Australian Government
Digital Transformation Agency

SAML 2.0 Profile

Trusted Digital Identity Framework
March 2019, version 1.2

dta

Digital Transformation Agency

Licence

Use of the Coat of Arms

The terms under which the Coat of Arms can be used are detailed on the It's an Honour website (http://www.itsanhonour.gov.au)

Conventions

TDIF documents refenced by this document are denoted in italics. For example, *TDIF: Overview and Glossary* is a reference to the TDIF document titled Overview and Glossary.

The key words "**MUST**", "**MUST NOT**", "**SHOULD**", "**SHOULD NOT**", and "**MAY**" in this document are to be interpreted as described in the current version of the *TDIF: Overview and Glossary.*

Contact us

Digital Transformation Agency is committed to providing web accessible content wherever possible. If you are having difficulties accessing this document or have questions or comments regarding this document please email the Director, Digital Identity Policy at identity@dta.gov.au.

Document Management

The TDIF Accreditation Authority has reviewed and endorsed this document for release.

Change log

Version	Date	Author	Description of the changes
0.01	Jan 2018	BF	Initial version.
0.02	Aug 2018	TM	Updates from stakeholder consultation.
1.0	Aug 2018		Endorsed for release by the TDIF Accreditation Authority.
1.1	Sep 2018	TM	Content updates.
1.2	Mar 2019	SJP	Minor content updates to align with other TDIF documents.

Contents

1 Introduction

Agencies and organisations that apply to be accredited under the TDIF undergo a series of rigorous evaluations across all aspects of their identity service operations. The *TDIF: Accreditation Process* requires Applicants to demonstrate their identity service is usable, privacy enhancing and is secure and resilient to cyber threats. The intent of these evaluations is to determine whether the Applicant's identity service meets the TDIF Guiding Principles[1] and whether it is suitable to join the identity federation.

This document forms part of the TDIF technical integration requirements. This document provides the SAML 2.0 Profiles for the following interactions:

- Interactions between a Relying Party and an Identity Exchange.
- Interactions between an Identity Provider and an Identity Exchange.

The intended audience for this document includes:

- Applicants and Accredited Providers.
- Relying Parties.
- TDIF Accreditation Authority.

1.1 Relationship to Other Standards

This document should be read in conjunction with the following documents:

- *TDIF: Overview and Glossary*, which provides a high-level overview of the TDIF including its scope and objectives and the definition of key terms. It also includes an architecture overview that describes the functions of the participants and how they interact with each other.
- *TDIF: Technical Requirements,* which provides the core technical requirements for each participant in the TDIF architecture.

[1] See *TDIF: Overview and Glossary* for further information on the TDIF guiding principles.

2 The Identity Exchange Common Profile Requirements

2.1 Notation

- Conventional XML namespaces are used throughout the listings in this profile specification to stand for their respective namespaces as follows:
  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, urn:oasis:names:tc:SAML:2.0:metadata:attribute.

2.2 General

2.2.1 Clock skew

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

Items to which this directive apply include but are not limited to:

- `NotBefore`,
- `NotOnOrAfter`, `and`
- `validUntil`.

These attributes are found on the following elements:

- `Conditions`,
- `SubjectConfirmationData`,
- `LogoutRequest`,

- `EntityDescriptor`,
- `EntitiesDescriptor`,
- `RoleDescriptor`, and
- `AffiliationDescriptor`

Configurability is a suggested practice but tolerances of 3-5 minutes are considered reasonable defaults.

2.2.2 Data Size

Where 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 containing up to 256 characters. This requirement applies to both user defined types and the types defined within the SAML standards such as transient and persistent `NameIDs`.

All data sizes and constraints are specified within **[TDIF.Attributes]**.

2.2.3 Document Type Definitions

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

2.3 Metadata and Trust Management

Although metadata is optional in the original SAML 2.0 standards, it is now recognised that it is a critical component of all modern SAML software. To support a scalable federation model, implementations **MUST** adhere to the following procedures related to the exchange and validation of metadata.

2.3.1 Metadata Exchange

2.3.1.1 Metadata Acquisition Method

Implementations **MUST** support the routine consumption of SAML metadata from a remote location via HTTP/1.1 **[RFC 2616]** on a scheduled or recurring basis with the contents applied automatically upon successful validation. HTTP/1.1 redirects (status codes 301, 302, and 307) **MUST** be honoured. Implementations **MUST** support the consumption of SAML metadata rooted in both `<md:EntityDescriptor>` and `<md:EntitiesDescriptor>` elements by this mechanism. Any number of child elements must be allowed for `<md:EntitiesDescriptor>`.

This method is less flexible and less efficient/scalable for larger metadata aggregates than the Metadata Query Protocol.

2.3.1.2 Metatdata Query Protocol

Identity Providers **SHOULD** 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 utilise metadata from one or more MQD responders for any per entity from which a SAML protocol message is received.

2.3.1.3 Validation

Implementations **MUST** validate the authenticity and integrity of SAML metadata by verifying an enveloped XML signature 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 within X.509 certificates but it **MUST** be possible to ignore the other content in the certificate and validate the XML Signature based on the public key.

It **MUST** be possible to limit the use of a trusted key to a single metadata source.

Implementations **MUST** reject metadata if any one of the following conditions is true:

- The `validUnitl` XML attribute on the root element is missing.

- The value of the `validUntil` XML attribute on the root element is a **xsd:dateTime** in the past.
- The value of the `validUnitl` 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: this requirement applies to the root element only. Any `validUnitl` XML attributes in child elements **MUST** be processed in accordance with **[SAML2Meta].**

2.3.2 Metadata Usage

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

- SAML V2.0 Metadata **[SAML2Meta]** as updated by Errata **[MetaAttr]**.
- SAML V2.0 Metadata Schema **[SAML2MD-xls]**.
- SAML V2.0 Metadata Interoperability Profile **[SAML2MDIOP]**.
- SAML V2.0 Metadata Extension for Algorithm Support **[SAML2MetaAlgSup]**.

Implementations **MAY** support:

- SAML V2.0 Metadata Extension for Entity Attributes **[MetaAttr]**.

SPs **SHOULD** support:

- SAML V2.0 Metadata Extensions for Login and Discovery User Interface **[MetaUI].**

The list above is not intended to be exhaustive but includes all material relevant to functionality required by these profiles. In accordance with the Extensibility section 2.5 below, other metadata may be present and **MUST NOT** prevent the consumption and use of the metadata.

Implementations **MUST** support the interpretation and application of metadata as defined by SAML 2.0 Metadata Interoperability Profile **[SAML2MDIOP]**.

Implementations **MUST** be able to interoperate with any number of SAML peers for which metadata is available without additional inputs or separate configuration. This requirement does not preclude supporting a variety of configuration options on a peer to peer or other basis; it simply requires that the default behaviour be possible.

2.3.2.1 Key Rollover

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 until the signature is verified or there are no remaining keys and the signature verification is then deemed to have failed.

If an implementation supports out bound 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.

2.3.2.2 Algorithm Support

Migration from weak or broken algorithms deployed in production systems requires a coordinated update at a single point in time and is not feasible for large federations. Implementations **SHOULD** be able to support the use of good and bad algorithms for some time to relax the schedule of updates. Implementations **SHOULD** select the most secure algorithm from those that are available.

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 the can be included in metadata according to **[SAML2MetaAlgSup]**.

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 those declared in the messages they produce for that peer.

2.3.2.3 Avoiding Common Errors

A `<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 2.0 Errata.

2.4 Web Browser SSO

Implementations **MUST** support the SAML 2.0 Web Browser SSO profile as defined in and as updated by **[SAML2Errata]**.

Identity Providers **MUST** support both the HTTP-Redirect and HTTP-POST bindings for authentication requests.

Service Providers **MUST** support either the HTTP-Redirect and HTTP-POST bindings for authentication requests.

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

Implementations **MUST** support the following SAML 2.0 name identifier formats, in accordance with the normative obligations associated with them by **[SAML2Core]** section 8.3:

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

Implementations **MUST** support the consumption of peer configuration values from SAML metadata, without additional inputs or separate configuration, for any metadata element that:

- 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 corresponds to settings supported by the implementation.

- Unless specifically noted 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. this content must result in errors or be ignored, as directed by the processing rules for the element or attribute in **[SAML2Core]**.

2.5 Extensibility

Support for extensibility 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 valid and acceptable practice.

Implementations **MUST** successfully consume any well-formed extension. Unless otherwise noted in these profiles the content of `<samlp:Extension>`, `<md:Extensions>` and `<saml:Advice>` elements **MAY** be ignored but **MUST NOT** result in software failures. Any element established in **[SAML2MD-xls]** or **[SAML2-xls]** with a type definition containing an **xsd:anyAttribute** sub-element may include undefined attribute content. This content **MAY** also be ignored but **MUST NOT** result in software failures.

2.6 Cryptographic Algorithms

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]**.

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

- http://www.w3.org/2001/04/xmldsig-more - rsa-sha256 **[RFC 4051]**.

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 **[RFC 4051]**.

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.
- http://www.w3.org/2009/xmlenc11 - aes256-gcm.

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]**.

Note: These algorithms should be avoided for new applications. Implementations supporting them **SHOULD** warn on use.

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/2001/04/xmlenc - rsa-oaep **[XMLEnc]**.

The following `DigestMethod` Algorithm **MUST** be supported for both of the algorithms above.

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

The following `DigestMethod` algorithms **SHOULD** be supported for both of the key transport algorithms shown above for backwards compatibility only.

- 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.

This document is not normative with respect to TLS security. It is RECOMMENDED that implementers consider **[RFC 7457]** however.

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 algorithms **MUST** be configurable and it is RECOMMENDED that the default set include:

Digest

- http://www.w3.org/2001/04/xmldsig-more - md5 **[RFC 4051]**.

Signature

- http://www.w3.org/2001/04/xmldsig-more - rsa-md5 **[RFC 4051]**.

Key Transport

- http://www.w3.org/2001/04/xmlenc - rsa-1_5 **[XMLEnc]**.

2.7 Authentication Context Class Reference

Assurance levels are defined in section 2.2.8 of the *TDIF: Attribute Profile* **[TDIF.Attributes]**.

The TDIF Levels of Assurance are implemented in SAML using the standard Authentication Context Class Reference element as defined in **[SAML2Core]**. The `<saml:AuthnContextClassRef>` should be considered the same as the `acr` claim from OIDC when performing translation between the two protocols.

A single acr value can be requested by the Relying Party to specify the minimum level of assurance that is required by the Relying Party as described in section 3.1.3.

The Service Provider **MAY** request a single `<saml:AuthnContextClassRef>` that will meet the SPs minimum Identity and Credential requirements.

```
<samlp:RequestedAuthnContext Comparison="minimum">
    <saml:AuthnContextClassRef>
        urn:id.gov.au:tdif:acr:ip2:cl3
    </saml:AuthnContextClassRef>
</samlp:RequestedAuthnContext>
```

The IdP **MUST** return the `<saml:AuthnContextClassRef>` that is the representation of the assurance levels that are defined in defined in section 2.2.8 of the *TDIF: Attribute Profile* **[TDIF.Attributes]**.

```
<saml:AuthnStatement AuthnInstant="2017-07-17T01:01:48Z"
        SessionNotOnOrAfter="2017-07-17T09:01:48Z"
        SessionIndex="_be9967abd904ddcae3c0eb4189adbe3f71e327cf93">
    <saml:AuthnContext>
        <saml:AuthnContextClassRef>
                urn:id.gov.au:tdif:acr:ip2:cl3
        </saml:AuthnContextClassRef>
    </saml:AuthnContext>
</saml:AuthnStatement>
```

The SP is required to determine if the `<saml:AuthnContextClassRef>` meets the minimum requirements for the authentication context that was specified.

2.8 Attributes

Attributes supported as part of the federation are described in **[TDIF.Attributes]**.

As part of the establishment of the federation, the attributes required by the Relying party, whether this is an SP talking to an Identity Exchange or an Identity Exchange talking to an Identity Provider. These agreed attributes will be returned as part of the `<saml:AttributeStatement>` returned in the `<samlp:Response>`.

```
<saml:AttributeStatement>
    <saml:Attribute Name="family_name"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">
                Michaels
        </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute Name="given_name"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
       <saml:AttributeValue xsi:type="xs:string">
            Stephen
        </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute Name="birthdate"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
       <saml:AttributeValue xsi:type="xs:string">
            1974-02-29
        </saml:AttributeValue>
    </saml:Attribute>
</saml:AttributeStatement>
```

Attributes **MAY** be requested as part of the SAML Authentication Request. These attributes are requested through the Extension element. The Sender and the

Recipient of the request **SHOULD** agree to the semantics of data sent this way. An example of the Extension element with a request for attributes is shown in the non-normative example below:

```
<samlp:AuthnRequest>
   ....
   <samlp:Extensions>
      <Attribute name="family_name">
         <Value></Value>
      </Attribute>
      <Attribute name="given_name">
         <Value></Value>
      </Attribute>
      <Attribute name="email">
         <Value></Value>
      </Attribute>
      <Attribute name="audit_id">
         <Value></Value>
      </Attribute>
   </samlp:Extensions>
   ....
</samlp:AuthnRequest>
```

3 Relying Party (SP) to Identity Exchange (IdP) Profile

3.1 Relying Party Profile (SP)

In this section all references to Service Provider or SP refer to the Relying Party and any references to Identity Provider or IdP refer to the Identity Exchange.

3.1.1 Web Browser SSO

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.

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. There may be some future attributes defined within the **[TDIF.Attributes]** that **MAY** require the Service Provider to support complex content.

Service Providers **MUST NOT** require the presence of the **xsi:type** XML attribute.

Service providers **MUST** generate `<saml:AuthnRequest>` messages with a `<samlp:NameIDPolicy>` element with a `<samlp:NameIDPolicy>` Format of **urn:oasis:names:tc:SAML:2.0:nameid-format:persistent** and `AllowCreate` set to `true`.

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

Note: this requirement only implies support for the simple redirection convention defined by that profile and does demand implementation of an actual discovery interface, though that is not precluded. Also note that the discovery mechanism should use SAML metadata to determine the endpoints to which requests are to be issued.

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 as described in 2.7 Authentication Context Class Reference.

Service Providers **MAY** support the acceptance or rejection of assertion based on the content of the `<saml:AuthnContext>` element.

Service Providers **MAY** support decryption of `<saml:EncryptedAssertion>` elements. To fully support key rollover, Service Providers **MUST** be configurable with at least two decryption keys.

When decrypting assertions, an attempt to use each decryption key **MUST** be made until the assertion is successfully decrypted or there are no more keys whereupon the decryption fails.

Service providers **MUST** support deep linking and maintain the direct accessibility of protected resources in the presence of Web Browser SSO. It **MUST** be possible to request an arbitrary protected resource and where the authorization permits, have it supplied as the result of a successful SAML SSO profile exchange. Service Providers **SHOULD** support the preservation of POST bodies across a successful SSO profile exchange, subject to size limitations dictated by policy or implementation constraints.

Note: the SAML binding-specific `RelayState` feature is typically used to maintain state required to satisfy both of these requirements. The exact details are left to implementations.

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

3.1.1.1 Avoiding Common Errors

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

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

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

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

Note: 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 Identity Exchange Profile (IdP)

3.2.1 Web Browser SSO

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.

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

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]**.

Identity Providers **MUST** be capable of determining whether or not to include specific SAML attributes or values in a response based on the presence of `<md:AttributeConsumingService>` elements (containing

<md:RequestedAttribute> elements) found in metadata for a relying party, including the value of the enclosed isRequired XML attribute. The Identity Provider **MUST** support the AttributeConsumingServiceIndex attribute in <samlp:AuthnRequest> messages as a means of determining the appropriate <md:AttributeConsumingService> element to process.

Note: <md:RequestedAttribute> elements in metadata can be used to help automate attribute release configurations in IdP deployments. An IdP could be configured to release attributes 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 for the relying party. Attributes **MUST** only be released in accordance with the Attribute Sharing Policies set out in **[TDIF.Attributes]**.

Identity Providers **MUST** support the issuance of <samlp:Response> messages with the appropriate status code in the event of an error condition, provided 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 possible malicious parties.

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

Note: ForceAuthn is most commonly used for privilege escalation or to initiate explicit user approval for an action.

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

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

Identity providers **SHOULD** support encryption of assertions. Support for encryption of identifiers and attributes is OPTIONAL.

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

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 Identity Exchange (SP) to Identity Provider (IdP) Profile

4.1 Identity Exchange Profile

In this section all references to Service Provider or SP refer to the Identity Exchange and any references to Identity Provider or IdP refer to the Identity Provider.

4.1.1 Web Browser SSO

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.

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.

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

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

Note: this requirement only implies support for the simple redirection convention defined by that profile and does demand implementation of an actual discovery interface, though that is not precluded. Also note that the discovery mechanism should use SAML metadata to determine the endpoints to which requests are to be issued.

Service providers **MUST** support the processing of responses from any number of issuing IdPs for any given resource URL. It **MUST NOT** be a restriction of an implementation that multiple IdPs can only be supported by requiring distinct resource

URLs for each IdP. The ability to satisfy this requirement should come naturally from the ability to support **[IdPDisco]**.

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.

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

Service Providers **MUST** support decryption of `<saml:EncryptedAssertion>` elements. To fully support key rollover, Service Providers **MUST** be configurable with at least two decryption keys. When decrypting assertions, an attempt to use each decryption key **MUST** be made until the assertion is successfully decrypted or there are no more keys whereupon the decryption fails.

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

4.1.1.1 Avoiding Common Errors

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

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

Service Providers **MUST NOT** require that the 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]** section 8.3.7.

Note: that if the name identifier format identifiers defined is **[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.

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

Note: 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.

4.2 Identity Provider Profile (IdP)

In this section all references to Service Provider or SP refer to the Identity Exchange and any references to Identity Provider or IdP refer to the Identity Provider.

4.2.1 Web Browser SSO

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.

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

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]**.

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

Note: `<md:RequestedAttribute>` elements in metadata can be used to help automate attribute release configurations in IdP deployments. An IdP could be configured to release attributes 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 for the relying party.

Identity Providers **MUST** support the issuance of `<samlp:Response>` messages with the appropriate status code in the event of an error condition, provided 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 form being sent to untrusted or possible malicious parties.

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

Note: `ForceAuthn` is most commonly used for privilege escalation or to initiate explicit user approval for an action.

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

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

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

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

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

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]**.

5 Acknowledgements

The authors of this document acknowledge the work of the Kantara Initiative, see https://kantarainitiative.org/, a non-profit based organisation passionate about giving control of data back to people. The Kantara Initiative Inc. provides real-world innovation and specifications and conformity assessment programs for the digital identity and personal data ecosystems.

To maximise the interoperability of this profile, applicable elements from the SAML V2.0 Implementation Profile for Federation Interoperability Version 1.0 (working draft) has been used. The Kantara Federation Interoperability Work Group is developing this profile, the home page for this working can be found at https://kantarainitiative.org/confluence/display/fiwg/Home.

Annex A – Interactions

The following sequence diagrams show the logical sequence of interactions for the authentication of a user.

***Figure 1** – User Authentication Sequence Diagrams (steps 1 to 5)*

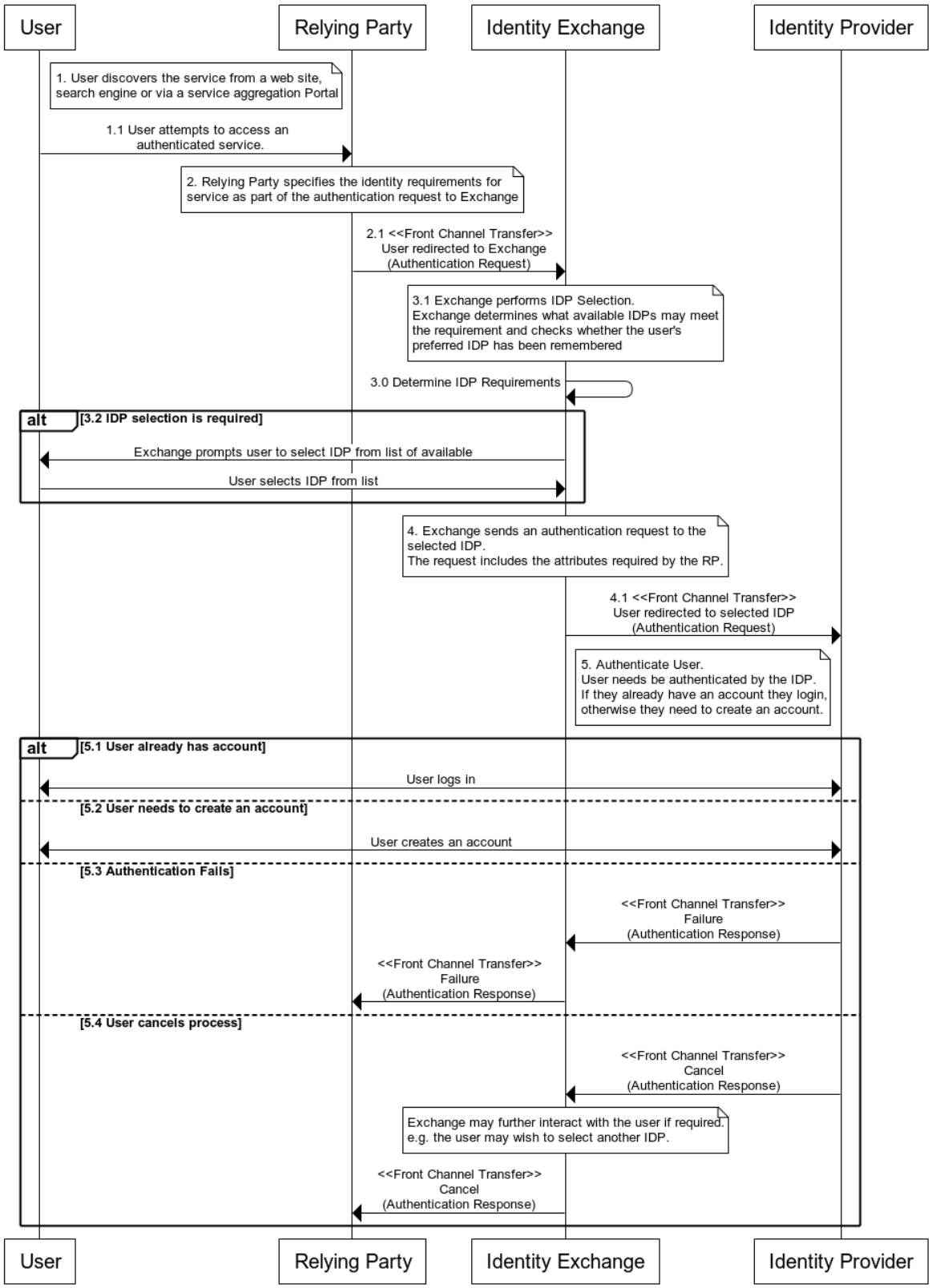

***Figure 2** – User Authentication (steps 6 to 11)*

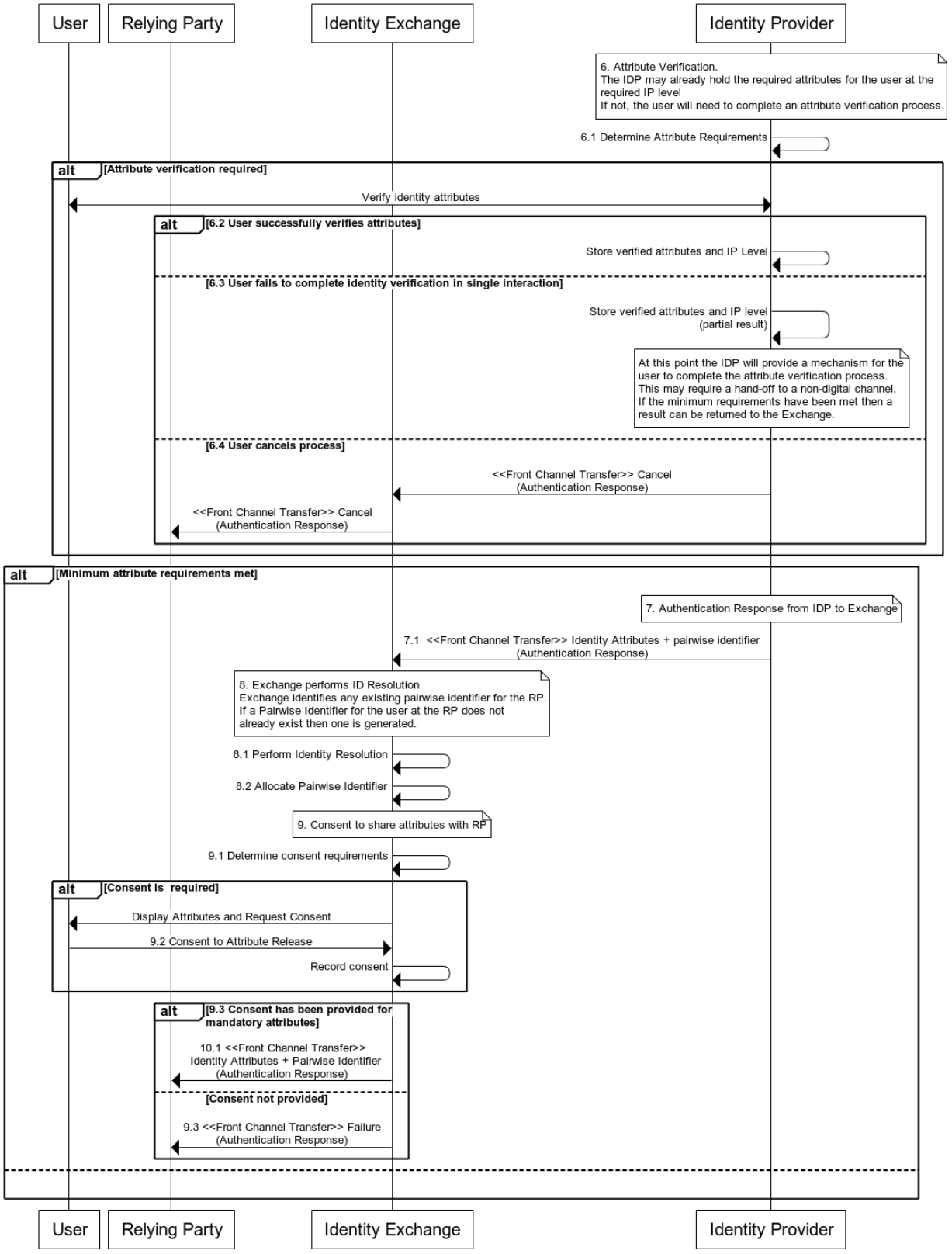

Figure 1 and 2 are sequence diagrams show the sequence of logical interactions for the authentication of a user. These interactions are intended to illustrate the

application of the SAML profiles described in this document to an end-to-end user experience. Where is the user is transferred between entities via the user agent, e.g. web browser, the interaction is annotated with the `<<Front Channel Transfer>>` label. Each step in the diagram is described in detail below. Each step in the diagram is described in detail below.

1. User discovers the digital service.

    1.1. User attempts to access an authenticated digital service.

        - The user discovers the digital service at a Relying Party. This may be from content on unauthenticated web site, a search engine, or from within a service aggregation portal.
        - The user accessing the service triggers the authentication process and verification of identity attributes may occur as part of this authentication process.
        - A user may initiate the attribute verification process independently of accessing a service by going directly to an Identity Provider.

2. Authentication Request from Relying Party to Exchange.

    2.1. User redirected to Exchange by the Relying Party using an authentication request.

        - The Relying Party specifies the identity requirements for the digital service as part of the authentication request. The request includes the required TDIF Assurance Levels and required identity attributes.
        - The Relying Party specifies the minimum assurance level that is required. The minimum assurance level may be specified as mandatory. If the specified minimum IP level is mandatory it must be reached for a successful authentication response to be returned to the Relying Party.
        - The identity attributes may be specified as optional or mandatory. If a mandatory attribute cannot be returned (not available or consent not provided) then the authentication response will be a failure.

Step 2 uses the Relying Party to Identity Exchange Profile:

- A SAML Authentication Request from Relying Party is sent to the Identity Exchange. The Request includes:

- required identity attributes either as extensions in the request or pre-determined via the Relying Party's metadata.
- `<AuthnContextClassRef>` values that specify the required assurance levels.

3. Identity Provider Selection.

   3.1. The Identity Exchange determines the Identity Providers that will meet the requirements of the Authentication Request from the Relying Party. The Identity Exchange will determine what Identity Providers are available to meet the request. It will also check when a preferred Identity Provider for the user has been remembered.

   3.2. If more than one Identity Provider is available then the user will be prompted to select an Identity Provider from a list. This selection may be remembered to streamline further interactions.

   3.3. User Cancels Process. An Authentication Response indicating the cancellation of the process is sent back to the Relying Party.

4. Authentication Request from Identity Exchange to Identity Provider.

   4.1. Exchange redirects the user to the selected Identity Provider using an authentication request. The request includes the attributes and assurance levels that were originally requested by the Relying Party.

Step 4 uses the Identity Exchange to Identity Provider Profile.

- A SAML Authentication Request is sent to the Identity Provider. The request includes:
- the attributes that are required to service the request Relying Party request either as extensions in the authentication request or predetermined via the Identity Exchanges metadata.
- the set of `<AuthnContextClassRef>` values that meet or exceed the `<AuthnContextClassRef>` requested by the Relying Party.

5. Authenticate User. The user will either login to an existing account at the Identity Provider or create a new one.

5.1. User already has an account at the Identity Provider.

- The user logs into the Identity Provider using their existing credentials. If the existing credentials do not meet the required credential level the user will need to enrol additional credentials.

5.2. User does not have an account at the Identity Provider.

    - The user creates an account and is issued with credentials at the required credential level.

5.3. Authentication Fails.

    - If the user fails to authenticate at the required credential level then an Authentication Response indicating the authentication failure is sent back to the Identity Exchange. The Identity Exchange then sends the same Authentication Response back to the Relying Party.

5.4. User Cancels Process.

    - An Authentication Response indicating the cancellation of the process is sent back to the Exchange. The Exchange may interact with the user to determine if an alternate pathway is required to complete the process, e.g. to select a different Identity Provider. The Identity Exchange then sends the same Authentication Response back to the Relying Party if there is no identified alternate pathway.

Step 5 uses the Identity Exchange to Identity Provider Profile.

- Authentication Fails: IDP responds with the defined error status corresponding to the failure.
- User Cancels Process: IDP responds with the defined error status that the error was on the part of the responder with the following second-level status code: `urn:id.gov.au:tdif:SAML:2.0.status.AuthnCancelled`.

Step 5 then continues using the Relying Party to Identity Exchange Profile.

- Authentication Fails: Exchange responds with the error status that was received from the IdP.

- User Cancels Process: IDP responds with error code value that was received from the IdP.

6. Verify Attributes. The Identity Provider may already hold the attributes at the required IP level for the user. If not, an interaction with user is required to verify attributes at the required level.

   6.1. Identity Provider determines attribute requirements.

      - The Identity Provider checks the attributes already held for the user and determine if any further attribute verification is required. If attribute verification is required then steps 6.2 to 6.4 are possible paths.

   6.2. User successfully verifies attributes.

      - The user is able to successfully verify attributes at the required level.

   6.3. The user is unable to complete the attribute verification process to the desired IP level in a single digital interaction.

      - The Identity Provider will store the partial result and provide a process for the user to complete the attribute verification. This may require a hand-off to a non-digital channel. If the Relying Party originally specified a minimum IP level that has been met then a response can be returned to the Relying Party, otherwise this sequence of interactions end here.

   6.4. User Cancels Process.

      - An Authentication Response indicating the cancellation of the process is sent back to the Exchange. The Identity Exchange then sends the same Authentication Response back to the Relying Party if there is no identified alternate pathway.

Step 6 uses the Identity Exchange to Identity Provider Profile.

- User Cancels Process: IDP responds with the defined error status that the error was on the part of the responder with the following second-level status code: `urn:id.gov.au:tdif:SAML:2.0.status.AuthnCancelled`.

- Authentication Response to Exchange: If the minimum attribute requirements are met then a successful authentication response is sent back to the Exchange.

7. Authentication Response is sent back to the Identity Exchange.

   7.1. The Authentication Response from the Identity Provider includes:

   - Achieved `<AuthnContextClassRef>` level.
   - A pairwise identifier for the user at the Identity Provider.
   - Identity attributes.

Step 7 uses the Identity Exchange to Identity Provider Profile.

- A SAML Response is returned to the Identity Exchange containing the identity attributes defined from either the request or from the Identity Exchanges metadata within an Attribute Statement and the achieved. `<AuthnContextClassRef>` level. The Response is signed with the Identity Provider's Private key.

8. Exchange performs Identity Resolution.

- Identity Exchange identifies any existing pairwise identifier user at the Relying Party. If a pairwise Identifier for the user at the Relying Party does not already exist then one is generated.

   8.1. Perform Identity Resolution.

   - If a pairwise identifier is already mapped to the pairwise identifier from the Identity Provider then the Identity Exchange will use the pairwise identifier that is already allocated for the user.

   8.2. Allocate Pairwise Identifier.

   - If required, a pairwise identifier is generated for the user. A pairwise identifier is an anonymous, unique identifier for the user at the Relying Party.

9. Consent to share attributes.

   9.1. Determine consent requirements.

- Identity Exchange determines the user consent requirements for the attributes requested by the Relying Party. It will include checking for any enduring consent for sharing the attributes with the Relying Party.

9.2. Consent to Attribute Release.

- If user consent is required, the Identity Exchange will interact with the user to gather consent to release the attributes to the Relying Party. The Identity Exchange will record the provided consent and the user's preference for enduring this consent.

9.3. Consent not provided.

- If user consent is not provided for any mandatory attribute then a failure Authentication Response is returned to the Relying Party.

Step 9.3 uses the Relying Party to Identity Exchange Profile:

- Consent not provided (mandatory attribute): Exchange responds with the defined error status that the error was on the part of the requestor.

10. Authentication Response to Relying Party.

10.1. Authentication Response is sent back to the Relying Party.

- The Response includes:
  - Achieved `<AuthnContextClassRef>` level.
  - Pairwise identifier for user at the Relying Party.
  - Identity attributes for which consent has been provided.

Step 10 uses the Relying Party to Identity Exchange Profile.

- A SAML Response is returned to the Relying Party containing the identity attributes defined from either the request or from the Identity Exchanges metadata within an Attribute Statement and the achieved `<AuthnContextClassRef>` level. The Response is signed with the Identity Provider's Private key.

11. User accesses digital service.

11.1. Relying Party uses the identity attributes to enable the user to access the digital service.

- The first time the user accesses the first the Relying Party may need to determine if there is an existing customer record by using the identity attributes as part of an Identity Matching process. Once a customer record has been located or created at the Relying Party the Pairwise identifier is stored by the Relying Party, subsequent interaction by the user with the digital service will simply use the pairwise identifier to locate the customer record.
- **Note**: some transactions may be one-off and not require the above process.

Annex B – Worked Examples

The following example shows successful authentication interactions between a Relying Party (RP) and an Identity Provider (IdP) via an Identity Exchange using the Web SSO Profile using the HTTP-POST binding.

HTTP POST Binding

1. The End User requests access to a protected resource without a security context. The Relying Party determines the location of an endpoint for an Identity Exchange.

2. The Relying Party sends an HTML form back to the browser with a SAML request for authentication from the Identity Exchange.

3. The form is automatically posted to the Identity Exchange's SSO service.

4. The authentication request includes the name of the Relying Party requesting the authentication (ProviderName) and the <samlp:AuthenticationContextClassRef> to specify the required level of assurance.

   The following is a non-normative example of the SAML Authentication request (with line wraps with shortened cryptographic element values and line wraps for readability).

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
       ID="pfx41d8ef22-e612-8c50-9960-1b16f15741b3"
       Version="2.0"
       ProviderName="rp.example.com"
       IssueInstant="2014-07-16T23:52:45Z"
       Destination=http://idexchange.gov.au/SSOService
       ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
       AssertionConsumerServiceURL="http://client.example.org/someapp/acs">
  <saml:Issuer>http://client.example.org/someapp/metadata</saml:Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha256"/>
      <ds:Reference URI="#pfx41d8ef22-e612-8c50-9960-1b16f15741b3">
        <ds:Transforms>
          <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
```

```
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod
rAlgorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>
        <ds:DigestValue>yJN6cXUwQxTmME…sPesBP2NkqYFI=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue> g5eM9yPnKsmmE ….
rqngwTJk5KmujbqouR1SLFsbo7Iuwze933EgefBbAE4JRI7
V2aD9YgmB3socPqAi2Qf97E=
    </ds:SignatureValue>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>
MIICajCCAdOgAwIBAgIBADANBgkqhkiG9w0BAQQFADBSMQswCQYDVQQGEwJ1czETMBEGA1UECA
……
AGiFomHop1nErV6Q==
        </ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </ds:Signature>
  <samlp:NameIDPolicy Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent" AllowCreate="true"/>
  <samlp:RequestedAuthnContext Comparison="exact">

<saml:AuthnContextClassRef>urn:id.gov.au:tdif:acr:ip3:cl2</saml:AuthnContext
ClassRef>
  </samlp:RequestedAuthnContext>
</samlp:AuthnRequest>
```

5. The Identity Exchange validates the Authentication Request from the Relying Party.

6. The identity Exchange prompts the End User to select an Identity Provider (account). The Identity Exchange may provide a mechanism to remember a previous Identity Provider selection made by the End User.

7. The Identity Exchange then constructs a SAML Request for Authentication from the selected Identity Provider and sends it back to the End User's browser.

8. This request is automatically posted to the Identity Providers SSO service.

9. The authentication request includes the name of the Identity Exchange as the name of the Relying Party requesting the authentication and the same <`samlp:AuthenticationContextClassRef`> value that was requested in the authentication request from the relying party.

The following is a non-normative example of the SAML Authentication request (with line wraps with shortened cryptographic element values and line wraps for readability).

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
       ID="pfx41d8ef22-e612-8c50-9960-1b16f15741b3"
       Version="2.0"
       ProviderName="rp.example.com"
       IssueInstant="2014-07-16T23:52:45Z"
       Destination=http://idexchange.gov.au/SSOService
       ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
       AssertionConsumerServiceURL="http://client.example.org/someapp/acs">
  <saml:Issuer>http://client.example.org/someapp/metadata</saml:Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha256"/>
      <ds:Reference URI="#pfx41d8ef22-e612-8c50-9960-1b16f15741b3">
        <ds:Transforms>
          <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>
        <ds:DigestValue>yJN6cXUwQxTmME…sPesBP2NkqYFI=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
g5eM9yPnKsmmE/Kh2qS7nfK8HoF6yHrAdNQxh70kh8pRI4KaNbYNOL9sF8F57Yd+jO6iNga8nn
…
V2aD9YgmB3socPqAi2Qf97E=
    </ds:SignatureValue>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>
MIICajCCAdOgAwIBAgIBADANBgkqhkiG9w0BAQQFADBSMQswCQYDVQQGEwJ1czETMBEGA1UECA
…
AGiFomHop1nErV6Q==
        </ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </ds:Signature>
  <samlp:NameIDPolicy Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent" AllowCreate="true"/>
  <samlp:RequestedAuthnContext Comparison="exact">

<saml:AuthnContextClassRef>urn:id.gov.au:tdif:acr:ip3:cl2</saml:AuthnContext
ClassRef>
  </samlp:RequestedAuthnContext>
</samlp:AuthnRequest>
```

10. The Identity Provider validates the request.

11. The Identity Provider verifies if the End User is logged in and logs them if they aren't ensuring that any additional requirements are met to meet the requirements of the `<RequestedAuthnContext>`. If the End User was already logged in and their existing Authentication Context is lower that the `<RequestedAuthnContext>` they will be required to perform any additional interaction required to meet the requested `<RequestedAuthnContext>`.

12. After the successful authentication the Identity Provider builds a SAML Response including the retrieval of any attributes that have been predetermined to be required as part of the federation agreement between the Identity Exchange and the Identity Provider and the achieved `<saml:authnContextClassRef>`.

13. The SAML Response is send back to the End User's browser and automatically posted to the Identity Exchange's Assertion Consumer Service (ACS). Note that the response is signed as per the requirements of SAML for POST responses.

    The following is a non-normative example of the SAML Response (with line wraps with shortened cryptographic element values and line wraps for readability).

```
<?xml version="1.0"?>
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
               xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
               ID="pfxfd645c87-2a49-043e-6a76-68f837470374" Version="2.0"
               IssueInstant="2014-07-17T01:01:48Z"
               Destination=http://sp.idexchange.gov.au/cb/acs
               InResponseTo="IDP_4fee3b046395c4e751011e97f8900b5273d56685">
  <saml:Issuer>http://idp.gov.au/metadata</saml:Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha256"/>
      <ds:Reference URI="#pfxfd645c87-2a49-043e-6a76-68f837470374">
        <ds:Transforms>
          <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>
        <ds:DigestValue>VzmNr+Qm8FjOklx…jAsQw6yrzh9w=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
0g325sNOWoiB6+AymJwyRQKpjpy5xzLkMlEiAsr/CyolvhtjK5G7llRO5gh53CR3dGT0YBIdvm
…
Vcd8no4L6jMlrUlH0DXB9yY=
```

```
    </ds:SignatureValue>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>
MIICajCCAdOgAwIBAgIBADANBgkqhkiG9w0BAQ0FADBSMQswCQYDVQQGEwJ1czETMBEGA1UECA
…
4LzgD0CROMASTWNg==
        </ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </ds:Signature>
  <samlp:Status>
    <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </samlp:Status>
  <saml:Assertion xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  ID="pfx8f564359-2bce-a367-99d0-12c8601bada9"
                  Version="2.0" IssueInstant="2014-07-17T01:01:48Z">
    <saml:Issuer>http://idp.gov.au/metadata</saml:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
        <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha256"/>
        <ds:Reference URI="#pfx8f564359-2bce-a367-99d0-12c8601bada9">
          <ds:Transforms>
            <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
          </ds:Transforms>
          <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>
          <ds:DigestValue>/ExPJp1cibZFT/…sLI8BUNlgz1pI=</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>
lXrbu/LvXPm08ZOj/lFfKoS8mlZoiwINWab6AM5mv7LnSWb8IF/5cnijXN2k2C5xgGnM49WMbs
…
apb440F7I1AAhnEcTTIVKFw=
      </ds:SignatureValue>
      <ds:KeyInfo>
        <ds:X509Data>
          <ds:X509Certificate>
MIICajCCAdOgAwIBAgIBADANBgkqhkiG9w0BAQ0FADBSMQswCQYDVQQGEwJ1czETMBEGA1UECA
…
4LzgD0CROMASTWNg==
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </ds:Signature>
    <saml:Subject>
      <saml:NameID
            SPNameQualifier="http://sp.idexchange.gov.au/cb/metadata"
            Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">
         _ce3d2948b4cf20146dee0a0b3dd6f69b6cf86f62d7
      </saml:NameID>
      <saml:SubjectConfirmation
                  Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
```

```
        <saml:SubjectConfirmationData NotOnOrAfter="2024-01-18T06:21:48Z"
            Recipient=http://sp.idexchange.gov.au/cb/acs
            InResponseTo="IDP_4fee3b046395c4e751011e97f8900b5273d56685"/>
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Conditions NotBefore="2014-07-17T01:01:18Z"
                     NotOnOrAfter="2024-01-18T06:21:48Z">
      <saml:AudienceRestriction>
        <saml:Audience>
          http://sp.idexchange.gov.au/cb/metadata.php
        </saml:Audience>
      </saml:AudienceRestriction>
    </saml:Conditions>
    <saml:AuthnStatement AuthnInstant="2014-07-17T01:01:48Z"
                         SessionNotOnOrAfter="2024-07-17T09:01:48Z"
              SessionIndex="_be9967abd904ddcae3c0eb4189adbe3f71e327cf93">
      <saml:AuthnContext>
        <saml:AuthnContextClassRef>
            urn:id.gov.au:tdif:acr:ip3:cl2
        </saml:AuthnContextClassRef>
      </saml:AuthnContext>
    </saml:AuthnStatement>
    <saml:AttributeStatement>
      <saml:Attribute Name="family_name"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">
                Michaels
        </saml:AttributeValue>
      </saml:Attribute>
      <saml:Attribute Name="given_name"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">
                Stephen
        </saml:AttributeValue>
      </saml:Attribute>
      <saml:Attribute Name="birthdate"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">
                1974-02-29
        </saml:AttributeValue>
      </saml:Attribute>
    </saml:AttributeStatement>
  </saml:Assertion>
</samlp:Response>
```

14. The ACS of the Identity Exchange validates the signature and the assertions contained in the response message.

15. The Identity Exchange then proxies the received information and builds a SAML Response containing the attributes required by the Relying Party that have been predetermined as part of the federation agreement between the Relying Party and the Identity Exchange and the achieved `<saml:authnContextClassRef>`.

16. The Response object is sent back the End User's Browser where it is automatically posted to the Relying Party's ACS.

    The following is a non-normative example of the SAML Response (with line wraps with shortened cryptographic element values and line wraps for readability).

```
<?xml version="1.0"?>
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                ID="pfxfd645c87-2a49-043e-6a76-68f837470374"
                Version="2.0"
                IssueInstant="2014-07-17T01:01:48Z"
                Destination="http://client.example.org/someapp/acs"
              InResponseTo="IDEX_4fee3b046395c4e751011e97f8900b5273d56685">
  <saml:Issuer>http://idexchange.gov.au/metadata</saml:Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
        <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha256"/>
        <ds:Reference URI="#pfxfd645c87-2a49-043e-6a76-68f837470374">
            <ds:Transforms>
                <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>
            <ds:DigestValue>VzmNr+Qm8FjOk…lxjAsQw6yrzh9w=</ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
0g325sNOWoiB6+AymJwyRQKpjpy5xzLkMlEiAsr/CyolvhtjK5G7llRO5gh53CR3dGT0YBIdvm
…
Vcd8no4L6jMlrUlH0DXB9yY=
    </ds:SignatureValue>
    <ds:KeyInfo>
        <ds:X509Data>
            <ds:X509Certificate>
MIICajCCAdOgAwIBAgIBADANBgkqhkiG9w0BAQ0FADBSMQswCQYDVQQGEwJ1czETMBEGA1UECA
…
4LzgD0CROMASTWNg==
            </ds:X509Certificate>
        </ds:X509Data>
    </ds:KeyInfo>
  </ds:Signature>
  <samlp:Status>
    <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </samlp:Status>
  <saml:Assertion xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  ID="pfx8f564359-2bce-a367-99d0-12c8601bada9"
                  Version="2.0" IssueInstant="2014-07-17T01:01:48Z">
    <saml:Issuer>http://idexchange.gov.au/metadata</saml:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
```

```
        <ds:SignedInfo>
            <ds:CanonicalizationMethod
               Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod
               Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha256"/>
            <ds:Reference URI="#pfx8f564359-2bce-a367-99d0-12c8601bada9">
                <ds:Transforms>
                    <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
                </ds:Transforms>
                <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>
                 <ds:DigestValue>/ExPJp1cibZFT/…sLI8BUNlgz1pI=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
lXrbu/LvXPm08ZOj/lFfKoS8mlZoiwINWab6AM5mv7LnSWb8IF/5cnijXN2k2C5xgGnM49WMbs
…
apb440F7I1AAhnEcTTIVKFw=
        </ds:SignatureValue>
    <ds:KeyInfo>
        <ds:X509Data>
            <ds:X509Certificate>
MIICajCCAdOgAwIBAgIBADANBgkqhkiG9w0BAQ0FADBSMQswCQYDVQQGEwJ1czETMBEGA1UECA
…
4LzgD0CROMASTWNg==
            </ds:X509Certificate>
        </ds:X509Data>
    </ds:KeyInfo>
   </ds:Signature>
   <saml:Subject>
      <saml:NameID
            SPNameQualifier="http://sp.example.com/demo1/metadata"
            Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">
          _ce3d2948b4cf20146dee0a0b3dd6f69b6cf86f62d7
      </saml:NameID>
      <saml:SubjectConfirmation
           Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <saml:SubjectConfirmationData NotOnOrAfter="2014-01-18T06:21:48Z"
             Recipient="http://client.example.org/someapp/acs"
             InResponseTo="IDEX_4fee3b046395c4e751011e97f8900b5273d56685"/>
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Conditions NotBefore="2014-07-17T01:01:18Z"
                     NotOnOrAfter="2014-01-18T06:21:48Z">
      <saml:AudienceRestriction>
        <saml:Audience>
             http://client.example.org/someapp/metadata
        </saml:Audience>
      </saml:AudienceRestriction>
    </saml:Conditions>
    <saml:AuthnStatement AuthnInstant="2014-07-17T01:01:48Z"
                         SessionNotOnOrAfter="2014-07-17T09:01:48Z"
                SessionIndex="_be9967abd904ddcae3c0eb4189adbe3f71e327cf93">
      <saml:AuthnContext>
        <saml:AuthnContextClassRef>
             urn:id.gov.au:tdif:acr:ip3:cl2
        </saml:AuthnContextClassRef>
```

```
      </saml:AuthnContext>
    </saml:AuthnStatement>
    <saml:AttributeStatement>
      <saml:Attribute Name="family_name"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">
              Michaels
        </saml:AttributeValue>
      </saml:Attribute>
      <saml:Attribute Name="given_name"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">
              Stephen
        </saml:AttributeValue>
      </saml:Attribute>
      <saml:Attribute Name="birthdate"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">
              1974-02-29
        </saml:AttributeValue>
      </saml:Attribute>
    </saml:AttributeStatement>
  </saml:Assertion>
</samlp:Response>
```

17. The Relying Party then validates the signatures, the required attributes and the achieved `<saml:authnContextClassRef>`. If all these are valid the Relying Party establishes a session for the End User and redirects their browser to the originally targeted resource.