SAML 2.0 Profile

Trusted Digital Identity Framework
August 2018, version 1.0
Digital Transformation Agency

This work is copyright. Apart from any use as permitted under the Copyright Act 1968 and the rights explicitly granted below, all rights are reserved.

Licence

With the exception of the Commonwealth Coat of Arms and where otherwise noted, this product is provided under a Creative Commons Attribution 4.0 International Licence. (http://creativecommons.org/licenses/by/4.0/legalcode)

This licence lets you distribute, remix, tweak and build upon this work, even commercially, as long as they credit the DTA for the original creation. Except where otherwise noted, any reference to, reuse or distribution of part or all of this work must include the following attribution:

Trusted Digital Identity Framework: SAML 2.0 Profile © Commonwealth of Australia (Digital Transformation Agency) 2018

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

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 Trusted Digital Identity Framework: Overview and Glossary.

Contact us

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

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

Change log

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Description of the changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>Jan 18</td>
<td>BF</td>
<td>Initial version</td>
</tr>
<tr>
<td>0.02</td>
<td>Aug 18</td>
<td>TM</td>
<td>Updates from stakeholder consultation.</td>
</tr>
<tr>
<td>1.0</td>
<td>Aug 18</td>
<td></td>
<td>Endorsed for release by the TFAA</td>
</tr>
</tbody>
</table>
## Contents

1 Introduction ...................................................................................................................... 1  
2 Technical Standards Overview ......................................................................................... 2  
3 The Identity Exchange .................................................................................................... 5  
   3.1 Trust Broker ................................................................................................................ 5  
      3.1.1 Trust Broker Protocol Mappings ................................................................. 7  
      3.1.2 Pairwise Identifiers and Identity Resolution ............................................... 7  
      3.1.3 Assurance Levels ......................................................................................... 8  
      3.1.4 SAML to SAML Brokering ........................................................................... 9  
      3.1.5 SAML to OIDC Brokering .......................................................................... 10  
   3.2 Attribute Sharing .................................................................................................... 12  
4 Common Profile Requirements ....................................................................................... 13  
   4.1 Notation ................................................................................................................... 13  
   4.2 General ................................................................................................................... 13  
      4.2.1 Clock skew ...................................................................................................... 13  
      4.2.2 Data Size ......................................................................................................... 14  
      4.2.3 Document Type Definitions ......................................................................... 14  
   4.3 Metadata and Trust Management ............................................................................ 14  
      4.3.1 Metadata Exchange .................................................................................... 15  
      4.3.2 Metadata Usage ......................................................................................... 16  
   4.4 Web Browser SSO ................................................................................................. 18  
   4.5 Extensibility ........................................................................................................... 19  
   4.6 Cryptographic Algorithms ...................................................................................... 19  
   4.7 Authentication Context Class Reference ............................................................ 21  
   4.8 Attributes ............................................................................................................... 22  
5 Relying Party (SP) to Identity Exchange (IdP) Profile .................................................... 24  
   5.1 Relying Party Profile (SP) ...................................................................................... 24  
      5.1.1 Web Browser SSO ....................................................................................... 24
1 Introduction

The Digital Transformation Agency (DTA), in collaboration with other government agencies and key private sector bodies, is leading the development of a national federated identity ‘eco-system’ (the ‘identity federation’). Implementation and operation of the identity federation is underpinned by the Trusted Digital Identity Framework (TDIF). This document should be read in conjunction with the TDIF: Overview and Glossary, which provides a high-level overview of the TDIF including its scope and objectives and the definition of key terms.

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.
2 Technical Standards Overview

The TDIF enables the implementation of a national identity federation. This identity federation can support multiple Identity Exchanges that connect Relying Parties to Identity Providers using established and standardised federation protocols in an interoperable fashion. The currently supported federation protocols are OpenID Connect 1.0 (OIDC) and SAML 2.0 (SAML).

The following table notes the correspondence between the terminology used in the TDIF and the terms used to describe entities in the federation protocols.

<table>
<thead>
<tr>
<th>Trust Framework Term</th>
<th>OIDC Term</th>
<th>SAML Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relying Party (RP)</td>
<td>Relying Party (RP)</td>
<td>Service Provider (SP)</td>
</tr>
<tr>
<td>Identity Provider (IdP)</td>
<td>OpenID Provider (OP)</td>
<td>Identity Provider (IdP)</td>
</tr>
</tbody>
</table>

Figure 1 – Identity Federation Topologies
Figure 1 illustrates the possible identity federation topologies that may exist in a mature identity eco-system. Digital services implemented that are relying on the identity federation can establish connections to any number of available Identity Exchanges that support their required federation protocol. These Identity Exchanges in turn can connect to any number of Identity Providers using their supported federation protocols.

The TDIF requires the presence of the Identity Exchange as a trusted broker between Relying Parties and Identity Providers. The Identity Exchange acts as an IdP Proxy from the perspective of a Relying Party. The Identity Exchange must proxy the original request from a Relying Party to the user’s selected identity Provider. Hence there are two hops in the interaction between a Relying Party and an Identity Provider via the Identity Exchange. Each of these hops is an instantiation of the federation protocols, with the Identity Exchange being responsible for maintaining the correspondence between the hops. In each of the hops the Identity Exchange acts as a different entity in terms of the federation protocols:

- Identity Exchange to Identity Provider: The Identity Exchange acts as a Relying Party.

With more advanced Identity Exchanges this process may also include a translation in the federation protocol used. For example, a Relying Party connecting to an Identity Exchange using the SAML 2.0 federation may have their requests serviced by the Identity Exchange performing a protocol translation to provide authentication from an Identity Provider that uses the OpenID Connect 1.0 federation protocol.

As well as providing the ability to proxy requests from a Relying Party to an Identity Provider chosen by the user, The Identity Exchange, as a trusted intermediary, enforces the attribute sharing policies required by the TDIF, such as any requirement for user consent.

A key characteristic of the presence of the Identity Exchange as a trusted broker is that it enables a double-blind federation. In a double-blind federation the Identity Provider does not know which service is requesting the authentication and or attributes of an identity and ultimately the Relying Party does not know which Identity
Provider has provided the credential and the attributes. This double-blind implementation is a key privacy requirement of the TDIF.

Relying Parties operating as part of the federation **MUST** implement:

- TDIF OpenID Connect 1.0 Profile as a OIDC Relying Party (RP);
  OR
- TDIF SAML 2.0 Profile as a SAML Service Provider (SP)

OIDC is the preferred protocol for new implementations.

Identity Providers operating as part of the identity federation **MUST** support:

- TDIF OpenID Connect 1.0 Profile as an OIDC Provider (OP); or
- TDIF SAML 2.0 Profile as an Identity Provider (IdP).

OIDC is the preferred protocol for Identity Providers.

Identity Exchanges operating as part of the identity federation **MUST** support:

- TDIF OpenID Connect 1.0 Profile as an OIDC Provider (OP) and TDIF OpenID Connect 1.0 Profile as a Relying Party (RP);

Identity Exchanges operating as part of the identity federation **MAY** support:

- TDIF SAML 2.0 Profile as a SAML IdP and TDIF SAML Profile 2.0 as a SAML(SP).

The need for an Identity Exchange SAML will be driven by the needs of the Relying Parties that are connecting to, and by the presence of any Identity Provider that requires the use of SAML 2.0. Accreditation of an Identity Exchange is not predicated on the up-front need to support SAML 2.0.

Identity Exchanges operating as part of the federation **MUST** broker authentication requests from a Relying Party to an Identity Provider. The response from an Identity Provider to an Identity Provider **MUST** use a pairwise identifier to identify the subject of the authentication. In turn, the response from the Identity Exchange to Relying Party must use different pairwise identifier to identify the subject of the authentication to the Relying Party. This use of pairwise identifiers is a key requirement to implement a double-blind federation.
3 The Identity Exchange

The Identity Exchange is the core component of the identity federation and acts as a trust broker between the Relying Parties and the Identity Providers.

The Identity Exchange allows the federation to be double blind federation by ensuring that the Identity Provider does not know which Relying Party is requesting the authentication and attributes and the Relying Party does not know which Identity Provider performed the authentication and provided the attributes.

3.1 Trust Broker

Figure 2 illustrates the interactions that an Identity Exchange **MUST** implement in order perform the function of a trust broker. These are the logical interactions that are implemented using the federation protocols, the implementation of the logical interactions may involve multiple protocol steps.

**Figure 2 – Trust Broker Interactions**

The Trust Broker interactions are:

1. The Relying Party generates an authentication requestion and sends request to the Identity Exchange.
2. The Identity Exchange:
   a. validates the request from the Identity Provider
   b. generates a unique audit Id for the request and logs the authentication request from the Relying Party including the audit Id
   c. performs any processing required to determine the Identity Provider to send the request to
3. The Identity Exchange:
a. maps the elements from the request from the Relying Party to the required elements to request from an Identity Provider and generates the authentication request to send to the Identity Provider
b. logs the generated authentication request using the audit Id generated for the request from the Relying Party in step 2
c. sends the generated authentication request to the Identity Provider

4. The Identity Provider:
   a. validates the request form the Identity Exchange
   b. logs the request from the Identity Exchange
   c. performs any additional processing to service the request.
d. generates the authentication response to the Identity Exchange
e. logs the authentication response to the Identity Exchange

5. The Identity Provider sends the authentication response to the Identity Exchange

6. The Identity Exchange:
   a. validates the response from the Identity Provider
   b. logs the response from the Identity Provider, including the audit Id generated for the request in step 2
c. performs any processing required to apply the attribute sharing policies
d. maps the elements from the response from the Identity Provider to the required elements in the authentication response to the Relying Party and generates the authentication response to send to the Relying Party
e. logs the response to the Relying Party, including the audit Id generated for the request in step 2

7. The Identity Exchange sends the authentication response to the Relying Party.

A Relying Party **MUST** generate the authentication request in step 1 in accordance with one of the following profiles:

- TDIF: OpenID Connect 1.0 Profile – using the Relying Party to Identity Exchange Profile specified in this document.
- TDIF: SAML 2.0 Profile – using the Relying Party to Identity Exchange Profile specified in [TDIF.SAML]
The Identity Exchange **MUST** generate a unique audit Id for from Relying Party as described in step 2. It **MUST** log the all related interactions between Relying Parties and Identity Providers using this unique audit id. The Identity Exchange must return this audit Id to the Relying Party using the `RP_audit_id` attribute described in the TDIF Attribute Profile [TDIF.Attributes].

The Identity Exchange **MUST** implement the mapping described in steps 3 and 6 using the mappings specified in section 3.1.1.

### 3.1.1 Trust Broker Protocol Mappings

The following mappings apply when an Identity Exchange brokers a request from a Relying Party to an Identity Provider:

- Where the authentication request from the Relying Party is SAML, and is brokered to an Identity Provider using OIDC, then the processing rules specified in Section 3.1.5 must be applied.
- Where the authentication request from the Relying Party is SAML, and is being brokered to an Identity Provider using SAML 2.0, then the processing rules specified in Section 3.1.4 must be applied.
- Where the authentication request is OIDC, the required processing rules are described in the TDIF OpenID Connect 1.0 Profile document [TDIF.OIDC].

### 3.1.2 Pairwise Identifiers and Identity Resolution

To operate as a Trust Broker in the identity federation an Identity Exchange is required to implement an Identity Resolution process whereby the Identity Exchange maps the identities managed by Identity Providers to the records held by the Relying Parties that consume these identities. Central to the function of identity resolution is the use of pairwise identifiers to map the identity of a user at an Identity Provider to a record at a Relying Party. Separate pairwise identifiers are used in the following interactions for a user as follows:

- Between the Identity Exchange and an Identity Provider. A unique pairwise identifier is generated by an Identity Provider for each they that they authenticate as part of a Trust Broker interaction.
• Between the Identity Exchange and a Relying Party. A unique pairwise identifier for the user is allocated by the Identity Exchange for each Relying Party that the takes part in a Trust Broker Interaction,

An Identity Exchange **MUST** implement an identity resolution process that maps the pairwise identifier presented by an Identity Provider in response to an authentication request to the pairwise identifier for the user at the Relying Party that initiated the authentication interaction. A Relying Party will always be presented with the same pairwise identifier as the subject identifier whenever the user uses the same Identity Provider as part of a Trust Broker interaction. There is no correlation of identity across Identity Providers.

Recommendations for the creation Pairwise Identifiers are contained in section 8.1 of [OpenID.Core] for OIDC and as described in section 3.4 of [SAML-SubjectID-v1.0] for SAML. The requirements for an Identity Exchange interacting with Relying Parties are more flexible to cater for the transition of Relying Parties from a legacy federation to an Identity Exchange.

Identity Providers **MUST NOT** generate Identifiers greater than 255 ASCII characters.

The Identify Exchange **MUST** be able to receive pairwise identifiers of up to 255 ASCII characters.

An Identity Exchange **MAY** generate Identifiers in accordance with the OIDC specification [OpenID.Core] and use these to interact with Relying Parties regardless of federation protocol.

An Identity Exchange **MAY** advertise a maximum length of its pairwise Identifiers based on the mechanism it uses.

### 3.1.3 Assurance Levels

TDIF assurance levels are represented in the technical integration standards by values of an Authentication Context Class Reference (acr). Acr is a concept supported by both the OpenID Connect 1.0 and SAML 2.0 standards.
TDIF assurance levels are represented using the values that defined in section 2.2.8 of the TDIF Attribute Profile [TDIF.Attributes].

Required acr values are represented in an OIDC authentication request using either the acr_values parameter or the acr claim. Required acr values are represented in a SAML request using the `<saml:AuthnContextClassRef>` element. Multiple acr values may be included in OIDC and SAML authentication requests. OIDC does not provide a mechanism for specifying an acr value as a minimum required acr. This profile describes a mechanism whereby a Relying Party can specify a single acr value as a minimum required value in an SAML authentication request to an Identity Exchange and have it reliably proxied to a SAML or OIDC Identity Provider.

### 3.1.4 SAML to SAML Brokering

When the Identity Exchange is accepting requests from a SAML Relying Party and translating those requests to a SAML Identity Provider, the Identity Exchange **MUST** interact with the Identity Provider as per the SAML profile, specified in this document, with the following processing rules.

#### 3.1.4.1 Mapping Attributes

Where the attributes required are predefined within the Relying Parties metadata, the set of required attributes **MUST** be included in the request to the Identity Provider with the following processing rules:

- Where the requested attributes contained within the Relying Party’s metadata are the same as the Identity Exchanges requested attributes in its metadata exchanged with the Identity Provider; the Identity Exchange creates a standard authentication request
- Where the requested attributes are not available in the requested attributes as part of the metadata shared with the Identity Provider by the Identity Exchange; the Identity Exchange is required to create an authentication request to the Identity Exchange using extensions to request the attributes required by the Relying Party

Where the attributes requested by a Relying Party are requested via extensions the Identity exchange **MUST** copy those attributes into the authentication request to the Identity Provider as extensions.
3.1.4.2 Subjects within Requests

The Relying Party may include a SAML Subject in the authentication request. As the subject identifier is pairwise identifier for the user at the Relying Party, the Identity Exchange must resolve this pairwise identifier in any request to an existing pairwise identifier for the user at the required Identity Provider. If no pairwise identifier for the user at the Identity Provider can be resolved then the Identity Exchange SHOULD return an error.

3.1.4.3 Mapping Assurance Levels

Where the Relying Party includes a <RequestedAuthnContext> in the authentication request, the Identity Exchange is required to send the set of <AuthnContextClassRef> to the Identity Provider that meet or exceed the originally requested <RequestedAuthnContext> as described in 4.6 Authentication Context Class Reference.

The Comparison attribute for the <RequestedAuthnContext> MUST be set to exact or minimum.

3.1.4.4 Other SAML Request Parameters

3.1.4.4.1 ForceAuthn Attribute

When the ForceAuthn attribute is set to true within the Authentication Request from the Relying Party this MUST be passed through in the Authentication sent by the Identity Exchange to the Identity Provider

3.1.4.4.2 isPassive Attribute

When the isPassive attribute is set to true within the Authentication Request from the Relying Party this MUST be passed through in the Authentication sent by the Identity Exchange to the Identity Provider

3.1.5 SAML to OIDC Brokering

When the Identity Exchange is accepting requests from a SAML Relying Party and translating those requests to an OIDC Identity Provider, the Identity Exchange MUST
interact with the Identity Provider as per of OIDC Profile [TDIF.OIDC] with the following processing rules.

3.1.5.1 Mapping Attributes to Claims or Scopes

The attributes requested within the Authentication Request either through extensions or via the Relying Party’s metadata MUST be processed in accordance with the following rules:

- All attributes included in the Relying Party’s Authentication Request MUST be included in the authentication request sent to the Identity Provider in either Scopes or Claims
- Where the attributes can be mapped fully into an available scope the Identity Exchange SHOULD request those scopes from the Identity Provider.
- Where the attributes do not map fully into a Scope the Identity Exchange MUST requests those attributes as claims from the Identity Provider.

3.1.5.2 Mapping Assurance Levels

Where the Relying Party includes a <RequestedAuthnContext> in the authentication request, the Identity Exchange is required to send the set of acr values to the Identity Provider that meet or exceed the originally requested <RequestedAuthnContext> as described in 4.7 Authentication Context Class Reference. The set of acr values SHOULD use the acr claim. The acr claim SHOULD be marked as essential.

The Comparison attribute for the <RequestedAuthnContext> MUST be set to exact or minimum.

3.1.5.3 Other SAML Request Parameters

3.1.5.3.1 ForceAuthn

Where the ForceAuthn attribute is included in the authentication request from the Relying Party, the Identity Exchange must set the prompt parameter to login in the OIDC authentication request to the Identity Provider.
3.1.5.3.2 isPassive

Where the `isPassive` attribute is included in the authentication request from the Relying Party, the Identity Exchange must set the `prompt` parameter to `none` in the OIDC authentication request to the Identity Provider.

3.1.5.3.3 Subject

Where a `Subject` is included in the authentication request from the Relying Party the Identity Exchange is required to validate the token and extract the subject. The Identity Exchange must resolve this to a subject identifier at the Identity Provider as per 3.1.4.2. The Identity Exchange **SHOULD** include the resolved subject identifier in the authentication request to the Identity Provider using the `sub` (subject) Claim.

3.2 Attribute Sharing

The Identity Exchange is required to implement the Attribute Sharing Policies defined in the TDIF Attribute Profile [TDIF.Attributes]
4 Common Profile Requirements

4.1 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

4.2 General

4.2.1 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.

4.2.2 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]

4.2.3 Document Type Definitions

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

4.3 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.
4.3.1 Metadata Exchange

4.3.1.1 Metadata Acquisition Method

Implementations **MUST** support the routine consumption of SAML metadata from a remote location via HTTP/1.1 ([RFC2616]) 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.

4.3.1.2 Metadata 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.

4.3.1.3 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 `validUntil` XML attribute on the root element is missing
• The value of the validUntil XML attribute on the root element is a \textit{xsd:dateTime} in the past
• The value of the validUntil XML attribute on the root element is a \textit{xsd:dateTime} too far into the future, where too far into the future is a configurable option

\textbf{Note:} this requirement applies to the root element only. Any validUntil XML attributes in child elements must be processed in accordance with [SAML2Meta]

4.3.2 Metadata Usage

Implementations \textbf{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-xsd]
• SAML V2.0 Metadata Interoperability Profile [SAML2MDIOP]
• SAML V2.0 Metadata Extension for Algorithm Support[SAML2MetaAlgSup]

Implementations \textbf{MAY} support:

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

SPs \textbf{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 4.4 below, other metadata may be present and \textbf{MUST NOT} prevent the consumption and use of the metadata.

Implementations \textbf{MUST} support the interpretation and application of metadata as defined by SAML 2.0 Metadata Interoperability Profile [SAML2MDIOP]. Implementations \textbf{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.
4.3.2.1 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.

4.3.2.2 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.

4.3.2.3 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 [SAML2Errata]
4.4 Web Browser SSO

Implementations **MUST** support the SAML 2.0 Web Browser SSO profile as defined in [SAML2Prof] 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].
4.5 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-xsd]` or `[SAML2-xsd]` 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.

4.6 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 `[RFC4051]`

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]`

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 - 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** **SHOUD** 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 [RFC7457] 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 [RFC4051]
4.7 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.

```xml
<saml:RequestedAuthnContext Comparison="minimum">
  <saml:AuthnContextClassRef>
  </saml:AuthnContextClassRef>
</saml: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].

```xml
<saml:AuthnStatement AuthnInstant="2017-07-17T01:01:48Z"
  SessionNotOnOrAfter="2017-07-17T09:01:48Z"
  SessionIndex="_be9967abd904ddcae3c0eb4189adbe3f71e327cf93">
  <saml:AuthnContext>
    <saml:AuthnContextClassRef>
    </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.
4.8 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>
5 Relying Party (SP) to Identity Exchange (IdP) Profile

5.1 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.

5.1.1 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 4.6 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.

### 5.1.1.1 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 make 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.

### 5.2 Identity Exchange Profile (IdP)

#### 5.2.1 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 <strong>MUST</strong> 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 <strong>MUST</strong> only be released in accordance with the Attribute Sharing Policies set out in [TDIF.Attributes]

Identity Providers <strong>MUST</strong> 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 <strong>MUST</strong> support the ForceAuthn attribute in the <samlp:AuthnRequest> messages as defined in [SAML2Core]. The authentication mechanism within an implementation <strong>MUST</strong> 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 <strong>MUST</strong> support the isPassive attribute in <samlp:AuthnRequest> messages as defined in [SAML2Core]

Identity Providers <strong>MUST</strong> 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]
6 Identity Exchange (SP) to Identity Provider (IdP) Profile

6.1 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

6.1.1 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.

*6.1.1.1 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.

### 6.2 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.

#### 6.2.1 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 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 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]
7 References

The following information sources have been used in developing this document.


7.1 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 3 – User Authentication Sequence Diagrams (steps 1 to 5)**

1. User discovers the service from a website, search engine or via a service aggregation portal.
   1.1 User attempts to access an authenticated service.

2. Relying Party specifies the identity requirements for service as part of the authentication request to Exchange.
   2.1 «Front Channel Transfer>> User redirected to Exchange (Authentication Request)

3. Exchange performs IDP Selection. Exchange determines what available IDPs may meet the requirement and checks whether the user's preferred IDP has been remembered.
   3.0 Determine IDP Requirements
   3.1 Exchange prompts user to select IDP from list of available
   User selects IDP from list

4. Exchange sends an authentication request to the selected IDP. The request includes the attributes required by the RP.
   4.1 «Front Channel Transfer>> User redirected to selected IDP (Authentication Request)

5. Authenticate User. User needs to be authenticated by the IDP. If they already have an account they login, otherwise they need to create an account.

alt [0.2 IDP selection is required]
Exchange prompts user to select IDP from list of available
User selects IDP from list

alt [0.1 User already has account]
User logs in

alt [0.2 User needs to create an account]
User creates an account

alt [0.3 Authentication Fails]
«Front Channel Transfer>> Failure (Authentication Response)

alt [0.4 User cancels process]
Exchange may further interact with the user if required e.g., the user may wish to select another IDP.
Figure 4 – User Authentication (steps 6 to 11)

Figure 3 and Figure 4 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 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.

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

5.3. Authentication Fails
   o 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.
   o 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.

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 requestor

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.

   - 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.

Step 6 uses the Identity Exchange to Identity Provider Profile
- User Cancels Process: IDP responds with error status that the error was on the part of the requestor
- 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.

  o 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

  o 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.

  o 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.

  o 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):

```xml
  <saml:Issuer>http://client.example.org/someapp/metadata</saml:Issuer>  
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha256"/>  
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>  
    <ds:Reference URI="#pfx41d8ef22-e612-8c50-9960-1b16f15741b3">  
      <ds:Transforms>  
        <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>  
      </ds:Transforms>  
    </ds:Reference>  
  </ds:Signature>
</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

\(<\text{samlp:AuthenticationContextClassRef}>urn:id.gov.au:tdif:acr:ip3:cl2</\text{samlp:AuthenticationContextClassRef}>\) value that was requested in the authentication request from the relying party.
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
<?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"
    InResponseTo="IDP_4fee3b046395c4e751011e9f8900b5273d56685">
    <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+AymJwyRQKpjp5xzLkM1EiAsr/CyolvhtjK5G71lRO5gh53CR3dGT0YBIIdvm ...Vcd8no4L6jMlrUH0DXB9yY=</ds:SignatureValue>
    </ds:Signature>
</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
<?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">
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <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:SignatureValue></ds:SignatureValue>
            <ds:KeyInfo>
                <ds:X509Data>
                    <ds:X509Certificate>
                        MIICajCCAdOgAwIBAgIBADANBgkqhkiG9w0BAQsFAdBMQswCQYDVQQGEwJ1czETM®EBA1UECA...
                        4LzgD0CROMASTWNG==
                    </ds:X509Certificate>
                    <ds:X509Data>
                        MIICajCCAdOgAwIBAgIBADANBgkqhkiG9w0BAQsFAdBMQswCQYDVQQGEwJ1czETMBEGA1UECA...
                        4LzgD0CROMASTWNG==
                    </ds:X509Data>
                </ds:X509Data>
            </ds:KeyInfo>
            <ds:SignatureValue></ds:SignatureValue>
        </ds:Reference>
    </ds:Signature>
    <samlp:Status>
    </samlp:Status>
    <saml:Assertion xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.w3.org/2001/XMLSchema"
        ID="pfx8f564359-2bce-a367-99d0-12c8601bad9"
        Version="2.0" IssueInstant="2014-07-17T01:01:48Z">
        <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/2001/10/xml-exc-c14n#"/>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>
    <ds:DigestValue>/ExPJp1cibZFT/...sLI8BUN1gzi1pI=/</ds:DigestValue>
  </ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
  lXrbu/LvXPm08Z0j/1FFkRo8smL2oiwINWab6AM5mv7LnSWb8IF/5cni jXN2k2C5xgGnM49WMbs...apb440F7I1AAhnEcTTIVKFe=
</ds:SignatureValue>
<ds:KeyInfo>
  <ds:X509Data>
    <ds:X509Certificate>
      MIICajCCAdOgAwIBAgIBADANBgkqhkiG9w0BAQsFADBSMQswCQYDVQQGEwJ1czETMBEGA1UECA...
    </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:AudienceRestriction />
  <saml:Conditions />
</saml:Conditions>
<saml:AuthnStatement AuthnInstant="2014-07-17T01:01:48Z" SessionNotOnOrAfter="2014-07-17T09:01:48Z" SessionIndex="_be9967abd904ddcae3c0eb4189adbe3f71e327cf93">
  <saml:AuthnContext>
  </saml:AuthnContext>
</saml:AuthnStatement>

http://client.example.org/someapp/metadata
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.