md:SurName>Goo Support</md:SurName>
      <md:EmailAddress>goo-support@disa.mil</md:EmailAddress>
    </md:ContactPerson>

  <md:NameIDFormat>
    urn:oasis:names:tc:SAML:1.1:nameid-format:x509SubjectName
  </md:NameIDFormat>

  <md:AttributeConsumingService
    xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" index="0">
```

```
Line
Ref# Code
624         <md:ServiceName xml:lang="en">
625           Goo Consumer Service
626         </md:ServiceName>
627         <md:RequestedAttribute
628             xmlns:xacmlprof=
629                  "urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
630              Name="urn:mil:disa:foo:Citizenship"
631              NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
632           xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"/>
633 </md:AttributeConsumingService>
634
635   </md:RoleDescriptor>
636 </md:EntityDescriptor>
```

The entity descriptor includes a role descriptor of type "AttributeRequesterDescriptorType" which wants all assertions to be signed (line 583) and supports SAML 1.1 and SAML 2.0 (lines 584-585).

The attribute requestor’s X.509 information is given on lines 588-599. Furthermore, a name format of `urn:oasis:names:tc:SAML:1.1:nameid-format:x509SubjectName` is used by this attribute requestor on lines 618-620.

The attribute consuming service is described in depth on lines 622-633. The service is named **"Goo Consumer Service"** and is requesting an attribute called **"Citizenship"** within the `urn:mil:disa:foo` namespace. The attribute consuming service also requires this attribute to be in XML Schema string format for consumption. The service's organization, Goo Service Provider, is described on lines 602-611. On lines 613-616, a contact person called **"Goo Support"** is contactable via email at **goo-support@disa.mil**.

In response to this query, the attribute authority responds with its attribute authority descriptor which was previously described in the DEFINE flow. It is presented here again for continuity purposes. The SAML 2.0 attribute authority descriptor describes basic information about an attribute authority and the attributes that it can provide.

The following is an example of the metadata response:

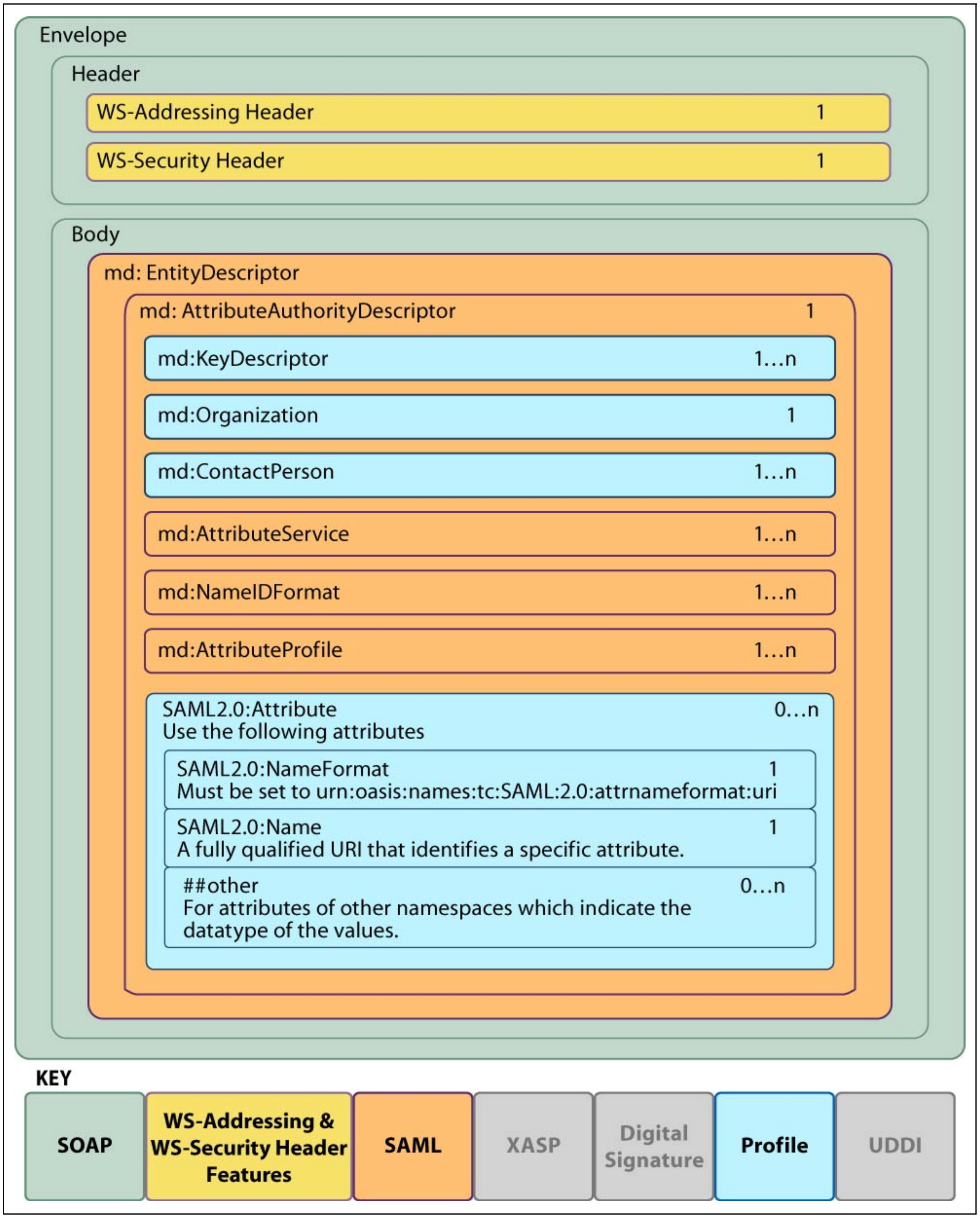


Figure 3–26: Metadata Response

The attribute authority descriptor beginning on line 671 includes the protocols it supports as well as a key descriptor, organization, and contact information. This particular attribute authority descriptor is provided by the "Foo Attribute Service Provider”. In order to get support, "Foo Support" is the contact person reachable via email at foo-support@nsa.gov. The following XML example illustrates a SAML 2.0 attribute authority descriptor:

Exhibit 3–12: SAML 2.0 Attribute Authority Descriptor – Example 2

```
Line
Ref# Code
671 <?xml version="1.0" encoding="UTF-8"?>
672    <md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
673       cacheDuration="PT30.000S"
674       entityID="urn:uniqueidentifier255orlesscharactersdeterminablefromA
675       ttributeServicecertificateIssuerDNandSerialNumber">
676
677    <md:AttributeAuthorityDescriptor
678       protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
679       <md:Extensions/>
680       <md:KeyDescriptor>
681          <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
682             <ds:X509Data>
683                <ds:X509IssuerSerial>
684                   <ds:X509IssuerName>
685                      cn=Some  Certificate,ou=PKI,ou=DoD,o=U.S.
686                      Government,c=US
687                   </ds:X509IssuerName>
688                   <ds:X509SerialNumber>42</ds:X509SerialNumber>
689                </ds:X509IssuerSerial>
690             </ds:X509Data>
691          </ds:KeyInfo>
692       </md:KeyDescriptor>
693
694       <md:Organization>
695          <md:OrganizationName xml:lang="en">
696             Foo Attribute Service Provider
697          </md:OrganizationName>
698          <md:OrganizationDisplayName xml:lang="en">
699             Foo Attribute Service Provider at Some Location
700          </md:OrganizationDisplayName>
701          <md:OrganizationURL xml:lang="en">
702             htt://www.foo.com
703          </md:OrganizationURL>
704       </md:Organization>
705
706       <md:ContactPerson contactType="support">
707          <md:SurName>Foo Support</md:SurName>
708          <md:EmailAddress>foo-support@nsa.gov</md:EmailAddress>
709       </md:ContactPerson>
710
711       <md:AttributeService
712       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
713       Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
714          <md:NameIDFormat>
715             urn:oasis:names:tc:SAML:1.1:nameid-format:x509SubjectName
716          </md:NameIDFormat>
717       <md:AttributeProfile/>
718
719       <saml:Attribute
720          xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
721          xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
722          Name="urn:mil:disa:foo:Citizenship"
```

Line Ref# Code

```
723          NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri""
724         xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string">
725         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
726         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
727         xsi:type="xs:string">USA</saml:AttributeValue>
728         </saml:Attribute>
729
730      <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
731         xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
732         Name="urn:mil:disa:foo:Clearance"
733         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
734         xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string">
735         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
736           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
737           xsi:type="xs:string">TS</saml:AttributeValue>
738         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
739           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
740           xsi:type="xs:string">S</saml:AttributeValue>
741         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
742           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
743           xsi:type="xs:string">C</saml:AttributeValue>
744         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
745           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
746           xsi:type="xs:string">U</saml:AttributeValue>
747        </saml:Attribute>
748
749        <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
750        xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
751        Name="urn:mil:disa:foo:SCIControls"
752        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
753        xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string">
754           <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
755           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
756           xsi:type="xs:string">CONTROL A</saml:AttributeValue>
757           <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
758           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
759           xsi:type="xs:string">CONTROL B</saml:AttributeValue>
760           <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
761           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
762           xsi:type="xs:string">CONTROL C</saml:AttributeValue>
763           <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
764           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
765           xsi:type="xs:string">CONTROL D</saml:AttributeValue>
766           <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
767           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
768           xsi:type="xs:string">CONTROL E</saml:AttributeValue>
769
770        </saml:Attribute>
771
772      </md:AttributeAuthorityDescriptor>
     </md:EntityDescriptor>
```

The entityID on lines 674-675 is the metadata entityID which must be 255 characters or less and derivable from the attribute service certificate's issuer DN and serial number.

Starting on line 711, the attribute service is described as having three registered attributes called citizenship (lines 719-728), clearance (lines 730-747), and SCI controls (lines 749-769). Citizenship contains one possible value "USA”. For clearance, the possible values included in the sample code are TS, S, C, and U. For SCI controls, the values that are articulated by the sample code as being allowed by this particular attribute service are CONTROL A, CONTROL B, CONTROL C, CONTROL D, and CONTROL E. An attribute service may have more attributes and attribute values available provided that the requestor has the access rights to view them. NOTE: the values that appear in the sample are meant to be illustrative, and do not represent the full array of available values. For more information see **Appendix A: Joint Enterprise Directory Services**.

Also note that entities making requests of a discovery service or a naming service for the sake of determining which attributes are available are not making a SAML attribute request. As defined by SAML, these attribute consuming entities may end up finding the answer in the overview document URL or other descriptive tModel (in the case of UDDI). At a minimum, this document should point to some location in which a SAML MD profile-compliant response for the attribute consumer can be obtained. How this mechanism is obtained is out of the scope of this profile (see **Section 3.1.7**).

An attribute service does not need to include information about all supported attributes in its published (available) ***md:AttributeAuthorityDescriptor***.

3.4.3 Mechanism to Map an Attribute Service to an Attribute Authority

When combined with NCES Service Discovery as the Discovery Mechanisms and Public Key Infrastructure (PKI) as the mechanism for message authentication, the use of the SAML metadata profile exposes the need to correlate the unique identification schemes expected by each of these mechanisms:

- The SAML v2.0 Metadata Descriptor Profile requires all entity identifiers to be expressed as a URI.
- NCES Service Discovery relies on UDDI keys.
- PKI relies on Subject names contained within an X.509 certificate.

The SAML entity identifier <*entityid*> is a URI (maximum of 1024 characters in length) representative of a provider of SAML-based services or an active participant in a SAML exchange. It MUST also be used in other elements such as the <*Issuer*>, to identify an issuer of a SAML statement and the <*NameID*>, to make an assertion about an entity making said assertion. For the purposes of traceability and uniqueness across systems this profile encourages coordination between SAML Issuance and the Services Discovery Infrastructure. The most obvious means by which to achieve that symbiotic relationship is by using the same URI for entityid as is used in an uddikey.

In a UDDI registry, to be capably referenced, each business entity, service, template, and tModel must have a unique key. The management of that key in a UDDI V3 system can be performed through the use of a key partition, as long as it is owned by the key publisher. The partition is created from a combination of generated and derived keys that result in a URI construct similar to an entityid. However the length of that key is relegated to a maximum value of 255 characters. Due to the 255 character constraint placed on the uddikey, this profile states that the entityid MUST be no larger than 255 characters.

Each attribute authority must have an X.509 certificate containing its entityid as specified in the applicable service provider certificate profile. The entityid may be represented as a UUID contained in the subject alternative name extension using the URI name type. Since UDDI accepts a UUID as the key specific string (KSS), a colon delimited uddikey may be generated by using the Root Partition Owner (uddi), Domain Name Service (DNS) Domain Name, and supplied KSS. An additional value in using key partitions is that it allows for the transfer of key ownership. Once that ownership is taken on by a new entity, the key can no longer be changed. In this manner uniqueness is maintained throughout the lifecycle of the key binding.

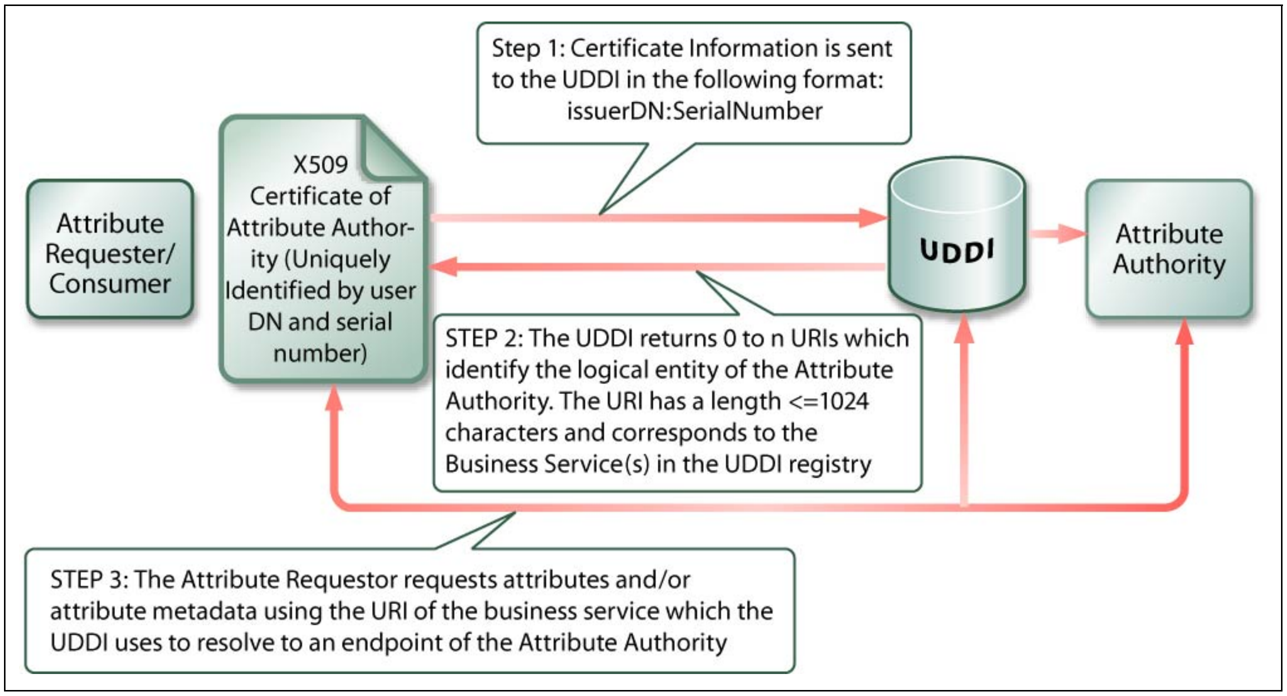

Step 1: Certificate Information is sent to the UDDI in the following format: issuerDN:SerialNumber
Attribute Requester/ Consumer
X509 Certificate of Attribute Authority (Uniquely Identified by user DN and serial number)
UDDI
Attribute Authority
STEP 2: The UDDI returns 0 to n URIs which identify the logical entity of the Attribute Authority. The URI has a length <=1024 characters and corresponds to the Business Service(s) in the UDDI registry
STEP 3: The Attribute Requestor requests attributes and/or attribute metadata using the URI of the business service which the UDDI uses to resolve to an endpoint of the Attribute Authority

Figure 3–27: Extracting Attribute Metadata from Attribute Authority Certificate

The translation approach proposed by this profile is depicted in **Figure 3–27**. The flexibility of this approach ensures that consumers can acquire attribute metadata both from attribute providers directly or utilize directory services, depending on the architectural and infrastructural supports that are available to the authorization model. **Figure 3–28** illustrates the message format for Step 1 of the translation approach and **Figure 3–29** illustrates the metadata response that is returned in Step 3.

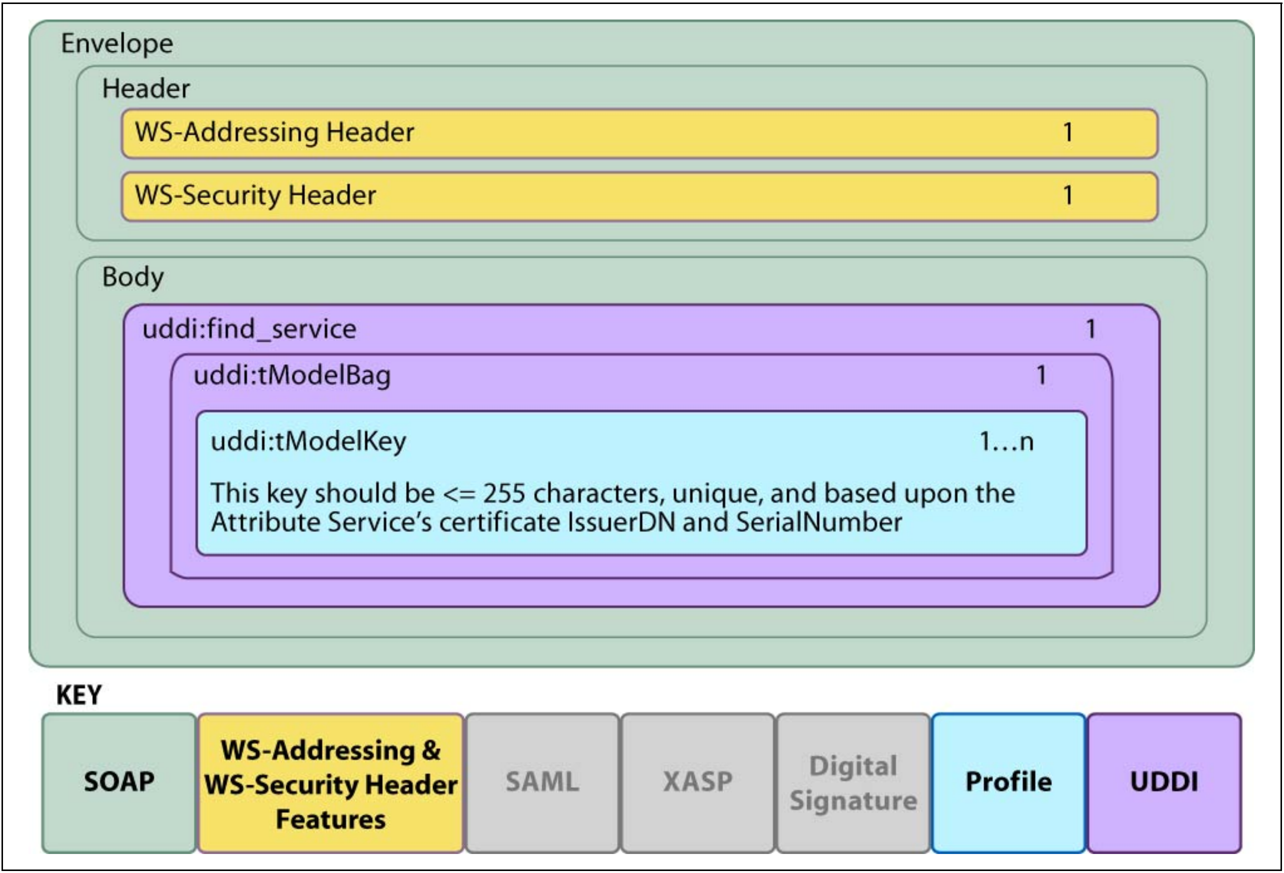

Envelope
Header
WS-Addressing Header 1
WS-Security Header 1
Body
uddi:find_service 1
uddi:tModelBag 1
uddi:tModelKey 1…n
This key should be <= 255 characters, unique, and based upon the Attribute Service's certificate IssuerDN and SerialNumber
KEY
SOAP
WS-Addressing & WS-Security Header Features
SAML
XASP
Digital Signature
Profile
UDDI

Figure 3–28: Illustrative UDDI Request for Service Key

This step includes the creation of the unique identifier which is used through out the metadata and attribute communication flows. The unique uddikey and entityID key is found on lines 855-856 in the following proposed implementation of how to construct the UDDI query:

Exhibit 3–13: Illustrative UDDI Request for Service Key - Details

```
Line
Ref# Code
847 <?xml version="1.0" encoding="UTF-8"?>
848 <SOAP-ENV:Envelope xmlns:SOAP-
849 ENV="http://schemas.xmlsoap.org/soap/envelope/">
850    <SOAP-ENV:Header/>
851    <SOAP-ENV:Body>
852       <uddi:find_service xmlns:uddi="urn:uddi-org:api_v3">
853          <uddi:tModelBag>
854             <uddi:tModelKey>
855                urn:uniqueidentifier255orlesscharactersdeterminablefrom
856                AttributeServicecertificateIssuerDNandSerialNumber
857             </uddi:tModelKey>
858          </uddi:tModelBag>
859       </uddi:find_service>
860    </SOAP-ENV:Body>
861 </SOAP-ENV:Envelope>
```

As previously stated, the UDDI response to this query may be constructed in the manner illustrated in **Figure 3–29**.

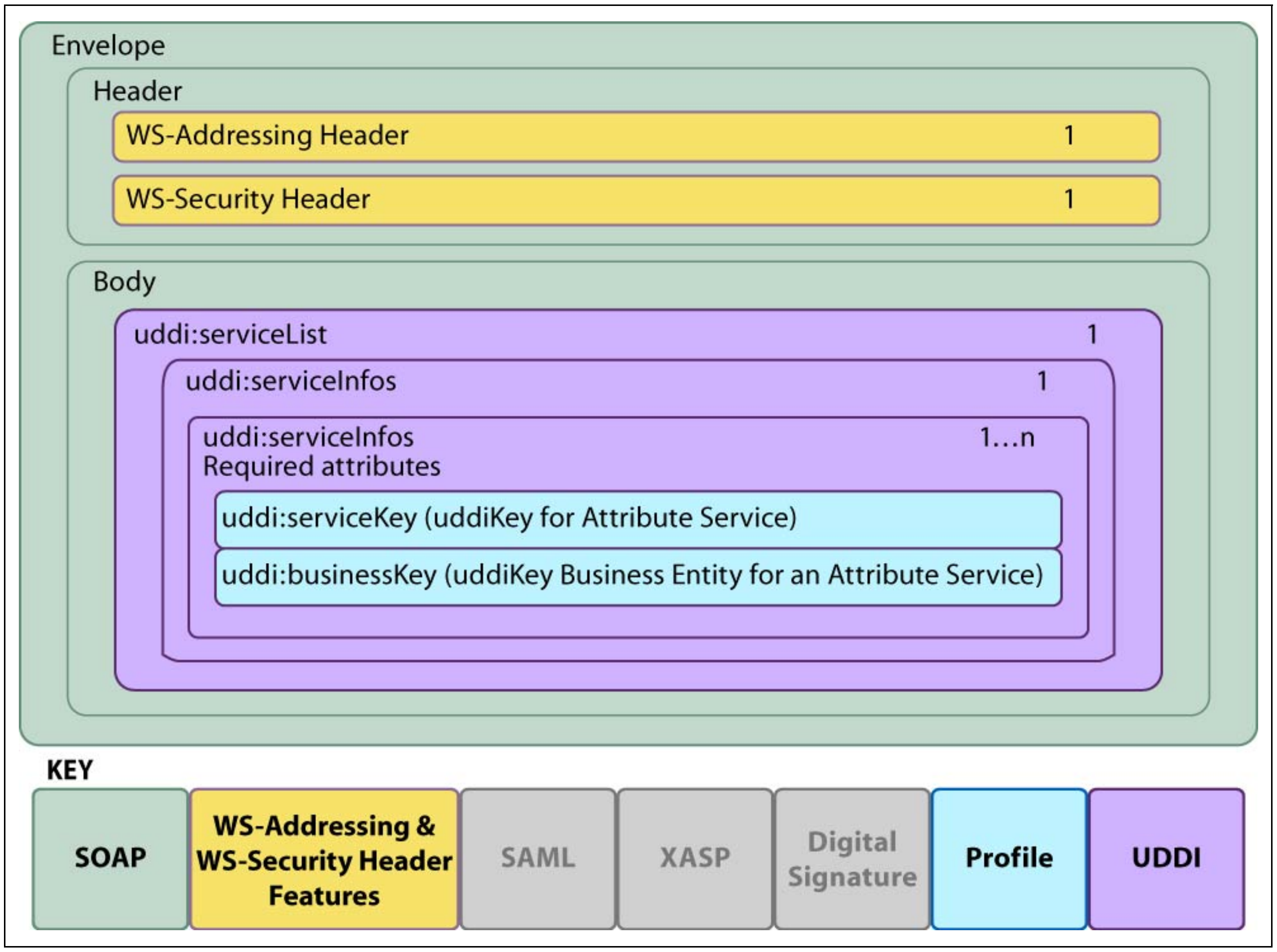


Figure 3–29: Illustrative UDDI Response for Service Key

Of note, the uddikey(s) are the attribute service which will be used to query for attribute metadata and values in the remaining LOCATE communication flows.

3.5 Flush Attribute Cache

As attribute consumers collect attributes over time, they may in some architectures, need to cache those attribute assertions for reuse in support of performance and control of latency. In such instances, rules must be established for the flushing of cached attribute assertions. The issue for assertions that are cached is that the assertions may no longer be valid as time passes. Such rules would look at ***saml:Assertion/@NotBefore*** and ***saml:Assertion/@NotAfter*** to define configuration settings in keeping with the duration that both supports performance and latency requirements, as well as supports the security needs of the architecture. Information in this section relating to flushing of attribute assertion caches is provided for purely informational reasons. This profile currently provides no normative requirements related to flushing of cache.

4 PROFILE REQUIREMENTS

4.1 Fault Codes and Error Conditions

If a conforming implementation does not support the referenced obligation, policy, or attribute set, the ***samlp:StatusCode/@Value*** MUST BE set to ***samlp:Responder*** in accordance with the SAML v1.1 Standard.

A conforming implementation SHOULD NOT provide a second level <*StatusCode*> to any consumer that makes a direct request to an attribute service of unknown origin or that includes an invalid signature in the request. By limiting this information, this reduces an attacker's ability to acquire more explicit information from error messages.

4.2 Attribute Requirements

Transformation of **saml2** attribute information MUST conform to the requirements defined in [SAML-XACML]. These rules include:

- **xacml:AttributeDesignator/@AttributeId** MUST contain the fully-qualified value of the **saml:Attribute/@Name**.
- **xacml:AttributeDesignator/@DataType** MUST contain the fully-qualified value of the **saml:Attribute/@DataType**.
- **xacml:AttributeDesignator/@Issuer** MUST contain the string value of the **saml:Issuer** the **saml:Assertion**.
- **xacmlc:AttributeValue** MUST contain the value from **saml:Attribute/AttributeValue**.

For additional guidance, please refer to the normative section of [SAML-XACML] which takes precedence over this non-normative treatment.

In addition, when a SAML2.0 attribute construct is used, additional data type information should be included to comply with additional SAML profiles such as [X500] and [XACML].

4.2.1 NameFormat

The *NameFormat* XML attribute in <*Attribute*> elements MUST be `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

The ***saml:Attribute/@Name*** MUST adhere to the rules specified for this format, as defined by [SAML2Core]. The optional XML attribute ***saml:Attribute/@FriendlyName*** (defined in [SAML2Core]) MAY be used to carry an optional string name together with the OID Uniform Resource Name (URN) for human readability, but MUST NOT contain the XACML attribute identifier.

4.2.2 Attribute Name Comparison

Two <*Attribute*> elements refer to the same SAML attribute if and only if the values of ***saml:Attribute/@Name*** are equal in a binary comparison. The ***saml:Attribute/@FriendlyName*** attribute MUST NOT play a role in the comparison.

4.2.3 Data Type Expression

Each attribute MUST carry explicit data type information in the XML attribute ***saml:Attribute/@DataType*** which is defined in the XML namespace. The value for this attribute MUST be a URI. While in principle any URI reference can be used as a data type, the standard values to be used are specified in **Appendix A** of the **XACML 2.0 Specification [XACML]**. 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.

4.2.4 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 ***saml:Attribute/@DataType***. For data types corresponding to the types defined in **Section 3.3** of [Schema2], the xsi:type XML attribute SHOULD also be used on the ***saml:Attribute*** element(s).

4.3 Attribute Assertion Requirements

A SAML attribute assertion is a ***saml:Assertion*** instance that contains one ***saml:AttributeStatement*** instance. Each attribute statement may contain one or more ***saml:Attribute*** instances. In order to be used in an XACML request context, each SAML attribute in the SAML attribute assertion SHALL comply with *XACML Attribute Profile* (**Section 4.7**), whose namespace is `urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML`. This is contained within the *Profiles for the OASIS Security Assertion Markup Language* [SAML2Prof]. An **xacml-context:Attribute** SHALL be constructed from the corresponding saml:Attribute.

4.3.1 Issuer

Any assertion issued by the attribute services MUST contain a ***saml:Issuer*** element with the same unique identifier as contained in the ***md:EntityDescriptor/@entityID*** for that attribute service.

4.3.2 Attribute Statements

An attribute statement MUST contain exactly one ***samlp:AttributeStatement*** element that reflects the attributes of the subject that the attribute service will expose to the service consumer.

4.3.2.1 Subject Element

An attribute statement MUST contain exactly one Subject element. See page 18 of [SAML2Core].

4.3.3 Digital Signature

All SAML assertions MUST be signed using XML Signature. See page 68 in [SAML2Core]. The ***saml:AttributeAssertion*** MUST be signed by the attribute service. The **saml:AttributeStatement** within the ***saml:AttributeAssertion*** MUST be covered by the signature. The attribute consumer MUST ensure that the signature is valid and that the ***saml:Assertion/Issuer*** is consistent with any ***ds:X509IssuerName*** value in the signature. The syntax and processing of digital signatures SHALL adhere to the guidelines regarding digital signatures in Section 5, *SAML and XML Signature Syntax and Processing* of the SAML core specification [SAML]. Additionally, the SAML assertion message should be signed following the WSSE profile (check **Figure 1–4**).

4.3.4 Conditions

The ***saml:AttributeAssertion*** MUST contain ***saml:AttributeAssertion/@NotBefore*** and ***saml:AttributeAssertion/@NotOnOrAfter*** which specify the validity period for the attributes contained within this assertion. When translating attribute information into an XACML context, the ***saml:AttributeAssertion/@NotBefore*** and ***saml:AttributeAssertion/@NotOnOrAfter*** values SHALL be consistent with the &environment;current-time, &environment;current-date, and &environment:current-dateTime <*xacml:Attribute*> values associated with the <*xacml:Request*>. The ***saml:AttributeAssertion*** MUST contain AudienceRestrictionCondition containing an audience element. The audience element consists of a URI identifying each of the parties relying on the assertion. All of these conditions help to defeat replay attacks.

4.4 Attribute Query Requirements

In general, an attribute service MUST process the <AttributeQuery> message and any enclosed <Attribute> elements as described in [SAMLCore] and in Section 6 of [SAMLProf].

If the ***saml2:Attribute*** construct is used, ***xacml:DataType*** SHOULD also be included.

4.4.1 Attribute Query

The ***samlp:AttributeQuery*** element MUST conform to the following rules:

The ***samlp:AttributeQuery/Subject*** element MUST contain a ***samlp:NameID*** element with the value of the *Subject DN* from the principal's X.509v3 certificate and MUST have a format with the value of `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`.

If multiple namespaces and/or expressions are contained in an attribute request, each query expression MUST be inserted in a separate ***saml2:Attribute.***

4.5 SAML Response Requirements

If the attribute service wishes to return an error, it MUST NOT include any assertions in the ***samlp:Response*** message. Otherwise, if the request is successful, the ***samlp:Response*** element MUST conform to the following rules:

- It MUST contain exactly one ***saml:Assertion*** element.
- The ***saml:Assertion*** element MUST satisfy the following conditions:
    - Condition elements MAY be included as requested by the service consumer or at the discretion of the attribute service.
    - The structure of the assertions provided in response to an attribute query is identical to the structure of an attribute assertion that is pushed.

Although the ***samlp:AudienceRestriction*** is a required element in ***samlp:Response;*** it is included for profile compliance only and SHOULD NOT bear weight upon the assertion decision or response.

4.5.1 InResponseTo

While [SAML2Prof] defines the *InResponseTo* element as optional; the potential threat of message injections suggests this attribute MUST be REQUIRED. The values of the ID attribute in a request and the *InResponseTo* attribute in the corresponding response MUST match.

4.6 Attribute Service Metadata Requirements

4.6.1 EntityID

The unique identifier in the ***md:EntityDescriptor/@entityID*** for a conforming attribute service implementation MUST contain the same value that will be inserted into any assertion issued by the attribute services as the value for the ***samlp:Issuer*** element.

The entityid MUST be no larger than 255 characters.

The value of the entityid URI MUST be contained in the attribute service provider's X.509 certificate.

4.6.2 X.509 URIs

As attribute requests may be accompanied by the authentication assertions, the use of base URIs as defined by [SAMLAC-X509] to validate the use of X.509 certification authentication method SHOULD be supported.

The URI defined for Public Key X.509 is defined in [SAMLAC-X509].as `urn:oasis:names:tc:SAML:2.0:ac:classes:X509`. Note that this URI is also used as the target namespace in the corresponding authentication context class schema document [SAMLAC-X509].

4.6.3 md:EntityDescriptor/@WantAssertionsSigned

This attribute indicates a requirement for assertions received by this requester to be signed. This profile REQUIRES the presence of this attribute. Moreover, it is RECOMMENDED that the value for this attribute be true. This requirement is in addition to any requirement for signing derived from the use of a particular profile/binding combination.

4.7 XACML Assertion

4.7.1 Vocabulary

The vocabularies must be supported within both the **capabilities** and **requirements** contexts: `urn:oasis:names:tc:xacml:3.0:vocabulary:xacml`.

Within an attribute authorization model, when using a requirement or capability element within a XACML Assertion, the Vocabulary element MUST be supported within both the capabilities and requirements contexts of the attribute authorization and usage model. NOTE: This requirement is included within this standard as a future compatibility requirement based on WS-XACML. At this time, WS-XACML is a draft standard, and inclusion of it supports use cases whereby service consumers requirements are mapped as XACML attributes to service providers capabilities or vice versa.

4.8 URI Normalization

SAML system entities MUST employ the URI normalized form. Specifically:

- SAML system entities MUST encode all resource URI references in normalized form.
- Relying parties MUST convert resource URI references to normalized form prior to processing. See IETF RFC 2396 Section 6 for URI normalization rules.
- A requestor MUST NOT be able to gain access to a denied resource by changing the case of a part of the resource's URI resource.
- A requestor MUST NOT be able to gain access to a denied resource by creating symbolic links or logical paths.

See page 32 in [SAML2Core] for more information about URI normalization with SAML.

4.9 SAML Protocol Version and SAML Assertion Version

- A SAML requestor MUST issue requests with the highest request version supported by both the SAML requestor and the SAML responder.
- If the SAML requestor does not know the capabilities of the SAML responder, then it MUST assume that the responder supports requests with the highest version supported by the requestor.

- A SAML responder MUST reject requests whose minor request version number is higher than the highest supported request version it supports.
- A SAML relying party MUST reject an assertion with whose minor assertion version number is higher than the minor assertion version number supported by the relying party.

See page 66 in [SAML2Core] for more information.

4.10 Request/Response Signing

- All SAML protocol request and response messages MUST be signed using XML Signature. See page 68 in [SAML2Core].

5 CONFORMANCE

Although SAML 2.0 is incompatible with its predecessor, SAML 1.0, implementations of this profile MUST support the normative portions of the [SAML2] and [SAML1] base standard. In addition, service consumers and providers MUST be authenticated and have their integrity protected by digital signatures using appropriate public key technology (e.g., PKI). For an implementation to conform to this profile, it MUST adhere to all mandatory aspects of the profile.

6 SECURITY CONSIDERATIONS

Security considerations of this profile will be explored in the near future, which may cover a variety of security vulnerability and analysis areas of considerations:

- Subject Authentication and Trust
- Service Chaining
- Digital Signatures
- Authoritative Sources & Attribute Provisioning Authorities
- Caching
- Confidentiality
- Integrity
- Monitoring
- Security Management
- Data Labeling

7 OTHER CONSIDERATIONS

7.1 Relationship to WS-I Basic Security Profile

The WS-I Basic profile was developed to address the need for a series of non-proprietary, commonly accepted application semantics that allow for non-intrusive but testable compatibility and interoperability amongst heterogeneous web services entities. The WS-I Basic Security profile addresses not only interoperability issues but the need to implement security best practices without disturbing that pre-established baseline conformance. The SAML Attribute Sharing profile considers the WS-I Basic Security Profile, in particular the WS-I SAML Token Profile. The inclusion of the WS-I profiles are such that all provisions have been made to deconflict any potential collisions with the guidance of any non-WS-I profile and this document.

7.2 Relationship to XCCDF

The eXtensible Configuration Checklist Description Format is a specification language designed to support a uniform data-model for automated compliance testing and scoring as expressed through security checklists, benchmarks, and other configuration guidance. The vision is that the format can be used as a means of ensuring compliance against multiple policies across varied community groups and vendors. However, this document format is not a feasible model by which to ensure application-based profile compliance.

7.3 Relationship to DoDD 8100

DoD Directives 8100.1 and 8100.2 establishes policy, assigns responsibilities, promotes interoperability for GIG configuration management, architecture, and the relationships with the Intelligence Community (IC) and defense intelligence components. This profile addresses the needs of both directives by taking into consideration the security and interoperability needs of both the IC and NCES. Although not explicitly expressed through DoDD 8100, requirements to meet its demands are defined within various supportive NCES and DNI documentation. An exhaustive effort has been made to analyze all known documentation relevant to scope throughout the development of this document. Additionally, since implementation of these directives will be realized predominantly through the use of COTS products, the profile performs a sanity check by considering private industry and current market trends as well.

8 WAY AHEAD

In future versions, this profile may expand on the requirements and mechanisms of performing binding of attributes through UDDI, as well as the usage of assertion ID-based retrieval of assertions, and the use of attribute artifact through REST bindings.

While the publishing and discovery of related SAML 2.0 Metadata document URLs can be facilitated through the use of Dynamic Delegation Discovery System (DDDS) mechanisms such as querying NAPTR resource records, those mechanisms are not being considered to date within the NCES architecture; they are possibly reserved for future considerations.

Also in the future is the notion that an attribute service can generate tModels based on what attribute structure it has and produce and/or immediately publish that attribute tModel.

New XACML attribute identifiers may be defined in the future, as well as additional SAML attributes, including:

- Maximum data retention days
- Maximum data retention hours
- Maximum data retention minutes

9 REFERENCES

This section identifies external documents that are referenced by this profile.

9.1 Normative

[RFC2396]	T. Berners-Lee et al. Uniform Resource Identifiers (URI): Generic Syntax. See http://www.ietf.org/rfc/rfc2396.txt. NOTE: This RFC which is referenced in XML Schema is made obsolete by [RFC 3986].
[RFC 3986]	T. Berners-Lee et al. Uniform Resource Identifier (URI): Generic Syntax. See http://www.ietf.org/rfc/rfc3986.txt.
[META-EXT]	SAML Metadata Extension for a Standalone Attribute Requester, draft-saml-metadata-ext-01, 11 March 2005.
[META-EXT-QUERY]	Metadata Extension for SAML V2.0 and V1.x Query Requesters, sst-saml-metadata-ext-query-cd-02, 1 September 2006.
[SAMLAC-X509]	J. Kemp et al. SAML context class schema for Public Key – X.509. OASIS SSTC, March 2005. Document ID saml-schema-authn-context-x509-2.0. See http://www.oasis-open.org/committees/security/.
[SAML1Core]	E. Maler et al. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1 OASIS Standard, 2 September 2003. See http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf.
[SAML2Core]	S. Cantor et al. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS SSTC, March 2005. Document ID saml-core-2.0-os. See http://www.oasis-open.org/committees/security/.
[SAML-MD]	S. Cantor et al. Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, March 2005. Document ID saml-metadata-2.0-os. http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf.
[SAML2Prof]	S. Cantor et al. Profiles for the OASIS Security Assertion Markup Language(SAML) V2.0. OASIS Standard, March 2005. Document ID saml-profiles-2.0-os. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf.
[SAMLX500-xsd]	S. Cantor et al. SAML X.500/LDAP attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-x500-2.0. http://www.oasisopen.org/committees/security/.
[SAML-XACML]	A. Anderson, H Lockhart et al. SAML 2.0 Profile of XACML 2.0 Assertion Extension OASIS Standard, 1 February 2005. access_control-xacml-2.0-saml-profile-spec-os
[Schema2]	P. V. Biron et al. XML Schema Part 2: Datatypes. World Wide Web Consortium Recommendation, May 2001. See http://www.w3.org/TR/xmlschema-2/.

[UDDIv3]	L Clement et al. UDDI Version 3.0.2 UDDI Spec Technical Committee Draft, Dated 20041019. See http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3.
[WS-XACML]	H Lockhart et al. Web Services Profile of XACML (WS-XACML) Version 1.0. Working Draft 8, 12 December 2006. xacml-3.0-profile-webservices-spec-v1.0-wd-8.
[WSS: SAML Token Profile]	P. Hallem-Baker et al., Web Services Security: SAML Token Profile 1.0, December 2004. See http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf.
[WSS: SOAP Message Security V1.0]	A Nadelin et al. Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) See http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.
[x500]	S. Cantor et al. SAML V2.0 X.500/LDAP Attribute Profile, sstc-saml-attribute-x500-cd-01, December 2006.
[XACML]	eXtensible Access Control Markup Language (XACML) Version 2.0 OASIS Standard, 1 Feb 2005. See http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.
[XASP]	SAML Attribute Sharing Profile for X.509 Authentication-Based Systems, sst-saml-x509-authn-attrib-profile-cd-02, 28 March 2006.
[XML]	Extensible Markup Language (XML) 1.0 (Fourth Edition), W3C Recommendation 16 August 2006, edited in place 29 September 2006, See http://www.w3.org/TR/xml.
[XMLSig]	D. Eastlake et al. XML-Signature Syntax and Processing. World Wide Web Consortium, February 2002. See http://www.w3.org/TR/xmldsig-core/. Note that this specification normatively references [XMLSig-XSD], listed below.
[XMLSig-XSD]	XML Signature Schema. World Wide Web Consortium. See http://www.w3.org/.
[XMLSchema]	XML Schema, 12 September 2005, http://www.w3.org/2001/XMLSchema.

9.2 Informative

[ODNI-SOASRA]	Office of the Director of National Intelligence Chief Information Officer, Intelligence Community Enterprise Architecture Service-Oriented Architecture Security Reference Architecture, v1.1, 1 December 2006.
[NCESvuln]	*Global Information Grid (GIG) Network Centric Enterprise Services (NCES) Preliminary Vulnerability Assessment Report*, Draft Version 0.8, 6 October 2004.
[WS-vuln]	*Web Services Security Vulnerability Assessment*, Report # I333-020R-2004, 15 September 2004.
[NCESarch]	*Net-Centric Enterprise Services (NCES) Security Core Enterprise Services (CES) Architecture*, Version 0.5 (Pilot), 29 December 2004.

[NCESspec]	*Net-Centric Enterprise Services (NCES) Service Security Design Specification*, nces-soaf-ss-designspec-2005v01-WD-01, 23 May 2005.
[NCESintf]	*Net-Centric Enterprise Services (NCES) Service Security Interface Specification*, nces-soaf-ss-interfacespec-2005v01-WD-01, 23 May 2005.
[SAMLv1.1]	A collection of documents that make up Version 1.1 of the SAML profile (This is the SAML TC page for OASIS that includes links for all of the SAML documents.) SAML 1.1 should have the #name reference within the URL.
[SAMLv2.0]	A collection of documents that make up Version 1.1 of the SAML profile (This is the SAML TC page for OASIS that includes links for all of the SAML documents.) SAML 2.0 should have the #name reference within the URL.
[SAML-Gloss]	Glossary for the OASIS Security Assertion Markup Language (SAML)V2.0, saml-glossary-2.0-os, 15 March 2005.
[SAMLProfX509]	T. Scavo et al. SAML V1.1 Profiles for X.509 Subjects OASIS Draft, 29 August 2006.
[WS-SecurityPolicy]	S. Anderson, et al. Web Services Security Policy Language (WS-SecurityPolicy). Version 1.1. February 2005. http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf.
[WS-Trust]	S. Anderson, et al. Web Services Trust Language (WS-Trust). February 2005. http://specs.xmlsoap.org/ws/2005/02/trust/WS-Trust.pdf.
[WSSE]	*Net-Centric Enterprise Services (NCES) Profile of Web Service Security: SOAP Message Security (WSSE)*, NSA-IAD Serial: 121907WSSE, 12 December 2007.
[XACML]	*Net-Centric Enterprise Services (NCES) Profile of XACML*, NSA-IAD Serial: 110807XACML, 11 November 2007.

APPENDIX A: JOINT ENTERPRISE DIRECTORY SERVICES

The Joint Enterprise Directory Services (JEDS) is an attribute aggregation service, supported under the Identity Management Division (IA4) of the Defense Information Systems Agency (DISA) for NCES. The service's mission is to collect and provision all relevant directory attributes from across the DoD and the IC, which provides a centralized attribute retrieval service and authority store. To support this effort, a series of base attributes have been identified in the DoD and IC communities. JEDS datastore is a centralized reference to the authorized sources for those base attributes. Although there may be additional attributes in the local directory store of origin, those are not required for a basic provisioning decision. This profile recognizes the existence of this service as the authoritative white pages of attributes across DoD and IC for SAML-based decisions and assertions. The profile makes no assumptions about the JEDS architecture, implementation, or security posture.

In the future, working in conjunction with DISA - IA4, NSA-IAD may undertake a more complete analysis of the use of the base attributes within the SAML attribute profile.

APPENDIX B: GLOSSARY OF TERMS

Term	Definition
Asserting Party	A mission entity that issues assertions. [NCESarch]
Attribute Assertion	A SAML assertion that contains security context information related to the attributes for an entity. The Attribute Assertion contains a SAML attribute statement that indicates details such as the issuer of the assertion (the attribute authority), the date/time of assertion, the subject of the assertion (corresponding to the subject from the request), and the attributes associated with the subject. The Attribute Assertion is the response to an Attribute Assertion Request. [NCESarch]
Attribute Based Access Control (ABAC)	A policy model that allows for access control policy applicability; and the associated rules that govern access, to be formulated based on an extensible notion of subject, resource, and other attributes. ABAC encompasses the following three tenets: • An extensible notion of subject attributes encompassing identifiers, groups, roles, and any number of additional subject attribute types. • The use of attributes in policy rules where attributes are compared with fixed values or with each other, in accordance with the appropriate security business logic(s). • The use of resource attributes when specifying the applicability of a policy. [ODNI-SOASRA], [NCESarch]
Attribute Consumer	An Attribute Consumer is a special category of SAML Requesters that utilize the SAML protocol to request attributes about a subject on behalf of themselves as relying parties, on behalf of themselves as subjects, or on behalf of other entities acting as relying parties.
Attribute Information Service (AIS)	A service that provides attribute values that describe subjects (either human users or systems) for the purpose of enabling all Intelligence Community (IC) organizations and partners to make access control decisions related to their data. [ODNI-SOASRA]
Mission Entity	An entity that is operationally responsible for performing a mission function, or operationally in need of having a mission function performed. Mission entities may include entities such as individuals (humans), organizational units, Community of Interest (COIs), and programs of record. [NCESarch]
Principal	A Principal is a system entity that has an identity, that may be authenticated, that is capable of making decisions, and to whom actions performed within the enterprise may be attributed. A Principal may refer to human entities such as an individual user, an organization, or a legal entity; depending on the context, it may also refer to non-human system entities such as a Web Service provider. NOTE: This document makes a distinction between Principals and Identities. A Principal may have multiple local identities in different security domains. For example, a user Principal can have a work account called "JDoe" in his employer's network and also a personal account called "John_Doe" issued by his Internet Service Provider (ISP). [NCESarch]
Service Provider	A mission entity that performs a mission function for another mission entity. NOTE: A service provider may be a consumer of other service providers. [NCESarch]

APPENDIX C: ACRONYMS LIST

Acronym	Description
ABAC	Attribute Based Access Control
AP	Attribute Provider
AS	Attribute Service
CES	Core Enterprise Service
CVS	Certificate Validation Service
DDDS	Dynamic Delegation Discovery System
DISA	Defense Information Systems Agency
DN	Domain Name
DNS	Domain Name Service
HTTP	Hypertext Transfer Protocol
IA	Information Assurance
IAD	Information Assurance Directorate
IC	Intelligence Community
ID	Identification Data
IETF	Internet Engineering Task Force
ISP	Internet Service Provider
JEDS	Joint Enterprise Directory Services
KSS	Key Specific String
NAPTR	Naming Authority Pointer
NCES	Net-Centric Enterprise Services
NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
ODNI	Office of the Director of National Intelligence
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PKI	Public Key Infrastructure
RBAC	Role-Based Access Control
REGEX	Regular Expression
REST	Representational State Transfer (alternative to SOAP)
RFC	Request for Comment
SAML	Security Assertion Markup Language
SC	Service Consumer
SOA	Service-Oriented Architecture
SP	Service Provider
S-Profiles	SOAP Profiles
UDDI	Universal Description Discovery and Integration
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UUID	Universally Unique Identifier
W3C	World Wide Web Consortium
WS	Web Services
WSSE	Web Service Security: SOAP Message Security
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language
XPath	XML Path Language