SAML V2.0 Implementation Profile for Federation Interoperability

Version	1.0
Date	2018-04-18
Location	https://docs.kantarainitiative.org/fi/rec-saml2-implementation-profile-for-fedinterop.html
Status	This document is a Kantara Initiative Recommendation produced by the Federation Interoperability Work Group. It has been approved by the Membership of the Kantara Initiative. See the Kantara Initiative Operating Procedures for more information.

Contributors

- Walter Hoehn (The University of Memphis) - *Editor*
- Scott Cantor (The Ohio State University)
- Rainer Hörbe (Identinetics GmbH, Kantara Initiative)
- Tom Scavo (InCommon)
- Eric Goodman (University of California, Office of the President)
- Brett Bieber (University of Nebraska-Lincoln)
- Nick Roy (InCommon)
- Barry Ribbeck (Rice University)
- Judith E. Bush (OCLC)
- Mike Grady (Unicon)

Abstract

This document encompasses a set of software conformance requirements intended to facilitate interoperability within the context of full mesh identity federations, such as those found in the research and education sector. It attempts to address a number of common barriers to interoperability and details features that are necessary in order to use SAML metadata as a foundation for scalable trust fabrics. It supercedes the eGovernment Implementation Profile V2.0bis from June 2011.

Table of Contents

1. Introduction

The material contained herein is intended for use by implementers of SAML software. It does not specify a fixed set of behaviors for all deployments or limit in any way the features that can be provided in a given implementation, but rather serves as a complement to deployment profiles by identifying a standard set of software capabilities necessary for scalable federation. This profile can form the basis for testing frameworks capable of providing for the validation of conformant software packages.

1.1. Notation

This specification uses normative text to describe the implementation of SAML software capabilities. 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 [RFC2119].

Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces as follows, whether or not a namespace declaration is present in the example:

- The prefix saml: stands for the SAML 2.0 assertion namespace, urn:oasis:names:tc:SAML:2.0:assertion
- The prefix samlp: stands for the SAML 2.0 protocol namespace, urn:oasis:names:tc:SAML:2.0:protocol
- The prefix md: stands for the SAML 2.0 metadata namespace, urn:oasis:names:tc:SAML:2.0:metadata
- The prefix mdattr: stands for the Metadata Extension for Entity Attributes Version 1.0 namespace, [MetaAttr] urn:oasis:names:tc:SAML:metadata:attribute

This specification uses the following typographical conventions in text: `<ns:Element>`, `Attribute`, **Datatype**, `OtherCode`. The normative requirements of this specification are individually labeled with a unique identifier in the following form: **[IIP-EXAMPLE01]**. All information within these requirements should be considered normative unless it is set in *italic* type. Italicized text is non-normative and is intended to provide additional information that may be helpful in implementing the normative requirements.

2. Common Requirements

2.1. General

[IIP-G01]

Implementations MUST allow for reasonable clock skew between systems when interpreting **xsd:dateTime** values and enforcing security policies based thereupon.

The following is a non-exhaustive list of items to which this directive applies: `NotBefore`, `NotOnOrAfter`, *and* `validUntil` *XML attributes found on* `Conditions`, `SubjectConfirmationData`, `LogoutRequest`, `EntityDescriptor`, `EntitiesDescriptor`, `RoleDescriptor`, *and* `AffiliationDescriptor` *elements.*

Tolerances of 3–5 minutes are a reasonable default, but allowing for configurability is a suggested practice.

Guidance is offered with regard to clock skew in E92 of [SAML2Errata]. Empirical evidence has shown that allowance for clock skew is critical to robust SAML deployments. It has thus been made a mandatory feature in this profile.

[IIP-G02]

When specific constraints are absent in the SAML standards or profile documents, implementations MUST be able to accept, without error or truncation, element and attribute values of type **xs:string** that are comprised of any combination of valid XML characters and contain up to 256 characters. This requirement applies both to types defined within the SAML standards (such as transient and persistent NameIDs) and to user-defined types.

It is acceptable to allow for selective configuration of more restrictive constraints on specific NameID and AttributeValue types.

[IIP-G03]

Implementations MUST not send and MUST have the ability to reject SAML protocol messages containing a DTD (Document Type Definition).

2.2. Metadata and Trust Management

This section identifies a basic set of features required for the interoperable use of SAML metadata. Although metadata support was optional in the original SAML V2.0 specification, it is now recognized as a critical component of all modern SAML software. Implementations must carefully adhere to the following procedures related to the exchange and validation of metadata in order to support a scalable federation model.

The underlying intent of these requirements is to facilitate a configuration and provisioning model that is, to the greatest extent practical, based on metadata alone.

The requirements are divided into to general categories: exchange and usage. Unless otherwise specified, requirements around the verification and use of metadata apply regardless of what means of exchange is used in a particular case.

2.2.1. Metadata Exchange

[IIP-MD01]

Identity Providers MUST and Service Providers SHOULD support the acquisition of SAML metadata rooted in `<md:EntityDescriptor>` elements via the Metadata Query Protocol, defined in [SAML-MDQ] and [MDQ]. Implementations that claim support for this protocol MUST be able to request and utilize metadata from one or more MDQ responders for any peer entity from which a SAML protocol message is received.

[IIP-MD02]

Implementations MUST support the routine consumption of SAML metadata from a remote location via HTTP/1.1 [RFC2616] on a scheduled/recurring basis, with the contents automatically applied upon successful validation. HTTP/1.1 redirects (status codes 301, 302, and 307) MUST be honored. Implementations MUST support the consumption of SAML metadata rooted in both `<md:EntityDescriptor>` and `<md:EntitiesDescriptor>` elements by this mechanism. In the latter case, any number of child elements MUST be allowed.

This approach is less flexible than the on-demand model supported by [MDQ] and has scalability issues with larger metadata aggregates. It is required, however, for compatibility with federations that have not begun offering newer approaches and addresses common Service Provider needs around the IdP discovery process. This requirement will be reevaluated in a future revision of this profile.

[IIP-MD03]

Implementations MUST be able to validate the authenticity and integrity of SAML metadata by verifying an enveloped XML Signature [XMLSig] attached to the root element of the metadata. Public keys used for signature verification of the metadata MUST be configured out of band. These keys MAY be contained in X.509 certificates but it MUST be possible to ignore the other contents of the certificate and verify the XML Signature based solely on the public key. It MUST be possible to limit the use of a trusted key to a single metadata source.

[IIP-MD04]

Implementations MUST be capable of rejecting SAML metadata if any one of the following conditions is true:

1. The `validUntil` XML attribute on the root element is missing
2. The value of the `validUntil` XML attribute on the root element is a **xsd:dateTime** in the past
3. The value of the `validUntil` XML attribute on the root element is a **xsd:dateTime** too far into the future, where "too far into the future" is a configurable option.

Note that this requirement applies to the root element only. If the root element encloses any child elements containing additional `validUntil` *XML attributes, these must be processed in accordance with [SAML2Meta].*

2.2.2. Metadata Usage

[IIP-MD05]

Implementations MUST support SAML Metadata as defined in the following OASIS specifications:

- SAML V2.0 Metadata [SAML2Meta], as updated by Errata [SAML2Errata]
- SAML V2.0 Metadata Schema [SAML2MD-xsd]
- SAML V2.0 Metadata Interoperability Profile [SAML2MDIOP]
- SAML V2.0 Metadata Extension for Entity Attributes [MetaAttr]
- SAML V2.0 Metadata Extension for Algorithm Support [SAML2MetaAlgSup]
- SAML V2.0 Metadata Extensions for Login and Discovery User Interface [MetaUi]

The above list of specifications includes all material relevant to functionality required by this profile, but is not intended to be exhaustive. In accordance with the Extensibility section, other metadata extension content may be present and MUST NOT prevent consumption and use of the metadata.

[IIP-MD06]

Implementations MUST support the interpretation and application of metadata as defined by the SAML V2.0 Metadata Interoperability Profile [SAML2MDIOP]. It follows that implementations MUST be capable of interoperating (leading to success or failure as dictated by default configuration) with any number of SAML peers for which metadata is available, without additional inputs or separate configuration. In accordance with the SAML V2.0 Metadata Interoperability Profile [SAML2MDIOP], metadata MUST be:

```
“usable as a self-contained vehicle for communicating trust such that a user of a
conforming implementation can be guaranteed that any and all rules for processing
digital signatures, encrypted XML... can be derived from the metadata alone, with no
additional trust requirements imposed.”
```

As an example, a separate trust store must not be required to verify the signature on a signed SAML message or to negotiate a TLS connection for SOAP messaging. The latter is a particularly common limitation of many TLS libraries; such libraries must not be used

unmodified if TLS is used for SAML messaging. If this is a concern, signed and encrypted messages should be exchanged, or implementations should confine themselves to supporting front-channel bindings.

Note that this requirement does not preclude supporting a variety of configuration options on a per-peer (or other) basis; it simply requires that default behavior be possible without such.

Key Rollover

[IIP-MD07]

Implementations MUST have the ability to consume and make use of any number of signing keys bound to a single role descriptor in metadata. When verifying digital signatures, implementations MUST attempt to use each signing key (in unspecified order) until the signature is successfully verified or there are no remaining keys, in which case signature verification fails.

[IIP-MD08]

If an implementation supports outbound encryption, it MUST be able to consume any number of encryption keys bound to a single role descriptor in metadata. If multiple encryption keys are specified, any one of them may be used to encrypt outbound messages.

Algorithm Support

The migration from suspicious and broken algorithms deployed in production systems requiring a coordinated update at a single point in time is not feasible in large federations. An alternative strategy is the support of both good and bad algorithms at the endpoints for some time to relax the schedule for system updates. Negotiating better algorithms automatically provides immediate benefit to communications within the group of updated entities. Thus the network as a whole is not completely held back because of entity operators unable/unwilling to update. Algorithm support for TLS endpoints is out of scope for SAML metadata.

[IIP-MD09]

Implementations MUST be capable of publishing the cryptographic capabilities of their runtime configurations with regard to XML Signature and Encryption. It is RECOMMENDED that they support dynamic generation and export of this information and provide it in a machine-readable format that can be included in metadata according to [SAML2MetaAlgSup].

[IIP-MD10]

If a SAML peer has declared algorithm support according to [SAML2MetaAlgSup] in its metadata, Identity providers MUST and Service Providers SHOULD limit the use of algorithms for XML Signature and Encryption to these declared algorithms in the messages they produce for that peer.

As noted by [SAML2MetaAlgSup], the use of metadata to negotiate algorithms requires that the exchange and verification of metadata be subject to appropriate security controls. Implementations must always be prepared to reject the use of algorithms that are deemed insufficiently secure. In all cases, the ultimate decision as to algorithm choice is up to the implementation, as necessarily limited by local configuration. An implementation may be unable to successfully complete a request in the event that a mutually supported and acceptable algorithm does not exist.

Avoiding Common Errors

[IIP-MD11]

An `<md:KeyDescriptor>` element in metadata that contains no `use` XML attribute MUST be valid as both a signing and encryption key. This is clarified in E62 of the SAML V2.0 Errata [SAML2Errata]:

```
If the use attribute is omitted, then the contained key information is applicable to
both of the above uses.
```

[IIP-MD12]

Support for any number of long-lived, self-signed end entity certificates is REQUIRED as is support for expired certificates, and certificates signed with any digest algorithm. The SAML V2.0 Metadata Interoperability Profile [SAML2MDIOP] states:

```
In the case of an X.509 certificate, there are no requirements as to the content of
the certificate apart from the requirement that it contain the appropriate public
key. Specifically, the certificate may be expired, not yet valid, carry critical or
non-critical extensions or usage flags, and contain any subject or issuer. The use of
the certificate structure is merely a matter of notational convenience to communicate
a key and has no semantics in this profile apart from that.
```

2.3. Web Browser SSO

[IIP-SSO01]

Implementations MUST support the SAML V2.0 Web Browser SSO Profile as defined in [SAML2Prof] and as updated by [SAML2Errata].

[IIP-SSO02]

Implementations MUST support the HTTP-Redirect and HTTP-POST bindings for authentication requests.

[IIP-SSO03]

Implementations MUST support the HTTP-POST binding for authentication and error responses.

[IIP-SSO04]

Implementations MUST support the signing of assertions and responses, both together and independently.

[IIP-SSO05]

Implementations MUST support the following SAML V2.0 name identifier formats, in accordance with the normative obligations associated with them by [SAML2Core] sect. 8.3:

- urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
- urn:oasis:names:tc:SAML:2.0:nameid-format:transient

[IIP-SSO06]

Implementations MUST support the consumption of peer configuration values from SAML metadata, without additional inputs or separate configuration, for any metadata element that: (a) is identified as "MUST" or "MAY" in the "Use of Metadata" section for the Web Browser SSO Profile in [SAML2Prof] (section 4.1.6), and (b) corresponds to settings supported by the implementation.

Note that the above requires that most metadata elements identified in [SAML2Prof] section 4.1.6 must be configurable via SAML metadata. It only applies, however, to functionality that is supported in a given implementation. For example, if an implementation allows a deployer to control AttributeProfile configuration values, that implementation MUST support the configuration of these values via the `<md:AttributeProfile>` *element. This requirement by itself does not necessitate attribute profile support.*

This requirement does not prevent implementations from offering additional mechanisms for deployers to set or modify SAML peer configuration values. For example, it is acceptable to allow deployers a choice between manual peer configuration and metadata-based configuration, to provide them with the ability to override values found in metadata, or to add options in the peer configuration that cannot be expressed via SAML metadata.

[IIP-SSO07]

Unless specifically called out by subsequent requirements in this profile, it is OPTIONAL for implementations to support the inclusion of optional elements and attributes in the protocol messages and assertions issued. It is REQUIRED that implementations successfully process messages and assertions containing any optional content they do not support (i.e., such content MUST either result in errors or be ignored, as directed by the processing rules for the element or attribute in [SAML2Core]).

For example, the `<saml:Subject>` *and* `<saml:Conditions>` *elements are optional elements that may appear in a* `<samlp:AuthnRequest>` *message, and it is therefore optional for an SP implementation to support their inclusion in requests. They have differing semantics for the IdP: the former has required semantics and the latter optional semantics. As such, an implementation that receives a* `<saml:Subject>` *may ignore it, but must return an error in so doing, whereas an implementation that receives a* `<saml:Conditions>` *element may also ignore it, but is under no additional obligation.*

2.4. Extensibility

Support for SAML extensibility is of paramount importance since it allows deployments to evolve and meet future needs. The SAML standard has explicit support for extensibility in metadata, protocol messages, and assertions. Most extension points in SAML have optional semantics, which means that ignoring extension content is a valid and acceptable practice.

[IIP-EXT01]

Implementations MUST successfully consume any and all well-formed extensions. Unless otherwise noted in this profile, the content of `<samlp:Extensions>`, `<md:Extensions>`, and `<saml:Advice>` elements MAY be ignored but MUST NOT result in software failures. Additionally, any element established in [SAML2MD-xsd] or [SAML2-xsd] with a type definition containing an <**xsd:anyAttribute**> subelement may include undefined attribute content. This content MAY likewise be ignored but MUST NOT result in software failures.

2.5. Cryptographic Algorithms

[IIP-ALG01]

Implementations MUST support the digest algorithms identified by the following URIs in conjunction with the creation and verification of XML Signatures [XMLSig]:

- `http://www.w3.org/2001/04/xmlenc#sha256` [XMLEnc]

[IIP-ALG02]

Implementations MUST support the signature algorithms identified by the following URIs in conjunction with the creation and verification of XML Signatures [XMLSig]:

- `http://www.w3.org/2001/04/xmldsig-more#rsa-sha256` [RFC4051]

[IIP-ALG03]

Implementations SHOULD support the signature algorithms identified by the following URIs in conjunction with the creation and verification of XML Signatures [XMLSig]:

- `http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256` [RFC4051]

[IIP-ALG04]

Implementations MUST support the block encryption algorithms identified by the following URIs in conjunction with the use of XML Encryption [XMLEnc]:

- `http://www.w3.org/2009/xmlenc11#aes128-gcm` [XMLEnc]
- `http://www.w3.org/2009/xmlenc11#aes256-gcm` [XMLEnc]

[IIP-ALG05]

Implementations MAY support the block encryption algorithms identified by the following URIs in conjunction with the use of XML Encryption [XMLEnc] for backwards compatibility:

- `http://www.w3.org/2001/04/xmlenc#aes128-cbc` [XMLEnc]
- `http://www.w3.org/2001/04/xmlenc#aes256-cbc` [XMLEnc]

The use of these algorithms is widespread at the time of authoring, but is known to be broken [XMLEncBreak] and should be avoided in new applications. Implementations supporting them SHOULD warn on use.

[IIP-ALG06]

Implementations MUST support the key transport algorithms identified by the following URIs in conjunction with the use of XML Encryption [XMLEnc]:

- `http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p` [XMLEnc]
- `http://www.w3.org/2009/xmlenc11#rsa-oaep` [XMLEnc]

The following DigestMethod algorithms MUST be supported for both of the above key transport algorithms:

- `http://www.w3.org/2001/04/xmlenc#sha256`

The following DigestMethod algorithms SHOULD be supported for both of the above key transport algorithms for backward compatibility.

- `http://www.w3.org/2000/09/xmldsig#sha1`

The default mask generation function (MGF1 with SHA1) MUST be supported for the KeyTransport algorithm identified by `http://www.w3.org/2009/xmlenc11#rsa-oaep`.

[IIP-ALG07]

This document is not normative with respect to TLS security. It is, however, RECOMMENDED that implementers consider [RFC7457] and current best practice reflected in documents such as [BetterCrypto].

[IIP-ALG08]

Implementations MUST support the ability to prevent the use of particular algorithms such that any attempt to configure or select them will result in failure. The set of such algorithms MUST be configurable and it is RECOMMENDED that the default set include:

- Digest
  - `http://www.w3.org/2001/04/xmldsig-more#md5` [RFC4051]
- Signature
  - `http://www.w3.org/2001/04/xmldsig-more#rsa-md5` [RFC4051]
- Key Transport
  - `http://www.w3.org/2001/04/xmlenc#rsa-1_5` [XMLEnc]

3. Service Provider Requirements

3.1. Web Browser SSO

[IIP-SP01]

Service Providers MUST support the consumption of `<saml:Attribute>` elements containing any arbitrary **xs:string** value in the `Name` attribute and any arbitrary **xs:anyURI** value in the `NameFormat` attribute.

[IIP-SP02]

Service Providers MUST support the consumption of `<saml:AttributeValue>` elements containing any "simple" element content; that is, element content consisting only of text nodes, not mixed/complex content that may contain nested XML elements. It is OPTIONAL to support complex content. Service Providers MUST NOT require the presence of the `xsi:type` XML attribute.

[IIP-SP03]

Service Providers MUST be capable of generating `<samlp:AuthnRequest>` messages without a `<samlp:NameIDPolicy>` element and with a `<samlp:NameIDPolicy>` element but no `Format` attribute.

[IIP-SP04]

Service Providers MUST support IdP discovery in accordance with [IdPDisco].

Note that this requirement only implies support for the simple redirection conventions defined by that profile and does not demand implementation of an actual discovery interface, though that is of course not precluded.

Also note that discovery mechanisms SHOULD use SAML metadata to determine the endpoint(s) to which requests are to be issued.

[IIP-SP05]

Service Providers MUST support the processing of responses from any number of issuing IdPs for any given resource URL. That is, it MUST NOT be a restriction of an implementation that multiple IdPs can only be supported by requiring distinct resource URLs for each IdP.

Note that the requirement for support of [IdPDisco] in [IIP-SP09] leads naturally to the ability to satisfy this requirement.

[IIP-SP06]

Service Providers MUST be capable of generating `<samlp:AuthnRequest>` messages with a `<samlp:RequestedAuthnContext>` element containing the **exact** comparison method and any number of `<samlp:AuthnContextClassRef>` elements.

[IIP-SP07]

Service Providers MUST support the acceptance or rejection of assertions based on the content of the `<saml:AuthnContext>` element.

[IIP-SP08]

Service Providers MUST support decryption of `<saml:EncryptedAssertion>` elements. In order to fully support key rollover, Service Providers MUST be configurable with at least two decryption keys. When decrypting assertions, they MUST attempt to use each decryption key (in unspecified order) until the assertion is successfully decrypted or there are no more keys, in which case decryption fails.

[IIP-SP09]

Service Providers MUST support deep linking and maintain the direct addressability of protected resources in the presence of Web Browser SSO. That is, it MUST be possible to request an arbitrary protected resource and (authorization permitting) have it supplied as the result of a successful SAML SSO profile exchange. In addition, it is RECOMMENDED that Service Providers support the preservation of POST bodies across a successful SSO profile exchange, subject to size limitations dictated by policy or implementation constraints.

The SAML binding-specific RelayState feature is typically used to maintain state required to satisfy both of these requirements, the exact detail of which is left to implementations.

Support for unsolicited responses (or so-called IdP-initiated SSO) is not a substitute for this requirement.

3.1.1. Avoiding Common Errors

[IIP-SP10]

Service Providers MUST NOT fail or reject responses due to the presence of unrecognized `<saml:Attribute>` elements.

[IIP-SP11]

Service Providers MUST NOT treat the `FriendlyName` attribute normatively or make comparisons based on its value.

[IIP-SP12]

Service Providers MUST NOT require that name identifiers with a format of 'urn:oasis:names:tc:SAML:2.0:nameid-format:persistent' be overloaded with semantics or content beyond what is outlined in [SAML2Core] sect. 8.3.7.

Note that if the name identifier format identifiers defined in [SAML2Core] *are inapplicable to a given use case, it should be possible for new ones to be established. Implementations not specific to a single deployment should support the use of arbitrary formats.*

[IIP-SP13]

Service Providers MUST support the ability to reject unsigned `<samlp:Response>` elements and SHOULD do so by default.

Note that this requirement is intended to offer some protection against known attacks when XML Encryption is used with AES in CBC mode. While the use of AES-GCM is strongly preferred, requiring signed responses limits the potential range of attack sources to those with verifiable signatures.

3.2. Single Logout

[IIP-SP14]

Service Providers SHOULD support the SAML V2.0 SingleLogout profile [SAML2Prof], as updated by [SAML2Errata]. Service Providers that claim support for this profile MUST be capable of issuing logout requests. It is OPTIONAL to support consumption of logout requests and responses.

The intent is to allow for the minimum support possible while still enabling applications to initiate logouts in a federated environment. Thus, implementations must be able to generate a request to the Identity Provider, but may ignore (or not even support) responses, and may also omit support for inbound logout requests entirely.

[IIP-SP15]

Service Providers that support the SAML V2.0 SingleLogout profile MUST support the HTTP-Redirect binding for logout requests and responses.

[IIP-SP16]

Service Providers that support the SAML V2.0 SingleLogout profile MUST support decryption of <saml:EncryptedID> elements in logout requests. In order to fully support key rollover, Service Providers MUST be configurable with at least two decryption keys. When decrypting encrypted identifiers, they MUST attempt to use each decryption key (in unspecified order) until the identifier is successfully decrypted or there are no more keys, in which case decryption fails.

[IIP-SP17]

Service Providers that support the SAML V2.0 SingleLogout profile MUST support the consumption of peer configuration values from SAML metadata, without additional inputs or separate configuration, for any element listed in the "Use of Metadata" section for the Single Logout Profile in [SAML2Prof] (section 4.4.5).

4. Identity Provider Requirements

4.1. Web Browser SSO

[IIP-IDP01]

Identity Providers MUST support the generation of `<saml:Attribute>` elements containing any arbitrary **xs:string** value in the `Name` attribute and any arbitrary **xs:anyURI** value in the `NameFormat` attribute.

[IIP-IDP02]

Identity Providers MUST be capable of determining whether or not to include specific SAML attributes (or specific values) in a response based on the the entityID of the relying party.

[IIP-IDP03]

Identity Providers MUST be capable of determining whether or not to include specific SAML attributes (or specific values) in a response based on the presence of `<mdattr:EntityAttributes>` extension elements [MetaAttr] found in the metadata for a relying party.

[IIP-IDP04]

Identity Providers MUST be capable of determining whether or not to include specific SAML attributes (or specific values) in a response based on the presence of `<md:AttributeConsumingService>` elements (containing `<md:RequestedAttribute>` elements) found in the metadata for a relying party, including the value of the enclosed `isRequired` XML attribute. Accordingly, they MUST support the `AttributeConsumingServiceIndex` attribute in `<samlp:AuthnRequest>` messages as a means of determining the appropriate `<md:AttributeConsumingService>` element to process.

<md:RequestedAttribute> elements in metadata can be used to help automate attribute release configurations in IdP deployments. An IdP could, for instance, be configured to release attributes requested in metadata, typically in combination with other criteria. Example criteria include the acquisition of user consent and/or the presence of a particular qualifying entity attribute (see IIP-IDP04) for the relying party.

[IIP-IDP05]

Identity Providers MUST support the issuance of `<samlp:Response>` messages (with appropriate status codes) in the event of an error condition, provided that the user agent remains available and an acceptable location to which to deliver the response is known. The criteria for "acceptability" of a response location are not formally specified, but are subject to Identity Provider policy and reflect its responsibility to protect users from being sent to untrusted or possibly malicious parties.

[IIP-IDP06]

Identity Providers MUST support the `ForceAuthn` attribute in `<samlp:AuthnRequest>` messages as defined in [SAML2Core]. The authentication mechanisms within an implementation MUST have access to the ForceAuthn indicator so that their behavior may be influenced by its value.

Note that ForceAuthn is most commonly used for privilege escalation or to initiate explicit user approval for an action. When using forms-based username/password authentication, this is typically accomplished by re-prompting the user for credentials. When more sophisticated authentication schemes are employed (certificates, hard tokens, GSS-API), it is less clear what the correct ForceAuthn behavior should be. Thus, in addition to following [SAML2Core], *implementations should allow for the authentication process to be modified based on the existence of ForceAuthn in a request. This provides the necessary flexibility for specific needs to be met.*

[IIP-IDP07]

Identity Providers MUST support the `IsPassive` attribute in `<samlp:AuthnRequest>` messages as defined in [SAML2Core].

[IIP-IDP08]

Identity Providers MUST support the `<saml:RequestedAuthnContext>` **exact** comparison method in `<samlp:AuthnRequest>` messages as defined in [SAML2Core].

[IIP-IDP09]

Identity Providers MUST support encryption of assertions. Support for encryption of identifiers and attributes is OPTIONAL.

[IIP-IDP10]

Identity Providers MUST support the `<samlp:NameIDPolicy>` element in `<samlp:AuthnRequest>` messages as defined in [SAML2Core].

[IIP-IDP11]

Identity Providers MUST be capable of generating `<saml:Assertion>` elements without a `<saml:NameID>` element in the `<saml:Subject>` element.

[IIP-IDP12]

Identity Providers MUST support the `AssertionConsumerServiceURL`, `ProtocolBinding`, and `AssertionConsumerServiceIndex` attributes in `<samlp:AuthnRequest>` messages for the identification of the response endpoint and binding, as defined in [SAML2Core].

4.2. Enhanced Client or Proxy

[IIP-IDP13]

Identity Providers MUST support the SAML V2.0 Enhanced Client or Proxy Profile Version 2.0 [SAML2ECP]. Full conformance is OPTIONAL, but implementations MUST support "Bearer" subject confirmation and verification of channel bindings. All applicable Web Browser SSO

requirements in this profile apply, excepting IIP-SSO02 and IIP-SSO03 since those bindings do not apply to the use of ECP. This requirement does not apply to token translation Proxies.

[IIP-IDP14]

Identity Providers MUST support the use of HTTP Basic Authentication [RFC2617] to authenticate the user agent. Other forms of authentication MAY be supported.

[IIP-IDP15]

Identity Providers MUST support the generation and inclusion of a random key in accordance with [SAML-EC], Section 5.3.1.

[IIP-IDP16]

Identity Providers MUST support the consumption of peer configuration values from SAML metadata, without additional inputs or separate configuration, for any element listed in the "Use of Metadata" section in [SAML2ECP] (section 2.3.10).

4.3. Single Logout

[IIP-IDP17]

Identity Providers MUST support the SAML V2.0 SingleLogout profile [SAML2Prof], as updated by [SAML2Errata], and the SAML V2.0 Asynchronous Single Logout Protocol Extension [SAML2ASLO]. It is OPTIONAL to support propagation of logout requests to other session participants.

[IIP-IDP18]

Identity Providers MUST support the HTTP-Redirect binding for logout requests and responses.

[IIP-IDP19]

Identity Providers MUST support decryption of `<saml:EncryptedID>` elements in logout requests. In order to fully support key rollover, Identity Providers MUST be configurable with at least two decryption keys. When decrypting encrypted identifiers, they MUST attempt to use each decryption key (in unspecified order) until the identifier is successfully decrypted or there are no more keys, in which case decryption fails.

[IIP-IDP20]

Identity Providers MUST support the consumption of peer configuration values from SAML metadata, without additional inputs or separate configuration, for any element listed in the "Use of Metadata" section for the Single Logout Profile in [SAML2Prof] (section 4.4.5).

[IIP-IDP21]

Identity Providers MUST be able to generate name identifiers with a format of `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` in a manner that allows deployers to avoid assignment of identifiers that differ only by case to two different subjects.

The above requirement avoids conficts in the common case that consuming applications do not support case-aware or case-sensitive processing of identifiers. Use of base32 encoding [RFC4648] is a common mechanism for meeting this requirement.

5. References

- [RFC2119] IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels, March 1997. http://www.ietf.org/rfc/rfc2119.txt
- [RFC2616] IETF RFC 2616, Hypertext Transfer Protocol – HTTP/1.1, June 1999. http://www.ietf.org/rfc/rfc2616.txt
- [RFC4648] IETF RFC 4648, The Base16, Base32, and Base64 Data Encodings, October 2006. http://www.ietf.org/rfc/rfc4648.txt
- [SAML2Core] OASIS Standard, Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
- [SAML2Errata] OASIS Approved Errata, SAML Version 2.0 Errata 05, May 2012. http://docs.oasis-open.org/security/saml/v2.0/sstc-saml-approved-errata-2.0.pdf
- [SAML2Meta] OASIS Standard, Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf
- [SAML2Bind] OASIS Standard, Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf
- [SAML2Prof] OASIS Standard, Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf
- [SAML2-xsd] XML Schema, SAML V2.0 Assertion Schema, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-schema-assertion-2.0.xsd
- [SAML2MD-xsd] XML Schema, Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-schema-metadata-2.0.xsd

- [SAML2MDIOP] OASIS Committee Specification, SAML V2.0 Metadata Interoperability Profile Version 1.0, August 2009. http://docs.oasis-open.org/security/saml/Post2.0/sstc-metadata-iop.pdf
- [SAML2MetaAlgSup] OASIS Committee Specification, SAML V2.0 Metadata Profile for Algorithm Support Version 1.0, February 2011. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-metadata-algsupport-v1.0-cs01.pdf
- [SAML2ECP] OASIS Committee Specification, SAML V2.0 Enhanced Client or Proxy Profile Version 2.0, August 2013. http://docs.oasis-open.org/security/saml/Post2.0/saml-ecp/v2.0/cs01/saml-ecp-v2.0-cs01.pdf
- [MetaUi] OASIS Committee Specification, SAML V2.0 Metadata Extensions for Login and Discovery User Interface Version 1.0, April 2012. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-metadata-ui/v1.0/sstc-saml-metadata-ui-v1.0.pdf
- [MetaAttr] OASIS Committee Specification, SAML V2.0 Metadata Extension for Entity Attributes Version 1.0, August 2009. http://docs.oasis-open.org/security/saml/Post2.0/sstc-metadata-attr-cs-01.pdf
- [SAML2ASLO] OASIS Committee Specification, SAML V2.0 Asynchronous Single Logout Profile Extension Version 1.0, November 2012. http://docs.oasis-open.org/security/saml/Post2.0/saml-async-slo/v1.0/cs01/saml-async-slo-v1.0-cs01.pdf
- [MDQ] IETF Internet-Draft, Metadata Query Protocol, July 2017. https://tools.ietf.org/html/draft-young-md-query-07
- [SAML-MDQ] IETF Internet-Draft, SAML Profile for the Metadata Query Protocol, October 2014. https://tools.ietf.org/html/draft-young-md-query-saml-07
- [IdPDisco] OASIS Committee Specification, Identity Provider Discovery Service Protocol and Profile, March 2008. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-idp-discovery.pdf
- [RFC4051] IETF RFC 4051, Additional XML Security Uniform Resource Identifiers, April 2005. https://www.ietf.org/rfc/rfc4051.txt
- [XMLEnc] D. Eastlake et al. XML Encryption Syntax and Processing Version 1.1. World Wide Web Consortium Recommendation, April 2013. https://www.w3.org/TR/xmlenc-core1/
- [XMLSig] D. Eastlake et al. XML-Signature Syntax and Processing, Version 1.1. World Wide Web Consortium Recommendation, April 2013. https://www.w3.org/TR/xmldsig-core1/
- [SAML-EC] IETF Internet Draft, SAML Enhanced Client SASL and GSS-API Mechanisms, October 2017. https://tools.ietf.org/html/draft-ietf-kitten-sasl-saml-ec-16

- [BetterCrypto] BetterCrypto.org, Applied Crypto Hardening. https://bettercrypto.org
- [RFC7457] IETF RFC 7457, Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS), February 2015. https://www.ietf.org/rfc/rfc7457.txt
- [XMLEncBreak] Jager and Somorovsky, How to Break XML Encryption, October 2011. http://www.nds.rub.de/media/nds/veroeffentlichungen/2011/10/22/HowToBreakXMLenc.pdf