Australian Government
Digital Transformation Agency

OpenID Connect 1.0 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
0.1	Feb 18	BF	Initial version for internal review
0.2	Feb 18	BF	Major revision, moved to new template. Added chapter on Identity Exchange.
0.25	Mar 18	BF, TM	Updated content to conform to template.
0.3	Mar 18	TM	Minor updates
0.4	Aug 18	TM	Updates from consultation with key stakeholders.
1.0	Aug 18		Endorsed for release by the Commonwealth GovPass Authority
1.1	Sep 18	TM	Content updates
1.2	Mar 19	TM	Revision from consultation feedback from TDIF release 3.

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 OpenID Connect 1.0 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, the relationship between its various documents and the definition of key terms.
- *TDIF: Architecture Overview,* which provides 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.

The *TDIF: OpenID Connect 1.0 Profile* is built on top of and inherits the properties of the OpenID Connect 1.0 standard **[OpenID.Core]**. The OpenID Connect 1.0 standard itself is built on top of the OAuth 2.0 standard.

This OpenID Connect 1.0 profile is consistent with the International Government Assurance Profile (iGov) for OpenID Connect 1.0 – Draft 02. Some features of iGov profile are not used. The differences between the *TDIF: OpenID Connect 1.0 Profile* and iGov Profile are noted in **Appendix B iGov Profile**.

2 Relying Party to Identity Exchange Profile

This section describes the OpenID Connect 1.0 Profile for the use between Relying Parties and an Identity Exchange, acting as an OpenID provider (OP), and operating as part of the TDIF.

Note: an OpenID Provider (OP) is an OAuth 2.0 Authorisation Server (AS) that is capable of authenticating the End User and providing claims to a Relying Party (RP) about the authentication event and the End User. Any use of the terms OpenID Provider (OP) or Authorisation Server (AS) within this profile can be considered congruous.

2.1 Client Types

The following client type descriptions give patterns for deployment for use in different types of client applications based on the OAuth grant type. The resource owner password credentials grant type is defined in **[RFC 6749]** is intentionally omitted and **MUST NOT** be used under these profiles.

These client types are as described in the iGov OAuth 2.0 profiles.

2.1.1 Full Client With User Delegation

This client type applies to clients that act on behalf of a particular resource owner and require delegation of that user's authority to access the protected resource. Furthermore, these clients are capable of interacting with a separate web browser application to facilitate the resource owner's interaction with the authentication endpoint of the authorisation server.

These clients **MUST** use the authorisation code flow of OAuth 2 by sending the resource owner to the authorisation endpoint to obtain authorisation. The user **MUST** authenticate to the authorisation endpoint. The user's web browser is then redirected back to a URI hosted by the client service, from which the client can obtain an authorisation code passed as a query parameter. The client then presents that authorisation code along with its own credentials (`private_key_jwt`) to the authorisation server's token endpoint to obtain an access token.

These clients **MUST** be associated with a unique public key as described in the Client Keys section 2.4 below.

This client type **MAY** request and be issued a refresh token if the security parameters of the request allow for it.

2.1.2 Native Client with User Delegation

This client type applies to clients that act on behalf of a particular resource, such as an application on a mobile platform, and require delegation of that user's authority to the protected resource. Furthermore, these clients are capable of interacting with a separate web browser application to facilitate the resource owner's interaction with the authentication endpoint of the authorisation server. In particular this client type runs natively on the resource owner's device, often leading to many identical instances of a piece of software operating in different environments and running simultaneously for different end users.

These clients **MUST** use the authorization code flow of OAuth 2 by sending the resource owner to the authorisation endpoint to obtain authorisation. The user **MUST** authenticate to the authorisation endpoint. The user's web browser is then redirected back to a URI hosted by the client, from which the client can obtain an authorisation code passed as a query parameter. The client then presents that authorisation code along to the authorisation servers token endpoint to obtain an access token.

Native clients **MUST** either:

- Use dynamic client registration to obtain a separate client id for each instance.
- Use a common client id and use PKCE to protect calls to the token endpoint.

Native applications using dynamic registration **SHOULD** generate a unique public and private key pair on the device and register that public key value with the authorisation server. Alternatively, an authorisation server **MAY** issue a public and private key pair to the client as part of the registration process. In such cases, the authorisation server **MUST** discard its copy of the private key. Client credentials **MUST NOT** be shared among instances of client software.

Native Applications not registering a separate public key for each instance **MUST** use PKCE with the S256 code challenge mechanism.

Dynamically registered native applications **MAY** use PKCE.

Native applications not registering a separate public key for each instance are considered Public Clients, and **MUST** use PKCE with the S256 code challenge mechanism. Public Clients do not authenticate with the Token Endpoint in any other way.

This client type **MAY** request and be issued with a refresh token.

2.2 Client Registration

All clients **MUST** register with the authorisation server. For client software that may be installed on multiple client instances, each client instance must receive a unique client identifier from the authorisation server.

Client registration **MAY** be performed by either static configuration or dynamically.

To register a client statically the Client will need to provide the information required by the Identity Exchange. As a minimum the following will be required for every client for static configuration:

- `client_id`.
    - the `client_id` is generated by the Identity Exchange as the OP and is used to identify the client in requests.
- `redirect_uri`.
    - as described below in section 2.3.
- Grant type.
    - as described below in section 2.5.
- Client keys.
    - as described below in section 2.4.

Clients are also able to register dynamically as described in the IdP profile below in section 2.7.2.3.

2.3 Redirect URI

Clients using the authorisation code grant type **MUST** register their full redirect URIs. The authorisation server **MUST** validate the redirect URI given by the client at the authorisation endpoint using strict string comparison.

A client **MUST** protect the values passed back to its redirect URI by ensuring that the redirect URI is one of the following:

- Hosted on a website with TLS protection (HTTPS).
- Hosted on a local domain of the client (e.g. http://localhost/).
- Hosted on a client specific non-remote protocol URI scheme (e.g. myapp:// or au.gov.app://).

Clients **MUST NOT** have URIs in more than one category and **SHOULD NOT** have multiple redirect URIs on different domains.

Clients **MUST NOT** forward values passed back to their redirect URIs to other arbitrary or user-provided URIs (i.e. no open redirectors).

2.3.1 Native Client Applications

Native Mobile and Desktop applications have slightly different requirements to those of Web Applications as there may not be a browser available to perform redirections nor a specific RP to redirect to at the time of authentication. As such these applications are generally required to register the local domain or a client specific non-remote protocol URI scheme as the redirect URI. These schemes **MAY** continue to be used for existing applications.

Where the native platform supports the newer App-Claimed HTTPS URL redirection capability (Android, and iOS 9 or greater) this method **SHOULD** be used. This method allows the registration of a HTTPS URL e.g. https://app.gov.au/auth. When that URL is accessed the platform will open the application (if not already open) and the required actions can be performed.

2.4 Client Keys

Clients using the authorisation code grant type **MUST** have a public and private key pair type for use in authentication to the token endpoint. The client **MUST** register their public keys in their client registration metadata by either sending the public key directly in the jwks field or by registering a `jwks_uri` that **MUST** be reachable by the authorisation server. It is recommended that clients use a `jwks_uri` as it allows for easier key rotation.

The jwks field or the content available from the `jwks_uri` of a client **MUST** contain a public key in JSON Web Key Set (JWK Set) format. The authorisation server **MUST** validate the content of the clients registered `jwks_uri` document and verify that it contains a JWK Set. The example below is a 2048 bit RSA key.

```
{
   "keys": [
     {
       "alg": "RS256",
       "e": "AQAB",
       "n": "kAMYD62n_f2rUcR4awJX4uccDt0zcXRssq_mDch5-
ifcShx9aTtTVza23P
Tn3KaKrsBXwWcfioXR6zQn5eYdZQVGNBfOR4rxF5i7t3hfb4WkS50EK1gBYk2lO9NSrQ-
xG9QsUsAnN6RHksXqsdOqvnxjLexDfIJlgbcCN9h6TBC66ZXv7PVhl19gIYVifSU7liHk
Le0l0fw7jUI6rHLHf4d96_neR1HrNIK_xssr99Xpv1EM_ubxpktX0T925qej9fMEpzzQ5
HLmcNt1H2_VQ_Ww1JOLn9vRnH48FDj7TxlIT74XdTZgTv31w_GRPAOfyxEw_ZUmxhz5Z-
gTlQ",
       "kty": "RSA",
       "kid": "oauth-client"
     }
   ]
}
```

Native Client applications **MAY** omit their key during registration if they are a public client using PKCE.

2.5 Grant Types

The only supported grant type is `authorization_code`. Authentication using the Authorisation Code flow is described in section 3.1 of **[OpenIDCore]**.

2.6 Relying Party Profile

2.6.1 Requests to the Authorization Endpoint (Authentication Request)

Clients making a request to the authorisation endpoint **MUST** use an unpredictable value for the state parameter with at least 128 bits of entropy. Clients **MUST** validate the value of the state parameter upon return to the redirect URI and **MUST** ensure that the state value is securely tied to the user's current session e.g. by relating the `state` value to a session identifier issued by the client software to the browser.

Clients **MUST** include their full redirect URIs in the authorisation request. To prevent open redirection and other injection attacks, the authorisation server **MUST** match the entire redirect URI using a direct string comparison against registered values and **MUST** reject requests with invalid or missing redirect URIs.

Request Parameters:

- `client_id.`
    - REQUIRED. OAUTH 2.0 Client Identifier valid at the Authorisation Server.
- `response_type.`
    - REQUIRED. Must be set to `code.`
- `Scope.`
    - REQUIRED. `Indicates the attributes being requested. The `openid` scope **MUST** always be present. Other defined scopes are described in *TDIF: Attribute Profile*.
- `redirect_uri.`
    - REQUIRED. Indicates a valid endpoint where the client will receive the authentication response. The URI must match exactly one of the Redirection URIs preregistered at the OP. The redirection URI **SHOULD** use the https scheme.
- `State.`

- REQUIRED. Un-guessable random string generated by the RP used to protect against CSRF attacks. Must contain a sufficient amount of entropy to avoid guessing and is returned to the RP in the authentication response.

- `Prompt`.
  - OPTIONAL. A space delimited, case sensitive list of string values that specifies if the Authorisation Server prompts for the End User to re-authenticate or provide consent. Defined values are:

    - `none`: the Authorisation Server **MUST NOT** display any authentication or consent user interface pages. An error is returned if the end-user is not already authenticated or not already provided consent for the requested claims or does not fulfil any other conditions for processing the request.
    - `login`: the Authorisation Server **SHOULD** prompt the end user for re-authentication.
    - `consent`: The Authorisation Server **SHOULD** prompt the end-user for consent `before` returning information to the Client. Consent should be requested in accordance with the Attribute Sharing Policies defined in *TDIF: Attribute Profile.*
    - `select_account`: The Authorisation Server **SHOULD** prompt the end-user to select a user account. This allows an end-user with multiple accounts at the authorisation server to select amongst the accounts they currently have a session for.
- `Display`.
  - OPTIONAL: A string value that specifies how the Authorisation Server displays the authentication and consent interface pages to the end-user.
    - `page`: The Authorisation Server should the authentication and consent UI consistent with a full User Agent page view. This is the default where the display parameter is not specified.
    - `popup`: The Authorisation Server **SHOULD** display the authentication and consent UI consistent with a popup User Agent window. The popup User Agent window should be of an appropriate size for a login-focused dialog and should not obscure the entire window that it is popping up over.

        - `touch`: The Authorisation Server **SHOULD** display the authentication and consent UI consistent with a device that leverages a touch interface.
        - `wap`: The Authorization Server **SHOULD** display the authentication and consent UI consistent with a "feature phone" type display.
- `Nonce`.
    - REQUIRED. Un-guessable random string generated by the client, used to protect against CSRF attacks. Must contain sufficient entropy to avoid guessing. Returned to the Client in the ID Token.
- `acr_values`.
    - OPTIONAL. A Relying party may specify the required Level of Assurance as defined in the *TDIF: Attribute Profile*.
- `code_challenge` and `code_challenge_method`.
    - OPTIONAL. These parameters are required to support the use of PKCE with the S256 code challenge mechanism. as described in section 2.1.2.
- `max_age`.
    - OPTIONAL. Maximum Authentication Age. Specifies the allowable elapsed time in seconds since the last time the End-User was actively authenticated by the OP. If the elapsed time is greater than this value, the OP **MUST** attempt to actively re-authenticate the End-User.
- `ui_locales`.
    - OPTIONAL. End-User's preferred languages and scripts for the user interface, represented as a space-separated list of language tag values as specified in **[RFC 5646]**, ordered by preference.
- `id_token_hint`.
    - OPTIONAL. ID Token previously issued by the Authorisation Server being passed as a hint about the End-User's current or past authenticated session with the Client. If the End-User identified by the ID Token is logged in or is logged in by the request, then the Authorization Server returns a positive response; otherwise, it **SHOULD** return an error, such as login_required.
- `login_hint`.
    - OPTIONAL. Hint to the Authorisation Server about the login identifier the End-User might use to log in (if necessary). This hint can be used by an RP

if it first asks the End-User for their e-mail address (or other identifier) and then wants to pass that value as a hint to the discovered authorisation service.

A sample request may look like the following example:

```
https://idexchange.gov.au/oidc/authorization?
  response_type=code
  &client_id=827937609728-m2mvqffo9bsefh4di90saus4n0diar2h
  &scope=profile%20openid%20email
  &redirect_uri=https%3A%2F%2Frp.agency.gov.au%2Foidc%2FloginResponse
  &state=2ca3359dfbfd0
  &prompt=select_account
  &acr_values=urn%3Aid.gov.au%3Atdif%3Aacr%3Aip3%3Acl2
```

2.6.2 Requests to the Token Endpoint

Requests to the token endpoint require client authentication. The client authentication mechanism is a signed JWT and is defined in section 2.7.2.2.

The claims that are included in the JWT are summarised below:

- `Iss.`
  - The client Id of the client creating the JWT.
- `Sub.`
  - The client Id of the client creating the JWT.
- `Aud.`
  - The URL of the authorisation server's token endpoint.
- `Iat.`
  - The time that the token was created by the client.
- `Exp.`
  - The expiration time after which the token must be considered invalid.
- `Jti.`
  - A unique, random, identifier generated by the client for this authentication.

The following is an example of the use of the required claims for a client authentication JWT as defined in this profile. Additional claims **MAY** be included in this set.

```
{
   "iss": "55f9f559-2496-49d4-b6c3-351a586b7484",
   "sub": "55f9f559-2496-49d4-b6c3-351a586b7484",
   "aud": "https://idexchange.gov.au/token",
   "iat": 1418698788,
   "exp": 1418698848,
   "jti": "1418698788/107c4da5194df463e52b56865c5af34e5595"
}
```

The JWT assertion **MUST** be signed by the client using the client's private key. See section 2.4 for the mechanisms by the client can make its public key known to the authorization server.

For clients that are required to use PKCE as described in section 2.1.2, the following claims **MUST** be included in the request to the token endpoint.

- `code_verifier`.
    - Code verifier generated by client to use PKCE with the S256 code challenge mechanism.

The following claims **MUST** be included in the request to the token endpoint:

- `grant_type`.
    - **MUST** be set to `authorization_code`
- `Code`.
    - The value of the `code` parameter returned in the authorisation response.
- `redirect_uri`.
    - Value **MUST** be identical to the value of the `redirect_uri` parameter that was included in the authorisation request as described in section 2.6.1.
- `client_assertion_type`.
    - **MUST** be set to: `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.
- `client_assertion`.
    - The value of the signed client authentication JWT generated as described below in the ID Tokens section. The RP must generate a new assertion JWT for each call to the token endpoint.

These would be sent to the token endpoint as shown in the example below.

```
POST /token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: idexchange.gov.au

grant_type=authorization_code
&code=sedaFh
&scope=openid+email+profile
&redirect_uri=https%3A%2F%2Frp.agency.gov.au%2Foidc%2FloginResponse
&client_id=55f9f559-2496-49d4-b6c3-351a586b7484
&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-
assertion-type%3Ajwt-bearer
&client_assertion=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.ew0KICAgImlzcy
I6ICI1NWY5ZjU1OS0yNDk2LTQ5ZDQtYjZjMy0zNTFhNTg2Yjc0ODQiLA0KICAgInN1YiI
6ICI1NWY5ZjU1OS0yNDk2LTQ5ZDQtYjZjMy0zNTFhNTg2Yjc0ODQiLA0KICAgImF1ZCI6
ICJodHRwczovL2lkcC1wLmV4YW1wbGUuY29tL3Rva2VuIiwNCiAgICJpYXQiOiAxNDE4N
jk4Nzg4LA0KICAgImV4cCI6IDE0MTg2OTg4NDgsDQogICAianRpIjogIjE0MTg2OTg3OD
gvMTA3YzRkYTUxOTRkZjQ2M2U1MmI1Njg2NWM1YWYzNGU1NTk1Ig0KfQ.t_gX8JQGq3G2
OEc2kUCQ8zVj7pqff87Sua5nktLIHj28l5onO5VpsL4sRHIGOvrpo7XO6jgtPWy3iLXv3
NLyo1TWHbtErQEGpmf7nKiNxVCXlGYJXSDJB6shP3OfvdUc24urPJNUGBEDptIgT7Lhf6
BbwQNlMQubNeOPRFDqQoLWqe7UxuI06dKX3SEQRMqcxYSIAfP7CQZ4WLuKXb6oEbaqz6g
L4l6p83G7wKGDeLETOTHsztjKR38v4F_MnSrx8e0iIqgZwurW0RtetEWvynOCJXkp166T
7qZR45xuCxgOotXY6O3et4n77GtgspMgOEKj3b_WpCiuNEwQ
```

2.6.3 Request to the UserInfo Endpoint

The client may send a UserInfo Request using either a HTTP `GET` or HTTP `POST`. The Access Token obtained from an OpenID Connect Authentication Request **MUST** be sent as a Bearer Token as per section 2 of OAuth Bearer Token Usage.

It is RECOMMENDED that the request use the HTTP `GET` method and the Access Token is sent using the `Authorization` header field.

The following is an example of a request to a UserInfo Endpoint.

```
GET /userinfo HTTP/1.1
  Host: idexchange.gov.au
  Authorization: Bearer SlAV32hkKG
```

2.6.4 ID Tokens

All clients must validate the signature of an ID Token before accepting it using the public key of the issuing server, published in JSON Web Key (JWK) format. ID Tokens **MAY** be encrypted using the appropriate key of the requesting client. Note that the issuing server is the OP (Identity Exchange)

Clients **MUST** verify the following in received ID tokens:

- `Iss.`
    - The Issuer field is the URL of the expected issuer.
- `Aud.`
    - The audience field contains the client ID of the client.
- `exp`, `iat`, `nbf.`
    - The expiration, issued at and not before tokens are dates (integer number of seconds since 00:00:00Z 1st January 1970, i.e. Unix epoch) are within acceptable ranges.

2.6.5 Request Objects

Clients **MAY** optionally send requests to the Authorization Endpoint using the `request` parameter as defined in **[OpenIDCore]**.

Request objects **MUST** be signed by the clients registered key. Request objects **MAY** be encrypted to the Authorisation Server's public key.

2.6.6 Authentication Context

The permissible values for the acr (authentication context class reference) that Relying Party may request are and how they are requested is defined in section 2.8.3.

The RP **MAY** use either `acr_values` or the `acr` claim to request an acr.

The RP **MUST NOT** use `acr_values` and the `acr` claim in the same request.

2.6.7 Discovery

Clients and protected resources **SHOULD** cache OpenID Provider (OP) metadata once an OP has been discovered and used by the Client.

2.7 Identity Exchange Profile (IdP)

2.7.1 Audit Logging

The Identity Exchange **MUST** generate a unique audit Id for an authentication request from Relying Party. It **MUST** log the all related interactions with Relying Parties using this unique audit id. This unique audit id is returned to the Relying Party as a claim in the ID Token as specified in the *TDIF: Attribute Profile*.

2.7.2 Connecting to Clients

2.7.2.1 Grant Types

The only supported grant type is `authorization_code`. This grant type is described in section 4.1 of **[RFC 6749]**.

The Authorization Code Flow is the only authentication flow supported by this federation. The Authorization Code Flow returns an Authorization Code to the client that the client can then exchange for an ID Token and an Access Token. This provides the benefit of not exposing any tokens to the User Agent and potentially malicious applications with access to the User Agent.

The authorisation server **MUST** validate all redirect URIs for the `authorization_code` grant type.

2.7.2.2 Client Authentication

The authorisation server **MUST** enforce client authentication to the authorization servers token endpoint using a JWT assertion as defined by using only the `private_key_jwt` method as described in **[OpenIDCore]**. Clients that have registered a public key sign a JWT using the associated private key. The Client authenticates in accordance with JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants and Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants.

The JWT must expire and **SHOULD** have a lifetime no longer than five minutes. Short expiration times are recommended wherever practicable. The following guidance is provided in **[RFC 7523]** with regard to expiration times: the JWT MUST contain an "exp" (expiration time) claim that limits the time window during which the JWT can be used. The authorization server MUST reject any JWT with an expiration time that has passed, subject to allowable clock skew between systems. Note that the authorization server may reject JWTs with an "exp" claim value that is unreasonably far in the future.

The JWT **MUST** contain the following REQUIRED claims and **MAY** contain the following OPTIONAL Claim values:

- `Iss.`
    - REQUIRED. Issuer. This **MUST** contain the `client_id` of the client creating the token.
- `Sub.`
    - REQUIRED. Subject. This **MUST** contain the `client_id` of the client creating the token.
- `Aud.`
    - REQUIRED. Audience. The value that identifies the Authorisation Server as an intended audience. The Authorisation Server **MUST** verify that it is an intended audience for the token. The Audience **SHOULD** be the URL of the Authorisation Server's Token Endpoint.
- `Jti.`
    - REQUIRED. JWT ID. A unique identifier for the token generated by the client, which can be used to prevent reuse of the token. This identifier **MUST** contain at least 128 bits of entropy and **MUST NOT** be re-used by any subsequent authentication token.
- `Exp.`
    - REQUIRED. Expiration time on or after which the ID Token **MUST NOT** be accepted for processing.
- `Iat.`
    - OPTIONAL. Time at which the JWT was issued.

2.7.2.3 Dynamic Registration

Dynamic registration of Clients **MAY** be supported by an Identity Exchange.

2.7.2.4 Discovery

The OpenID Connect discovery standard provides a standard pragmatic way for clients to obtain configuration details for communicating with Identity Exchanges and is an important part of building a scalable federation ecosystem.

Exposing a Discovery endpoint does not put the exchange at risk of attack. Endpoints and parameters specified in the Discovery document **SHOULD** be considered public information regardless of the existence of the discovery document.

Access to the Discovery document **MAY** be protected with existing web authentication methods if required by the Identity Exchange. Credentials for the Discovery document are then managed by the Identity Exchange and support for these authentication methods is outside the scope of this specification.

Endpoints described in the Discovery document **MUST** be secured in accordance with this specification and **MAY** have additional controls the Identity Exchange wishes to support.

All OpenID Connect servers are uniquely identified by a URL known as the issuer, this will also be the case for Identity Exchanges. This URL serves as the prefix of a service discovery endpoint as specified in the OpenID Connect Discovery standard.

The discovery document **MUST** as a minimum contain the following fields:

- `Issuer.`
    - The fully qualified issuer URL of the server.
- `authorization_endpoint.`
    - The fully qualified URL of the server's authorisation endpoint defined by **[RFC 6749]**.
- `token_endpoint.`
    - The fully qualified URL of the server's token endpoint defined by **[RFC 6749]**.

- `introspection_endpoint.`
    - The fully qualified URL of the server's introspection endpoint defined by the OAuth Token Introspection RFC – **[RFC 7662]**.
- `revocation_endpoint.`
    - The fully qualified URL of the server's revocation endpoint as defined by **[RFC 7009]**.
- `jwks_uri.`
    - The fully qualified URI of the server's public key in JWK Set format as defined in **[RFC 7517]**.
- `scopes_supported.`
    - The list of scopes defined in the *TDIF: Attribute Profile* that the Identity Exchange supports.
- `claims_supported.`
    - The list of claims available in the supported scopes.

Below is an example JSON document found at the discovery endpoint for an authorisation server.

```
{
  "request_parameter_supported": true,
  "id_token_encryption_alg_values_supported": [
    "RSA-OAEP", "RSA1_5", "RSA-OAEP-256"
  ],
  "registration_endpoint": "https://idexchange.gov.au/register",
  "userinfo_signing_alg_values_supported": [
    "HS256", "HS384", "HS512", "RS256", "RS384", "RS512"
  ],
  "token_endpoint": "https://idexchange.gov.au/token",
  "request_uri_parameter_supported": false,
  "request_object_encryption_enc_values_supported": [
    "A192CBC-HS384", "A192GCM", "A256CBC+HS512",
    "A128CBC+HS256", "A256CBC-HS512",
    "A128CBC-HS256", "A128GCM", "A256GCM"
  ],
  "token_endpoint_auth_methods_supported": [
    "private_key_jwt",
  ],
  "userinfo_encryption_alg_values_supported": [
    "RSA-OAEP", "RSA1_5",
    "RSA-OAEP-256"
  ],
  "subject_types_supported": [
    "pairwise"
```

```
  ],
  "id_token_encryption_enc_values_supported": [
    "A192CBC-HS384", "A192GCM", "A256CBC+HS512",
    "A128CBC+HS256", "A256CBC-HS512", "A128CBC-HS256",
    "A128GCM", "A256GCM"
  ],
  "claims_parameter_supported": false,
  "jwks_uri": "https://idexchange.gov.au/jwk",
  "id_token_signing_alg_values_supported": [
    "HS256", "HS384", "HS512", "RS256", "RS384", "RS512", "none"
  ],
  "authorization_endpoint": "https://idexchange.gov.au/authorize",
  "require_request_uri_registration": false,
  "introspection_endpoint": "https://idexchange.gov.au/introspect",
  "request_object_encryption_alg_values_supported": [
    "RSA-OAEP",?RSA1_5", "RSA-OAEP-256"
  ],
  "service_documentation": "https://idexchange.gov.au/about",
  "response_types_supported": [
    "code"
  ],
  "token_endpoint_auth_signing_alg_values_supported": [
    "HS256", "HS384", "HS512", "RS256", "RS384", "RS512"
  ],
  "revocation_endpoint": "https://idexchange.gov.au/revoke",
  "request_object_signing_alg_values_supported": [
    "HS256", "HS384", "HS512", "RS256", "RS384", "RS512"
  ],
  "claim_types_supported": [
    "normal"
  ],
  "grant_types_supported": [
    "authorization_code",
  ],
  "scopes_supported": [
    "profile", "openid", "email", "phone"
  ],
  "userinfo_endpoint": "https://idexchange.gov.au/userinfo",
  "userinfo_encryption_enc_values_supported": [
    "A192CBC-HS384", "A192GCM", "A256CBC+HS512","A128CBC+HS256",
    "A256CBC-HS512", "A128CBC-HS256", "A128GCM", "A256GCM"
  ],
  "op_tos_uri": "https://idexchange.gov.au/about",
  "issuer": "https://idexchange.gov.au/",
  "op_policy_uri": "https://idexchange.gov.au/about",
  "claims_supported": [
    "sub", "name", "acr", "family_name", "given_name", "birthdate",
"email", "phone"
  ]
}
```

2.7.2.5 PKCE

An Authorisation Server **MUST** support the Proof Key for Code Exchange (PKCE) extension to the authorization code flow, including support for the S256 code exchange method. The Authorisation Server **MUST NOT** allow a client to use the plain code challenge method.

2.7.3 Requests to the Authorisation Endpoint (Authentication Request)

The Identity Exchange **MUST** support ALL of the mechanisms for requesting a LoA as described in the Levels of Assurance section of the *TDIF: Attribute Profile*.

The Identity Exchange **MUST** always return the `acr` claim. Valid acr values are described in section 2.8.3 below.

2.7.4 User Consent

The Identity Exchange **MUST** get consent from the user in accordance with the Attribute Sharing Policies set out in the *TDIF: Attribute Profile.*

2.7.5 Response to Authorisation Requests

The Authorization Response to the Authorization Code flow **MUST** return the following fields in the response:

- `State.`
    - The value of the state parameter passed in via the authentication request. This value **MUST** match exactly.
- `Code.`
    - The authorisation code, a random string issued by the IdP to be used in the request to the token endpoint.

The key requirements for these fields are described in the OAuth 2.0 specification **[RFC 6749]** section 4.1.2.

An example response is shown below:

```
https://idexchange.gov.au/web/oidc/loginResponse?
    state=2ca3359dfbfd0
   &code=gOIFJ1hV6Rb1sxUdFhZGACWwR1sMhYbJJcQbVJN0wHA
```

The authentication response is sent via HTTP redirect to the URI specified in the request.

2.7.6 ID Tokens

All tokens **MUST** be signed by the issuer Exchange's private key. ID Tokens **MAY** be encrypted using the appropriate key of the requesting client.

The ID Token must expire and **SHOULD** have a lifetime no longer than five minutes. Short expiration times are recommended as the ID token is consumed by the client and not presented to remote systems

The token response includes an access token, which can be used to make a UserInfo request, and ID token (a signed and optionally encrypted JSON Web Token). ID Token values have the following meanings:

- `Iss.`
    - REQUIRED. The issuer field is the URL of the expected issuer.
- `Aud.`
    - REQUIRED: The audience field contains the client ID of the client.
- `Sub.`
    - REQUIRED The identifier of the user. **MUST** be a pairwise anonymous identifier, and be unique per client to prevent linkability and traceability between clients.
- `Acr.`
    - REQUIRED. The level of assurance at which the user was authenticated at. **MUST** be a member of the `acr_values` list from the authentication request.
- `Nonce.`

    - The nonce value that was provided in the authentication request. **MUST** be included if it was provided in the authentication request.

- `Jti.`
    - REQUIRED. A unique identifier for the token which can be used to prevent the reuse of the token.

- `exp`, `iat`, `nbf.`
    - REQUIRED. The expiration, issued at, and not before timestamps for the tokens. They are dates presented as an integer representing the number of seconds since 1970-01-01T00:00:00Z UTC (Unix epoch) within acceptable ranges.

The following is an example of an ID token signed using the server's RSA key.

```
eyJhbGciOiJSUzI1NiJ9.eyJhdXRoX3RpbWUiOjE0
        MTg2OTg3ODIsImV4cCI6MTQxODY5OTQxMiwic3ViI
        joiNldaUVBwblF4ViIsIm5vbmNlIjoiMTg4NjM3Yj
        NhZjE0YSIsImF1ZCI6WyJjMWJjODRlNC00N2VlLTR
        iNjQtYmI1Mi01Y2RhNmM4MWY3ODgiXSwiaXNzIjoi
        aHR0cHM6XC9cL2lkcC1wLmV4YW1wbGUuY29tXC8iL
        CJpYXQiOjE0MTg2OTg4MTJ9mQc0rtL56dnJ7_zO_f
        x8-qObsQhXcn-qN-FC3JIDBuNmP8i11LRA_sgh_om
        RRfQAUhZD5qTRPAKbLuCD451lf7ALAUwoGg8zAASI
        5QNGXoBVVn7buxPd2SElbSnHxu0o8ZsUZZwNpircW
        NUlYLje6APJf0kre9ztTj-5J1hRKFbbHodR2I1m5q
        8zQR0ql-FoFlOfPhvfurXxCRGqP1xpvLLBUi0JAw3
        F8hZt_i1RUYWMqLQZV4VU3eVNeIPAD38qD1fxTXGV
        Ed2XDJpmlcxjrWxzJ8fGfJrbsiHCzmCjflhv34O22
        zb0lJpC0d0VScqxXjNTa2-ULyCoehLcezmssg
```

Its claims are as follows:

```
{
        "auth_time": 1418698782,
        "exp": 1418699412,
        "sub": "6WZQPpnQxV",
        "nonce": "188637b3af14a",
        "aud": "s6BhdRkqt3",
        "iss": "https:\\/\\/idexchange.gov.au\\/",
        "acr":"urn:id.gov.au:tdif:acr:ip3:cl2",
        "tdif_audit_id": "AA97B177-9383-4934-8543-0F91A7A02836"
        "iat": 1418698812
}
```

For clients using the Authorization Code grant type, access tokens **SHOULD** have a valid lifetime no greater than one hour and refresh tokens, if issued, **SHOULD** have a lifetime no longer than 24 hours. These token lifetime vales are recommended values

and MAY be varied in accordance with security risk assessment and the agreement of the Relying Parties that an Identity Exchange supports.

2.7.7 UserInfo Endpoint

Identity Exchanges **MUST** support the UserInfo endpoint for claims as described in the *TDIF: Attribute Profile*.The UserInfo endpoint **MUST** only return claims that are authorised within the authentication request that issued the access token that is being used to access the endpoint.

Support for the UserInfo endpoint is important for maximum client implementation interoperability even if no additional user information is returned. Clients are not required to call the UserInfo endpoint but should not receive an error if they do.

A request to the UserInfo endpoint would look like the following example:

```
GET /userinfo HTTP/1.1
Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.eyJleHAiOjE0MTg3MDI0MTIsIm
```

With the following response:

```
HTTP/1.1 200 OK
Date: Tue, 16 Dec 2017 08:00:12 GMT
Access-Control-Allow-Origin: *
Content-Type: application/json;charset=ISO-8859-1
Content-Language: en-US
Content-Length: 333
Connection: close

{
   "sub": "6WZQPpnQxV",
   "iss": "https://idexchange.gov.au"
   "given_name": "Stephen",
   "family_name": "Michaels",
   "birthdate": "1974-02-29"
}
```

UserInfo claims must be returned as members of a JSON object unless a signed or encrypted response was requested during Client Registration.

For privacy reasons, OpenID Providers **MAY** elect to not return values for some of the requested claims; it **SHOULD NOT** present with a null or empty string value.

The `sub` claim **MUST** always be returned in the UserInfo Response.

2.7.8 Request Objects

The Identity Exchange operating as an OP **MUST** accept requests containing a request object signed by the client's private key. The Identity Exchange **MUST** validate the signature on such requests against the Clients public key.

The Identity Exchange **MUST** accept request objects encrypted with the Identity Exchanges Public key.

2.7.9 Authentication Context

The Identity Exchange **MUST** provide an Authentication Context Class Reference (acr), See section 2.8.3 below on valid ACR Claims.

The Identity Exchange **MUST** return the acr value used for the authentication even if the `acr` claim was not marked essential or the `acr_values` parameter was used.

2.8 Entity Information

2.8.1 Claims Supported

Discovery mandates the inclusion of the `claims_supported` field that defines the claims a client **MAY** expect to receive for the supported scopes. Servers **MUST** return claims on a best effort basis. An Identity Exchange asserting that it can provide a user claim however, does not imply that the data is available for all its users. Clients **MUST** be prepared to receive partial data.

Identity Exchanges **MUST** only return claims as described in the *TDIF: Attribute Profile.*

Some attributes will only be available via the UserInfo endpoint. These attributes are noted within *TDIF: Attribute Profile.*

2.8.2 Scope Profiles

The available scope profiles and supported claims are described within the *TDIF: Attribute Profile.*

2.8.3 Valid ACR Claims

Assurance levels are represented using the acr values that are defined in section 2.2.8 of the *TDIF: Attribute Profile*. Required acr values can be represented in an OIDC authentication request using either the `acr_values` parameter or the `acr` claim as described in section 2.6.6.

Where the acr is requested using the acr claim, this `acr` claim **MAY** be marked as essential claim as per the example below:

```
"claims": {
    "id_token": {
       "acr": {
              "essential": true,
              values: ["urn:id.gov.au:tdif:acr:ip2:cl3"]
       }
    }
}
```

When the acr values are marked as an essential claim, the Identity Provider **MUST** return a value that matches the requested values. If the End-User is unable to achieve a level of assurance that matches the request then an authentication error response **MUST** be returned.

The `acr_values` parameter may be used to request the required acr value as per the example below:

```
acr_values=urn:id.gov.au:tdif:acr:ip2:cl3
```

When requesting the acr claim using the `acr_values` parameter it is requested as a voluntary claim i.e. cannot be marked as essential.

When the acr values are not marked as essential, i.e. they are a voluntary claim, the Identity Provider **SHOULD** return the level of assurance that the End-User was able to achieve.

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. The Identity Exchange interprets this as a request for any assurance level that meet or exceeds the requested level. The Identity Exchange will explicitly include all the acr values that will meet the requested minimum in the request it generates to the Identity Provider.

The specification of the `acr` claim within the request object is the preferred method for requesting the `acr`.

Relying Party clients **MUST NOT** specify both the `acr` claim and `acr_values`.

The Relying Party **MUST** determine if the returned acr meets the minimum requirement for the authentication context that was requested.

2.9 Privacy Considerations

Attributes are only be shared in accordance with the Attribute Sharing Policy set out in the *TDIF: Attribute Profile*.

Subject identifiers are always pairwise identifiers that results in a double blind identity federation.

The identity federation must not require or create a single universal identifier that exists between an Identity Provider and Relying Party for a user.

2.10 Security Considerations

All transactions **MUST** be protected by TLS.

All clients **MUST** conform to the applicable recommendations in the Security considerations section of **[RFC 6749]** and those found in the OAuth 2.0 Threat Model and Security Considerations document.

[RFC 8252] for best practice for native apps.

3 Identity Exchange to Identity Provider Profile

This section describes the OpenID Connect Profile for Identity Exchanges to authenticate and receive identity data, as a relying party, from an Identity Provider operated by an Accredited Provider within the TDIF.

Note: an OpenID Provider or OP is an OAuth 2.0 Authorisation Server that is capable of authenticating the End User and providing claims to a Relying Party (RP) about the authentication event and the End User. Any use of OP or authorisation server within this profile can be considered congruous.

Where an Identity Exchange interacts with an Identity Provider as a result of a request from a Relying Party then the Identity Exchange must generate authentication requests to an Identity Provider.

3.1 Client Types

The following profile descriptions give patterns for deployment for use in different types of client applications based on the OAuth grant type. The resource owner password credentials grant type as defined in **[RFC 6749]** is intentionally omitted and **MUST NOT** be used under these profiles.

As the Identity Exchange is acting as a proxy the Full Client with delegation is the only client type available. In this profile the only clients are Identity Exchanges.

These client types are as described in the iGov OAuth 2.0 profiles.

3.1.1 Full Client with User Delegation

This client type applies to clients that act on behalf of a particular resource owner and require delegation of that user's authority to access the protected resource. Furthermore, these clients are capable of interacting with a separate web browser application to facilitate the resource owner's interaction with the authentication endpoint of the authorisation server.

These clients **MUST** use the authorisation code flow of OAuth 2 by sending the resource owner to the authorisation endpoint to obtain authorisation. The user **MUST** authenticate to the authorisation endpoint. The user's web browser is then redirected back to a URI hosted by the client service, from which the client can obtain an authorisation code passed as a query parameter. The client then presents that authorisation code along with its own credentials (`private_key_jwt`) to the authorisation server's token endpoint to obtain an access token.

These clients **MUST** be associated with a unique public key as described in 3.4 below.

This client type **MAY** request and be issued a refresh token if the security parameters of the request allow for it.

3.2 Client Registration

All clients **MUST** register with the authorisation server, each client instance must receive a unique client identifier from the authorisation server.

Client registration **MUST** use a static configuration. There is no support for the dynamic registration of Identity Exchange clients by Identity Providers.

3.3 Redirect URI

As Clients, Identity Exchanges must use the `authorization_code` grant type so **MUST** register their full redirect URIs. The authorisation server **MUST** validate the redirect URI given by the client at the authorisation endpoint using strict string comparison.

A client must protect the values passed back to its redirect URI by ensuring that the redirect URI is:

- Hosted on a website with TLS protection (HTTPS).

Clients **SHOULD NOT** have multiple redirect URIs on different domains.

Clients **MUST** **NOT** forward values passed back to their redirect URIs to other arbitrary or user-provided URIs (i.e. no open redirectors).

3.4 Client Keys

As Clients using the authorisation code grant type, Identity Exchanges **MUST** have a public and private key pair type for use in authentication to the token endpoint. The client **MUST** register their public keys in their client registration metadata by either sending the public key directly in the jwks field or by registering a `jwks_uri` that **MUST** be reachable by the authorisation server. It is recommended that clients use a `jwks_uri` as it allows for easier key rotation.

The `jwks` field or the content available from the `jwks_uri` of a client **MUST** contain a public key in JSON Web Key Set (JWK Set) format. The authorisation server **MUST** validate the content of the clients registered `jwks_uri` document and verify that it contains a JWK Set. The example below is a 2048 bit RSA key.

```
{
   "keys": [
     {
       "alg": "RS256",
       "e": "AQAB",
       "n": "kAMYD62n_f2rUcR4awJX4uccDt0zcXRssq_mDch5-
ifcShx9aTtTVza23P
Tn3KaKrsBXwWcfioXR6zQn5eYdZQVGNBfOR4rxF5i7t3hfb4WkS50EK1gBYk2lO9NSrQ-
xG9QsUsAnN6RHksXqsdOqvnxjLexDfIJlgbcCN9h6TBC66ZXv7PVhl19gIYVifSU7liHk
Le0l0fw7jUI6rHLHf4d96_neR1HrNIK_xssr99Xpv1EM_ubxpktX0T925qej9fMEpzzQ5
HLmcNt1H2_VQ_Ww1JOLn9vRnH48FDj7TxlIT74XdTZgTv31w_GRPAOfyxEw_ZUmxhz5Z-
gTlQ",
       "kty": "RSA",
       "kid": "oauth-client"
     }
   ]
}
```

3.5 Grant Types

The only supported grant type is `authorization_code`.

3.6 Relying Party Profile (Identity Exchange)

3.6.1 Audit Logging

The Identity Exchange **MUST** generate a unique audit Id for an authentication request from a Relying Party as described in step 2. It **MUST** log all the related interactions with Identity Providers using this unique audit id. To enable a traceable audit trail for requests sent to an Identity Provider, an Exchange must implement a scheme to ensure that each request be uniquely identifiable at the Identity Provider.

A recommended scheme is for an Identity Exchange to generate a value for the `state` parameter that is unique. This enables an Identity Provider to maintain the required audit trail without requiring any additional bespoke elements in this profile.

3.6.2 Requests to the Authorization Endpoint (Authentication Requests)

Clients making a request to the authorisation endpoint **MUST** use an unpredictable value for the state parameter with at least 128 bits of entropy. Clients **MUST** validate the value of the state parameter upon return to the redirect URI and **MUST** ensure that the state value is securely tied to the user's current session e.g. by relating the state value to a session identifier issued by the client software to the browser.

Clients **MUST** include their full redirect URIs in the authorisation request. To prevent open redirection and other injection attacks, the authorisation server **MUST** match the entire redirect URI using a direct string comparison against registered values and **MUST** reject requests with invalid or missing redirect URIs.

Request Parameters:

- `client_id.`
    - REQUIRED. OAUTH 2.0 Client Identifier valid at the Authorisation Server.
- `response_type.`
    - REQUIRED. Must be set to code.
- `Scope.`

    - REQUIRED. Indicates the attributes being requested. The `openid` scope **MUST** always be present.
- `redirect_uri.`
    - REQUIRED. Indicates a valid endpoint where the client will receive the authentication response. The URI must match exactly one of the Redirection URIs preregistered at the OP. The redirection URI **SHOULD** use the https scheme.
- `State.`
    - REQUIRED. Un-guessable random string generated by the RP used to protect against CSRF attacks. Must contain a sufficient amount of entropy to avoid guessing and is returned to the RP in the authentication response.
    - This profile recommends that this value also be unique for each request generated by the RP (Identity Exchange).
- `Nonce.`
    - REQUIRED. Un-guessable random string generated by the client, used to protect against CSRF attacks. Must contain sufficient entropy to avoid guessing. Returned to the Client in the ID Token.
- `acr_values.`
    - OPTIONAL.

The values of the `scope` and `acr_values` parameters are mapped from the original request to the Identity Exchange that triggered the generation of this request from the Identity Exchange (acting as the Relying Party) to the Identity Provider. Additional OIDC parameters may also mapped from the original request to the Identity Exchange that triggered this request from the Identity Exchange to the Identity Provider.

A sample request may look like the following example:

```
https://idp.gov.au/oidc/authorization?
  response_type=code
  &client_id=827937609728-m2mvqffo9bsefh4di90saus4n0diar2h
  &scope=profile+email+phone+openid

&redirect_uri=https%3A%2F%2Fidexchange.gov.au%2Foidc%2FloginResponse
  &state=2ca3359dfbfd0
&acr_values=urn%3Aid.gov.au%3Atdif%3Aacr%3Aip3%3Acl2+urn%3Aid.gov.au%
3Atdif%3Aacr%3Aip3%3Acl3+urn%3Aid.gov.au%3Atdif%3Aacr%3Aip4%3Acl3
```

3.6.3 Requests to the Token Endpoint

Requests to the token endpoint require client authentication. The client authentication mechanism is a signed JWT and is defined in section 3.7.2.2.

The claims that are included in the JWT are summarised below:

- `Iss.`
    - The client Id of the client creating the JWT.
- `sub.`
    - The client Id of the client creating the JWT.
- `aud.`
    - The URL of the authorisation server's token endpoint.
- `iat.`
    - The time that the token was created by the client.
- `exp.`
    - The expiration time after which the token must be considered invalid.
- `jti.`
    - A unique, random, identifier generated by the client for this authentication.

The following is an example of the use of the required claims for a client authentication JWT as defined in this profile. Additional claims **MAY** be included in this set.

```
{
   "iss": "55f9f559-2496-49d4-b6c3-351a586b7484",
   "sub": "55f9f559-2496-49d4-b6c3-351a586b7484",
   "aud": "https://idp.gov.au/token",
   "iat": 1418698788,
   "exp": 1418698848,
   "jti": "1418698788/107c4da5194df463e52b56865c5af34e5595"
}
```

The JWT assertion **MUST** be signed by the client using the client's private key. See section 3.4 for the mechanisms by the client can make its public key known to the authorization server.

The following claims **MUST** be included:

- `grant_type`.
    - **MUST** be set to authorization_code.
- `Code`.
    - The value of the code parameter returned in the authorisation response.
- `redirect_uri`.
    - value **MUST** be identical to the value of the `redirect_uri` parameter that was included in the authorisation request as described in section 3.6.2.
- `client_assertion_type`.
    - **MUST** be set to: urn:ietf:params:oauth:client-assertion-type:jwt-bearer.
- `client_assertion`.
    - The value of the signed client authentication JWT generated as described below in the ID Tokens section. The RP must generate a new assertion JWT for each call to the token endpoint.

These would be sent to the token endpoint as shown in the example below:

```
POST /token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: idp.gov.au

grant_type=authorization_code
&code=sedaFh
&redirect_uri=https%3A%2F%2Fidexchange.gov.au%2Foidc%2FloginResponse
&client_id=55f9f559-2496-49d4-b6c3-351a586b7484
&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-
assertion-type%3Ajwt-bearer
&client_assertion=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.ew0KICAgImlzcy
I6ICI1NWY5ZjU1OS0yNDk2LTQ5ZDQtYjZjMy0zNTFhNTg2Yjc0ODQiLA0KICAgInN1YiI
6ICI1NWY5ZjU1OS0yNDk2LTQ5ZDQtYjZjMy0zNTFhNTg2Yjc0ODQiLA0KICAgImF1ZCI6
ICJodHRwczovL2lkcC1wLmV4YW1wbGUuY29tL3Rva2VuIiwNCiAgICJpYXQiOiAxNDE4N
jk4Nzg4LA0KICAgImV4cCI6IDE0MTg2OTg4NDgsDQogICAianRpIjogIjE0MTg2OTg3OD
gvMTA3YzRkYTUxOTRkZjQ2M2U1MmI1Njg2NWM1YWYzNGU1NTk1Ig0KfQ.t_gX8JQGq3G2
OEc2kUCQ8zVj7pqff87Sua5nktLIHj28l5onO5VpsL4sRHIGOvrpo7XO6jgtPWy3iLXv3
-NLyo1TWHbtErQEGpmf7nKiNxVCXlGYJXSDJB6shP3OfvdUc24urPJNUGBEDpt
IgT7Lhf6BbwQNlMQubNeOPRFDqQoLWqe7UxuI06dKX3SEQRMqcxYSIAfP7CQZ4WLuKXb6
oEbaqz6gL4l6p83G7wKGDeLETOTHsztZjKR38v4F_MnSrx8e0iIqgZwurW0RtetEWvynO
CJXk-p166T7qZR45xuCxgOotXY6O3et4n77GtgspMgOEKj3b_WpCiuNEwQ
```

3.6.4 Request to the UserInfo Endpoint

The client may send a UserInfo Request using either a HTTP `GET` or HTTP `POST`. The Access Token obtained from an OpenID Connect Authentication Request **MUST** be sent as a Bearer Token as per section 2 of OAuth Bearer Token Usage.

It is RECOMMENDED that the request use the HTTP `GET` method and the Access Token is sent using the `Authorization` header field.

The following is an example of a request to a UserInfo Endpoint.

```
GET /userinfo HTTP/1.1
  Host: idp.gov.au
  Authorization: Bearer SlAV32hkKG
```

3.6.5 ID Tokens

All clients must validate the signature of an ID Token before accepting it using the public key of the issuing server, published in JSON Web Key (JWK) format. ID Tokens **MAY** be encrypted using the appropriate key of the requesting client.

Clients **MUST** verify the following in received ID tokens:

- `Iss.`
    - The Issuer field is the URL of the expected issuer.
- `Aud.`
    - The audience field contains the client ID of the client.
- `exp`, `iat`, `nbf.`
    - The expiration, issued at and not before tokens are dates (integer number of seconds since 00:00:00Z 1st January 1970, i.e. Unix epoch) are within acceptable ranges.

3.6.6 Request Objects

Clients **MAY** optionally send requests to the Authorization Endpoint using the `request` parameter as defined in **[OpenIDCore].**

Request objects **MUST** be signed by the clients registered key. Request objects **MAY** be encrypted to the Authorisation Server's public key.

3.6.7 Authentication Context

The values for the acr that the Relying Party (Identity Exchange) includes in the request are mapped from the original request from a Relying Party.

3.6.8 Discovery

The Identity Exchange **SHOULD** cache OpenID Provider (OP) metadata once an OP has been discovered and used by the Identity Exchange.

3.7 Identity Provider Profile

3.7.1 Audit/Logging

The Identity Provider **MUST** always log all authentication requests and responses, including the values of `client_id` and the `state` parameters associated with the request.

3.7.2 Connecting to Clients

3.7.2.1 Grant Types

The only supported grant type is `authorization_code`. This grant type is described in section 4.1 of **[RFC 6749]**.

The Authorization Code Flow is the only authentication flow supported by this federation. The Authorization Code Flow returns an Authorization Code to the client that the client can then exchange for an ID Token and an Access Token. This provides the benefit of not exposing any tokens to the User Agent and potentially malicious applications with access to the User Agent.

The authorisation server **MUST** validate all redirect URIs for the `authorization_code` grant type.

3.7.2.2 Client Authentication

The authorisation server **MUST** enforce client authentication to the authorization servers token endpoint using a JWT assertion as defined by using only the `private_key_jwt` method as described in **[OpenIDCore]**. Clients that have registered a public key sign a JWT using the associated private key. The Client authenticates in accordance with JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants and Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants.

The JWT must expire and **SHOULD** have a lifetime no longer than five minutes. Short expiration times are recommended wherever practicable. The following guidance is provided in **[RFC 7523]** with regard to expiration times: the JWT MUST contain an "exp" (expiration time) claim that limits the time window during which the JWT can be used. The authorization server MUST reject any JWT with an expiration time that has passed, subject to allowable clock skew between systems. Note that the authorization server may reject JWTs with an "exp" claim value that is unreasonably far in the future.

The JWT **MUST** contain the following REQUIRED claims and **MAY** contain the following OPTIONAL Claim values:

- `Iss.`
    - REQUIRED. Issuer. This **MUST** contain the `client_id` of the client creating the token.
- `Sub.`
    - REQUIRED. Subject. This **MUST** contain the `client_id` of the client creating the token.
- `Aud.`
    - REQUIRED. Audience. The value that identifies the Authorisation Server as an intended audience. The Authorisation Server **MUST** verify that it is an intended audience for the token. The Audience **SHOULD** be the URL of the Authorisation Server's Token Endpoint.

- `Jti.`
    - REQUIRED. JWT ID. A unique identifier for the token generated by the client, which can be used to prevent reuse of the token. This identifier **MUST** contain at least 128 bits of entropy and **MUST NOT** be re-used by any subsequent authentication token.
- `Exp.`
    - REQUIRED. Expiration time on or after which the ID Token **MUST NOT** be accepted for processing.
- `Iat.`
    - OPTIONAL. Time at which the JWT was issued.

3.7.2.3 Dynamic Registration

Dynamic Registration of clients is not supported.

3.7.2.4 Discovery

The OpenID Connect discovery standard provides a standard pragmatic way for clients to obtain configuration details for communicating with Identity Exchanges and is an important part of building a scalable federation ecosystem.

Exposing a Discovery endpoint does not put the exchange at risk of attack. Endpoints and parameters specified in the Discovery document **SHOULD** be considered public information regardless of the existence of the discovery document.

Access to the Discovery document **MAY** be protected with existing web authentication methods if required by the Identity Exchange. Credentials for the Discovery document are then managed by the Identity Exchange and support for these authentication methods is outside the scope of this specification.

Endpoints described in the Discovery document **MUST** be secured in accordance with this specification and **MAY** have additional controls the Identity Exchange wishes to support.

All OpenID Connect servers are uniquely identified by a URL known as the issuer, this will also be the case for Identity Exchanges. This URL serves as the prefix of a service discovery endpoint as specified in the OpenID Connect Discovery standard.

The discovery document **MUST** as a minimum contain the following fields:

- `Issuer.`
    - The fully qualified issuer URL of the server.
- `authorization_endpoint.`
    - The fully qualified URL of the server's authorisation endpoint defined by **[RFC 6749]**.
- `token_endpoint.`
    - The fully qualified URL of the server's token endpoint defined by **[RFC 6749]**.
- `introspection_endpoint.`
    - The fully qualified URL of `the` server's introspection endpoint defined by the OAuth Token Introspection RFC – **[RFC 7662]**.
- `revocation_endpoint.`
    - The fully qualified URL of the server's revocation endpoint as defined by **[RFC 7009]**.
- `jwks_uri.`
    - The fully qualified URI of the server's public key in JWK Set format as defined in **[RFC 7517]**.
- `scopes_supported.`
    - The list of TDIF scopes as define in *TDIF: Attribute Profile* that the Identity Provider supports.
- `claims_supported`
    - The list of claims available in the supported scopes.

Below is an example JSON document found at the discovery endpoint for an authorisation server.

```
{
  "request_parameter_supported": true,
  "id_token_encryption_alg_values_supported": [
    "RSA-OAEP", "RSA1_5", "RSA-OAEP-256"
  ],
  "registration_endpoint": "https://idexchange.gov.au/register",
  "userinfo_signing_alg_values_supported": [
    "HS256", "HS384", "HS512", "RS256", "RS384", "RS512"
  ],
```

```
"token_endpoint": "https://idexchange.gov.au/token",
"request_uri_parameter_supported": false,
"request_object_encryption_enc_values_supported": [
  "A192CBC-HS384", "A192GCM", "A256CBC+HS512",
  "A128CBC+HS256", "A256CBC-HS512",
  "A128CBC-HS256", "A128GCM", "A256GCM"
],
"token_endpoint_auth_methods_supported": [
  "private_key_jwt",
],
"userinfo_encryption_alg_values_supported": [
  "RSA-OAEP", "RSA1_5",
  "RSA-OAEP-256"
],
"subject_types_supported": [
  "pairwise"
],
"id_token_encryption_enc_values_supported": [
  "A192CBC-HS384", "A192GCM", "A256CBC+HS512",
  "A128CBC+HS256", "A256CBC-HS512", "A128CBC-HS256",
  "A128GCM", "A256GCM"
],
"claims_parameter_supported": false,
"jwks_uri": "https://idexchange.gov.au/jwk",
"id_token_signing_alg_values_supported": [
  "HS256", "HS384", "HS512", "RS256", "RS384", "RS512", "none"
],
"authorization_endpoint": "https://idexchange.gov.au/authorize",
"require_request_uri_registration": false,
"introspection_endpoint": "https://idexchange.gov.au/introspect",
"request_object_encryption_alg_values_supported": [
  "RSA-OAEP", "RSA-OAEP-256"
],
"service_documentation": "https://idexchange.gov.au/about",
"response_types_supported": [
  "code"
],
"token_endpoint_auth_signing_alg_values_supported": [
  "HS256", "HS384", "HS512", "RS256", "RS384", "RS512"
],
"revocation_endpoint": "https://idexchange.gov.au/revoke",
"request_object_signing_alg_values_supported": [
  "HS256", "HS384", "HS512", "RS256", "RS384", "RS512"
],
"claim_types_supported": [
  "normal"
],
"grant_types_supported": [
  "authorization_code",
],
"scopes_supported": [
  "profile", "openid", "email", "phone"
],
"userinfo_endpoint": "https://idexchange.gov.au/userinfo",
"userinfo_encryption_enc_values_supported": [
```

```
    "A192CBC-HS384", "A192GCM", "A256CBC+HS512","A128CBC+HS256",
    "A256CBC-HS512", "A128CBC-HS256", "A128GCM", "A256GCM"
  ],
  "op_tos_uri": "https://idexchange.gov.au/about",
  "issuer": "https://idexchange.gov.au/",
  "op_policy_uri": "https://idexchange.gov.au/about",
  "claims_supported": [
    "sub", "acr","given_name", "family_name", "birthdate", "email",
"phone "
  ]
}
```

3.7.3 Requests to the Authorisation Endpoint (Authentication Request)

The IdP **MUST** support ALL of the mechanisms for requesting a LoA as described in *TDIF: Attribute Profile*.

3.7.4 User Consent

The Identity Exchange is a trusted party that is responsible for ensuring consent is provided by the user in accordance with the Attribute Sharing Policies set out in the *TDIF: Attribute Profile*. An Identity Provider connected to an Identity Exchange is not required to gather user consent for these releasing these attributes to the Relying Party since:

- The Identity Exchange will ensure that the required user consent has been provided;
- The Identity Exchange hides the Relying Party from the Identity Provider so any user consent for the sharing of attributes will not be specific enough to meet the consent requirements stated in the *TDIF: Privacy Requirements.*

3.7.5 Response to Authorisation Requests

The Authorization Response to the Authorization Code flow **MUST** return the following fields in the response:

- `State.`

    - the value of the state parameter passed in via the authentication request. This value **MUST** match exactly.
- `Code`.
    - The authorisation code, a random string issued by the IdP to be used in the request to the token endpoint.

The key requirements for these fields are described in the OAuth 2.0 specification section 4.1.2.

An example response is shown below:

```
https://idexchange.gov.au/web/oidc/loginResponse?
    state=2ca3359dfbfd0
   &code=gOIFJ1hV6Rb1sxUdFhZGACWwR1sMhYbJJcQbVJN0wHA
```

The authentication response is sent via HTTP redirect to the URI specified in the request.

3.7.5.1 Authentication Error Response

The Authentication Error Response is the message returned from the IdPs (OP) Authorisation Endpoint in response to the Authorisation Request sent by the Identity Exchange.

If the End-User denies the request or the End-User authentication fails, the OP informs the RP by using the error responses defined in either section 4.1.2.1 of OAuth 2.0 or the error codes defined in section 3.1.2.6 of the OpenID Connect Core 1.0 specification.

The additional authentication error responses defined by this Profile are:

- `authentication_cancelled`.
    - The end-user did not proceed with the authentication interaction.

3.7.6 ID Tokens

All tokens **MUST** be signed by the issuer IdP's private key. ID Tokens **MAY** be encrypted using the appropriate key of the requesting client.

The ID Token must expire and **SHOULD** have a lifetime no longer than five minutes. Short expiration times are recommended as the ID token is consumed by the client and not presented to remote systems.

The token response includes an access token, which can be used to make a UserInfo request, and ID token (a signed and optionally encrypted JSON Web Token). ID Token values have the following meanings:

- `Iss.`
    - REQUIRED. The issuer field is the URL of the expected issuer.
- `Aud.`
    - REQUIRED: The audience field contains the client ID of the client.
- `Sub.`
    - REQUIRED The identifier of the user. **SHOULD** be a pairwise anonymous identifier, and be unique per client to prevent linkability and traceability between clients.
- `Acr.`
    - OPTIONAL. The level of assurance at which the user was authenticated at. **MUST** be a member of the acr_values list from the authentication request.
- `Nonce.`
    - the nonce value that was provided in the authentication request. **MUST** be included if it was provided in the authentication request.
- `Jti.`
    - REQUIRED. A unique identifier for the token which can be used to prevent the reuse of the token.
- `exp`, `iat`, `nbf.`
    - REQUIRED. The expiration, issued at, and not before timestamps for the tokens. They are dates presented as an integer representing the number of seconds since 1970-01-01T00:00:00Z UTC (Unix epoch) within acceptable ranges.

The following is an example of an ID token signed using the server's RSA key.

```
eyJhbGciOiJSUzI1NiJ9.eyJhdXRoX3RpbWUiOjE0
        MTg2OTg3ODIsImV4cCI6MTQxODY5OTQxMiwic3ViI
```

```
        joiNldaUVBwblF4ViIsIm5vbmNlIjoiMTg4NjM3Yj
        NhZjE0YSIsImF1ZCI6WyJjMWJjODRlNC00N2VlLTR
        iNjQtYmI1Mi01Y2RhNmM4MWY3ODgiXSwiaXNzIjoi
        aHR0cHM6XC9cL2lkcC1wLmV4YW1wbGUuY29tXC8iL
        CJpYXQiOjE0MTg2OTg4MTJ9mQc0rtL56dnJ7_zO_f
        x8-qObsQhXcn-qN-FC3JIDBuNmP8i11LRA_sgh_om
        RRfQAUhZD5qTRPAKbLuCD451lf7ALAUwoGg8zAASI
        5QNGXoBVVn7buxPd2SElbSnHxu0o8ZsUZZwNpircW
        NUlYLje6APJf0kre9ztTj-5J1hRKFbbHodR2I1m5q
        8zQR0ql-FoFlOfPhvfurXxCRGqP1xpvLLBUi0JAw3
        F8hZt_i1RUYWMqLQZV4VU3eVNeIPAD38qD1fxTXGV
        Ed2XDJpmlcxjrWxzJ8fGfJrbsiHCzmCjflhv34O22
        zb0lJpC0d0VScqxXjNTa2-ULyCoehLcezmssg
```

Its claims are as follows:

```
{
        "auth_time": 1418698782,
        "exp": 1418699412,
        "sub": "6WZQPpnQxV",
        "nonce": "188637b3af14a",
        "aud": [
          "c1bc84e4-47ee-4b64-bb52-5cda6c81f788"
        ],
        "iss": "https:\\/\\/idp.gov.au\\/",
        "acr":"urn:id.gov.au:tdif:acr:ip3:cl2",
        "iat": 1418698812,
       "nbf": 1418698812
}
```

3.7.7 UserInfo Endpoint

Identity Providers **MUST** support the UserInfo endpoint for claims as described in the *TDIF: Attribute Profile*. The UserInfo endpoint **MUST** only return claims that are authorised within the authentication request that issued the access token that is being used to access the endpoint.

Support for the UserInfo endpoint is important for maximum client implementation interoperability even if no additional user information is returned. Clients are not required to call the UserInfo endpoint but should not receive an error if they do.

A request to the UserInfo endpoint would look like the following example:

```
GET /userinfo HTTP/1.1
Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.eyJleHAiOjE0MTg3MDI0MTIsIm
```

With the following response:

```
HTTP/1.1 200 OK
Date: Tue, 16 Dec 2017 08:00:12 GMT
Access-Control-Allow-Origin: *
Content-Type: application/json;charset=ISO-8859-1
Content-Language: en-US
Content-Length: 333
Connection: close

{
   "sub": "6WZQPpnQxV",
   "iss": "https://idp.gov.au"
   "given_name": "Stephen",
   "family_name": "Michaels",
   "birthdate": "1974-02-29",
   "email": "mailto:s.micheals@qmail.com"
}
```

UserInfo claims must be returned as members of a JSON object unless a signed or encrypted response was requested during Client Registration.

For privacy reasons, OpenID Providers **MAY** elect to not return values for some of the requested claims; it **SHOULD NOT** present with a null or empty string value.

The `sub` claim **MUST** always be returned in the UserInfo Response.

3.7.8 Request Objects

The Identity Provider **MUST** accept requests containing a request object signed by the Identity Exchange's private key. The Identity Provider **MUST** validate the signature on such requests against the Identity Exchange's public key.

The Identity Provider **MUST** accept request objects encrypted with the Identity Providers Public key.

3.7.9 Authentication Context

The IdP must provide the `acr` claim in ID Tokens as described in section 3.8.3.

The IdP **MUST** return the acr value used for the authentication even if the `acr` claim was not marked as essential or the `acr_values` parameter was used.

3.8 Entity Information

The availability, quality and reliability of an individual's identity attributes will vary across Identity Providers depending on the Level of Assurance used to register the identity and the identity information provided as part of the registration process.

The following recommendations set client expectations on the type of data they may acquire.

3.8.1 Claims Supported

Discovery mandates the inclusion of the `claims_supported` field that defines the claims a client **MAY** expect to receive for the supported scopes. Servers **MUST** return claims on a best effort basis. An Identity Exchange asserting that it can provide a user claim however, does not imply that the data is available for all its users. Clients **MUST** be prepared to receive partial data.

Identity Exchanges **MAY** return claims outside of the `claims_supported` list but **MUST** ensure that they do not violate the privacy policies set out by the federation.

Some attributes will only be available via the UserInfo endpoint. These attributes are noted within the *TDIF: Attribute Profile*.

3.8.2 Scope Profiles

The available scope profiles and supported claims are described within the *TDIF: Attribute Profile*.

3.8.3 Valid ACR Claims

Assurance levels are represented using the acr values that defined in section 2.2.8 of the *TDIF: Attribute Profile*. Required acr values can be represented in an OIDC authentication request using either the `acr_values` parameter or the `acr` claim as described in 2.6.6.

The Identity Exchange (acting as the Relying Party in this profile) **MUST** request the full set of acr values that will meets the original Relying Party's minimum assurance requirements. For example:

```
"claims": {
    "id_token": {
       "acr": {
              "essential": true,
              values: ["urn:id.gov.au:tdif:acr:ip3:cl2",
                       "urn:id.gov.au:tdif:acr:ip3:cl3",
                       "urn:id.gov.au:tdif:acr:ip4:cl3"
              ]
       }
    }
}
```

When the acr values are marked as an essential claim, the Identity Provider **MUST** return a value that matches the requested values. If the End-User is unable to achieve a level of assurance that matches the request then an authentication error response **MUST** be returned.

Where the `acr_values` parameter is used a space separated set of acr strings must be provided. The Authentication Context Class satisfied by the authentication performed is returned as the `acr` Claim Value.

```
acr_values=urn:id.gov.au:tdif:acr:ip3:cl2
urn:id.gov.au:tdif:acr:ip3:cl3 urn:id.gov.au:tdif:acr:ip4:cl3
```

When requesting the `acr` claim using this parameter it is requested as a voluntary claim i.e. cannot be marked as essential.

When the acr values are not marked as essential, i.e. they are a voluntary claim, the Identity Provider **SHOULD** return the level of assurance that the End-User was able to achieve.

The specification of the `acr` claim within the request object is the preferred method for requesting the `acr`.

The Identity Exchange **MUST NOT** specify both the `acr` claim and `acr_values`.

The Identity Exchange **MUST** determine if the returned acr meets the minimum requirement for the authentication context that was requested.

3.9 Privacy Considerations

Attributes are only be shared in accordance with the Attribute Sharing Policy set out in the *TDIF: Attribute Profile*. Data minimisation is an essential concept that underpins the Trust Federation. This is an important consideration in the design, for example, ensuring that only the minimum attribute set required to service the authentication request from a Relying Party is returned to the Exchange from an Identity Provider.

Subject identifiers are always pairwise identifiers that results in a double blind identity federation.

The identity federation must not require or create a single universal identifier that exists between an Identity Provider and Relying Party for a user.

3.10 Security Considerations

All transactions **MUST** be protected by TLS.

All clients **MUST** conform to the applicable recommendations in the Security considerations section of **[RFC 6749]** the those found in the OAuth 2.0 Threat Model and Security Considerations document **[RFC 6819]**.

4 Acknowledgements

The authors of this document acknowledge the work of the International Government Assurance Profile (iGov) Working Group, see http://openid.net/wg/igov/, operated by the OpenID Foundation. This profile is based on the draft specifications produced by this working group.

A.1 Appendix A Interactions

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

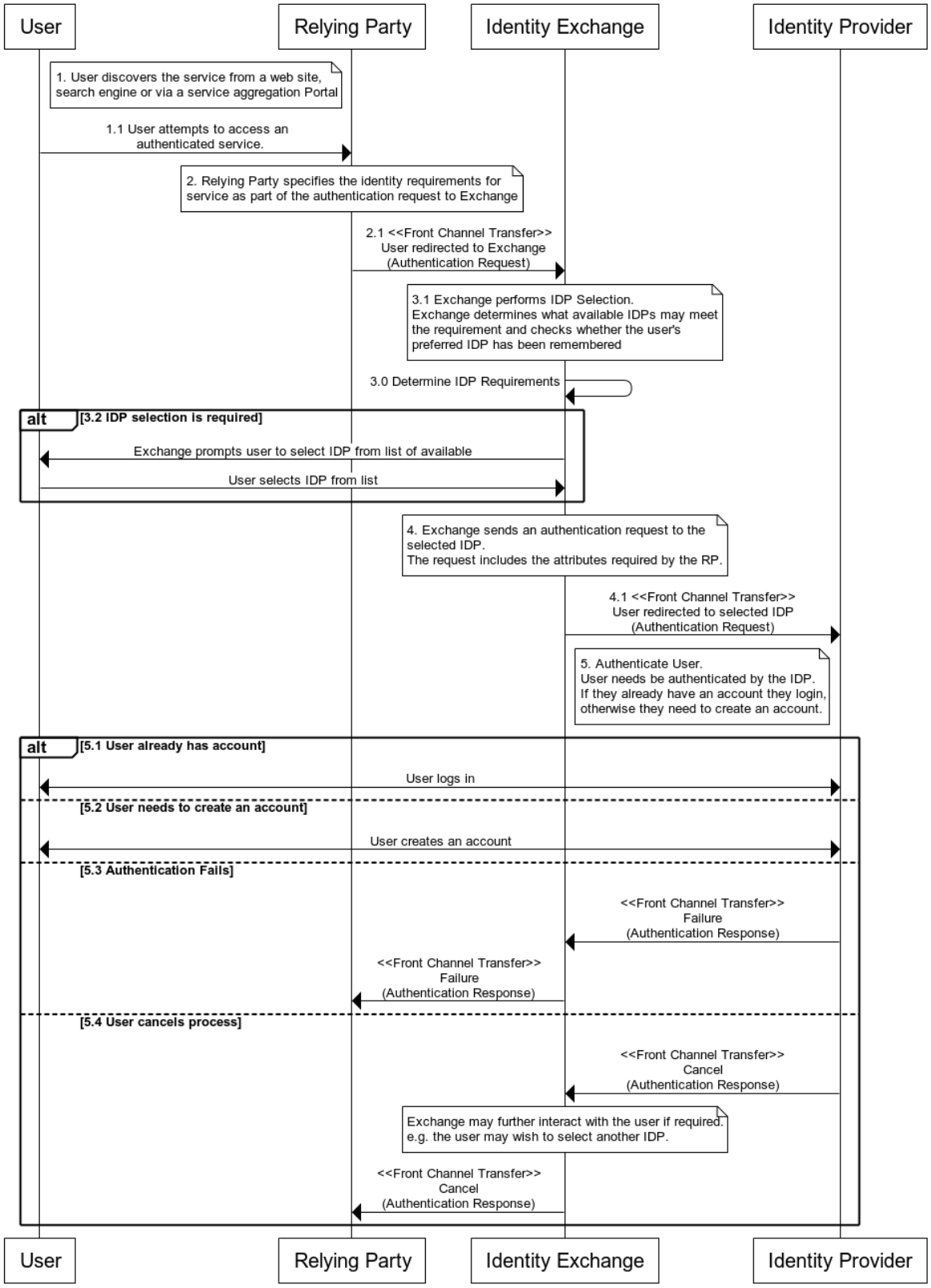

Figure 2: *User Authentication Sequence Diagrams (steps 6 to 11).*

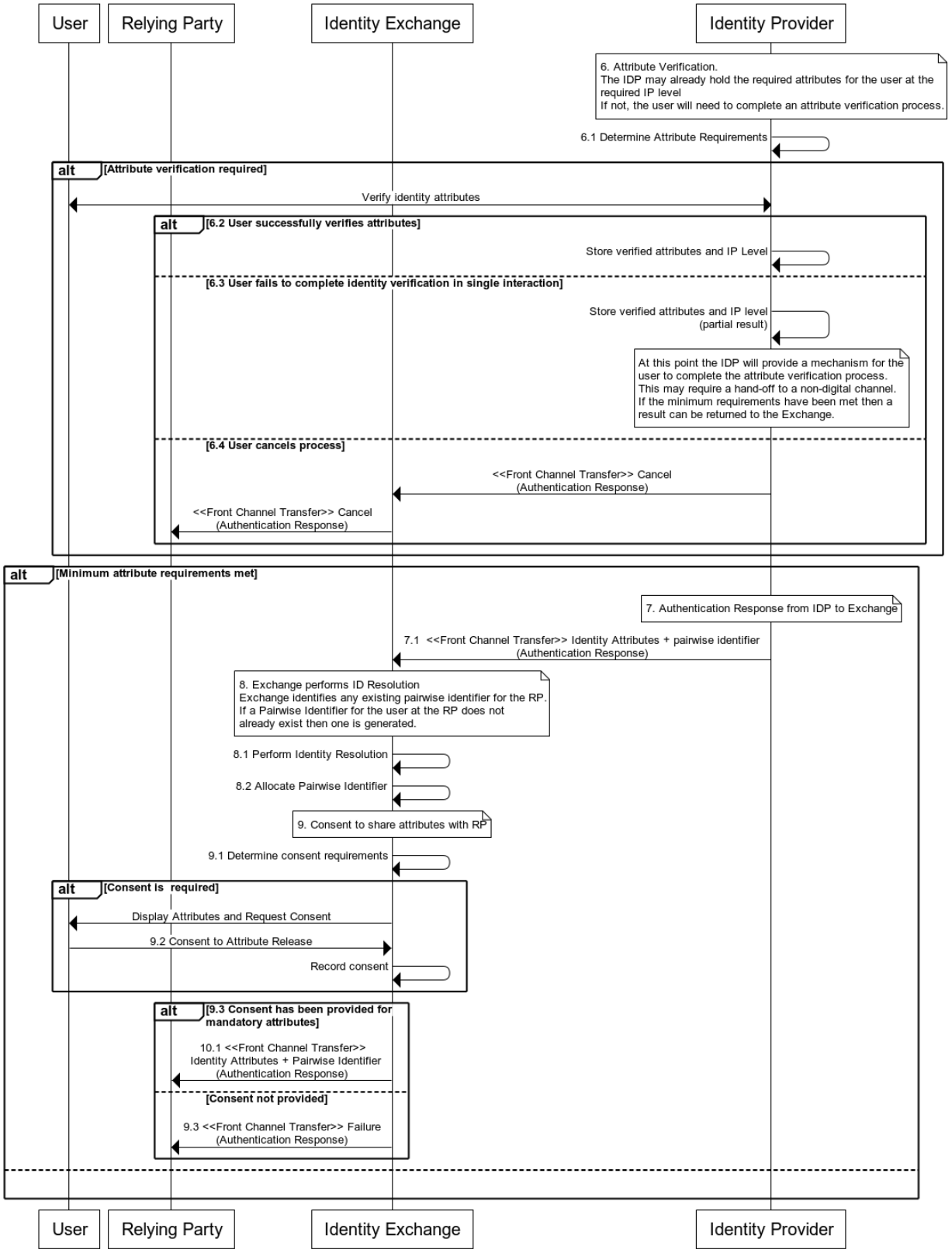

Figure 1 and ***Figure 2*** are sequence diagrams that show the sequence of logical interactions for the authentication of a user. These interactions are intended to illustrate the application of the OIDC profiles described in this document to an end-to-end user experience. Where is the user is transferred 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.

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 can 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 could 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 are 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.[2]

[2] The *TDIF: Attribute Profile* does not currently specify any attributes that may be requested as Mandatory by a Relying Party. In general, an authentication response should always be returned to the Relying Party as per the point above. This is consistent with the requests for claims in the OIDC standard, see https://openid.net/specs/openid-connect-core-1_0.html section 5.5.1.

Step 2 uses the Relying Party to Identity Exchange Profile:

- An OIDC Authentication Request from Relying Party is sent to the authorization endpoint at the Identity Exchange. The Request includes:
- scopes that specify required identity attributes.
- acr 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.

Step 3 uses the Relying Party to Identity Exchange Profile:

- User Cancels Process: Identity Exchange responds with error code value authentication_cancelled.

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.

- An OIDC Authentication Request is sent to authorization endpoint at the Identity Provider. The request includes:
- The scopes that are required to service the request Relying Party request.

- The set of acr values that meet or exceed the acr 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 Identity Exchange. The Identity 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 error code access_denied.
- User Cancels Process: IDP responds with error code value authentication_cancelled.

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

- Authentication Fails: Identity Exchange responds with error code access_denied.
- User Cancels Process: Identity Exchange responds with error code value authentication_cancelled.

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 error code authentication_cancelled.

- 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 acr level.
   - A pairwise identifier for the user at the Identity Provider.
   - Identity attributes.

Step 7 uses the Identity Exchange to Identity Provider Profile.

- An authorisation code is returned to the Identity Exchange client via a front-channel redirect. The Identity Exchange client then sends a token request including authorization code sent to the Identity Provider's Token Endpoint via a back-channel web api call. The Identity Exchange client is authenticated to the Identity Provider's Token Endpoint using a JWT signed with the Identity Exchange's private key. The response to the token request includes an ID Token (signed JWT) containing identity attributes, and an Access Token.
- The Identity Exchange verifies the ID Token using the public key for the Identity Provider. The Identity Exchange may use the Access Token to retrieve additional attributes from the Identity Provider's UserInfo Endpoint if required.

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 ongoing consent for sharing the attributes with the Relying Party.

9.2. Consent to Attribute Release.

- If user consent is required, the Identity Exchange must ensure that user consent has been the provided to the release the attributes to the Relying Party. The Identity Exchange may need to initiate an interaction with the user to gather the required consent. The Identity Exchange will record the provided consent and the user's preference for remembering this consent.

9.3. Consent not provided.

- If user consent is not provided then these attributes are not returned in the authentication response to the Relying Party.
- 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 error code access_denied.

10. Authentication Response to Relying Party.

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

- The Response includes:
  - Achieved acr 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.

- An authorisation code is returned to the Relying Party client via a front-channel redirect. The Relying Party client then sends a token request including the authorization code sent to Identity Exchange's Token Endpoint via a back-channel web api call. The Relying Party client is authenticated to the Identity Exchange Token Endpoint using a JWT signed with the Relying Party's private key. The response to the token request includes an ID Token (signed JWT) containing identity attributes, and an Access Token.
- The Relying Party verifies the ID Token using the public key for the Identity Exchange. The Relying Party may use the Access Token to retrieve additional attributes from the Identity Exchange's UserInfo Endpoint if required.

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 a Relying Party, 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. Where a Relying Party performs Identity Matching, the Relying Party is responsible for ensuring that the matching process is sufficient to manage risks of authorised access to a person's record and is accountable for any privacy breach that may occur as a result of improper matching. 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.

A.2 Appendix B iGov Profile Comparison

This appendix summarises the key differences and similarities between the iGov Profile and the *TDIF: Open ID Connect 1.0 Profile*.

This profile is interoperable with the iGov profile and can be considered a subset of the iGov profile in that is does not mandate all the features currently specified in the iGov profile. The key differences between the iGov Profile and this profile are:

- This profile is supports a brokered identity federation that implements a double-blind federation.
- This profile use the attributes and assurances levels that are specifically required to support the TDIF.

A.2.1 Relying Party to Exchange OIDC Profile

A.2.1.1 Overview

In this OIDC profile, an Exchange is an Open ID Provider (OP), a Relying Party is a Relying Party (RP).

A.2.1.1.1 Grant Types

As per iGov Profile, the RP must use the `authorization_code` grant type.

A.2.1.1.2 Client Types

As per iGov Profile:

- Full Client with User Delegation as defined by the iGov OAuth 2 profile.
- Native Client with User Delegation as defined by the iGov OAuth 2 profile.

A.2.1.2 Relying Party Profile

A.2.1.2.1 Requests to the Authorisation Endpoint

As per iGov Profile, with the following exceptions:

- The `vtr` request parameter is not used.
- No mandatory value for the prompt request parameter is specified. The iGov specification mandates the value `select_account` be used.

A Relying Party may specify a required LoA using the mechanism described in section 2.6.6 of this profile.

A.2.1.2.2 Requests to the Token Endpoint

As per iGov Profile:

- Requests to the token endpoint require client authentication.
- The client authentication mechanism is JWT signed using the client's private key.

A.2.1.2.3 ID Tokens

As per iGov Profile.

- The client must the verify the signature on the ID Token using the public key of the Identity Exchange.

A.2.1.3 OpenID Provider Profile (Identity Exchange)

A.2.1.3.1 Requests to the Authorisation Endpoint

The Exchange **MUST** support ALL the mechanisms for requesting a LoA described in Levels of Assurance.

A.2.1.3.2 ID Tokens

ID Tokens are as described in the iGov specification, with the following clarifications:

- ID Tokens are signed using the private key of the issuer, Exchange.
- `vot`, `vtm` claims are not used.

A.2.1.3.3 UserInfo Endpoint

Claims **MUST** be made available via the UserInfo endpoint as described in the *TDIF: Attribute Profile*. The UserInfo endpoint must only return claims that authorized in the authentication request that issued the access token that is being used to access the UserInfo endpoint.

A.2.1.3.4 Vectors of Trust

Vectors of Trust are not used.

A.2.1.3.5 Authentication Context

An Exchange **MUST** provide the `acr` claim in ID Tokens.

A.2.1.3.6 Discovery

As per iGov OIDC profile except that the `vot` claim is not used.

A.2.1.3.7 Dynamic Registration

Dynamic registration of clients **MAY** be supported by an Exchange.

A.2.1.3.8 Pairwise Identifiers

The TDIF profiles mandate the use of pairwise Subject Identifiers.

A.2.2 Identity Exchange to IDP OIDC Profile

A.2.2.1 Overview

In this OIDC profile, an IDP is an Open ID Provider (OP), an Exchange is a Relying Party (RP).

A.2.2.1.1 Client Types

The only supported client type is a Full Client with User Delegation as defined by the iGov OAuth 2 profile.

A.2.2.1.2 Grant Types

As per iGov Profile. RP must use the authorization_code grant type.

A.2.2.2 Relying Party Profile (Identity Exchange)

A.2.2.2.1 Requests to the Authorisation Endpoint

As per iGov Profile, with the following exceptions:

- The `vtr` request parameter is not used.
- No mandatory value for the prompt request parameter is specified. The iGov specification mandates the value `select_account` be used.

A Relying Party (Identity Exchange) may specify a required LoA using the mechanism described in section 2.6.6 of this profile.

A.2.2.2.2 Requests to the Token Endpoint

As per iGov Profile.

- Requests to the token endpoint require client authentication.
- The client authentication mechanism is JWT signed using the client's private key.

A.2.2.2.3 ID Tokens

As per iGov Profile.

- The client must the verify the signature on the ID Token using the public key of the Identity Provider.

A.2.2.3 OpenID Provider Profile (Identity Provider)

A.2.2.3.1 Requests to the Authorisation Endpoint

The Exchange **MUST** support ALL the mechanisms for requesting a LoA described in Levels of Assurance.

A.2.2.3.2 Authentication Error Responses

- In addition to the standard OIDC authentication error responses, this profile also defines additional error codes in section 3.7.4.

A.2.2.3.3 ID Tokens

ID Tokens are as described in the iGov specification, with the following clarifications:

- ID Tokens are signed using the private key of the issuer, the Identity Provider.
- `vot`, `vtm` claims are not used.

A.2.2.3.4 UserInfo Endpoint

Claims **MUST** be made available via the UserInfo endpoint as described in the *TDIF: Attribute Profile*. The UserInfo endpoint must only return claims that authorized in the authentication request that issued the access token that is being used to access the UserInfo endpoint.

A.2.2.3.5 Vectors of Trust

Vectors of Trust are not used.

A.2.2.3.6 Authentication Context

An Identity Provider **MUST** provide the `acr` claim in ID Tokens.

A.2.2.3.7 Discovery

As per iGov OIDC profile except that the `vot` claim is not used.

A.2.2.3.8 Dynamic Registration

Dynamic registration of an Identity Exchange is not supported.

A.2.2.3.9 Pairwise Identifiers

The TDIF profiles mandate the use of pairwise Subject Identifiers.

A.3 Appendix C – Worked Examples

The following examples show successful authentication interactions between a Relying Party (RP) and an Identity Provider (IdP) via an Identity Exchange. Examples are provided for the following types of applications:

- Web Application (with dedicated server-side component). The client is the web applications back-end that is a confidential client. This example illustrates the use of the OIDC authorisation code flow.
- Native Application. The client is a public client that is installed application. This example illustrates the use of PKCE in conjunction with OIDC authorisation code flow.

A.3.1 Web Application Example

1. The Relying Party Client constructs the Authentication Request and sends it to the Identity Exchange Authorization Endpoint using HTTPS.

   The Authentication Request includes the scope parameter to specify the required identity claims, and an acr value to specify the required level of assurance.

   The following is a non-normative example HTTP 302 redirect response by the Client, which triggers the User Agent to make an Authentication Request to the Authorization Endpoint (with line wraps within values for display purposes only):

```
HTTP/1.1 302 Found
  Location: https://idexchange.gov.au/authorize?
    response_type=code
    &scope=openid%20profile%20email%20phone
    &client_id=s6BhdRkqt3
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &acr_values=urn%3Aid.gov.au%3Atdif%3Aacr%3Aip3%3Acl2
```

   The following is the non-normative example request that would be sent by the User Agent to the Authorization Server in response to the HTTP 302 redirect response by the Client above (with line wraps within values for display purposes only):

```
  GET /authorize?
    response_type=code
```

```
    &scope=openid%20profile%20email%20phone
    &client_id=s6BhdRkqt3
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &acr_values=urn%3Aid.gov.au%3Atdif%3Aacr%3Aip3%3Acl2 HTTP/1.1
  Host: idexchange.gov.au
```

2. The Identity Exchange logs the request from the Relying Party validates and generates a unique audit id for the request (tdif_audit_id), all subsequent actions in the Identity Exchange are logged using this identifier. The Identity Exchange validates the Authentication Request from the Relying Party.

3. 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.

4. The Identity Exchange constructs an Authentication Request and sends it to the Authorization Endpoint of End-User's selected Identity Provider using HTTPS. In this request, the Identity Exchange is now acting as a Relying Party Client.

   The following is a non-normative example HTTP 302 redirect response by the Client, which triggers the User Agent to make an Authentication Request to the Authorization Endpoint (with line wraps within values for display purposes only):

```
HTTP/1.1 302 Found
  Location: https://idp.gov.au/authorize?
    response_type=code
    &scope=openid%20tdif_core%20email%20phone
    &client_id=m6BhdRkqt9
    &state=xf5ifjsldkj
    &redirect_uri=https%3A%2F%2Fidexchange.gov.au%2Fcb
    &acr_values=urn%3Aid.gov.au%3Atdif%3Aacr%3Aip3%3Acl2%20
urn%3Aid.gov.au%3Atdif%3Aacr%3Aip3%3Acl3%20
urn%3Aid.gov.au%3Atdif%3Aacr%3Aip4%3Acl3
```

   The following is the non-normative example request that would be sent by the User Agent to the Authorization Server in response to the HTTP 302 redirect response by the Client above (with line wraps within values for display purposes only):

```
GET /authorize?
    response_type=code
    &scope=openid%20tdif_core%20email%20phone
    &client_id=m6BhdRkqt9
    &state=xf5ifjsldkj
```

```
    &redirect_uri=https%3A%2F%2Fidexchange.gov.au%2Fcb
    &acr_values=urn%3Aid.gov.au%3Atdif%3Aacr%3Aip3%3Acl2%20
urn%3Aid.gov.au%3Atdif%3Aacr%3Aip3%3Acl3%20
urn%3Aid.gov.au%3Atdif%3Aacr%3Aip4%3Acl3 HTTP/1.1
  Host: idp.gov.au
```

5. The Identity Provider validates the Authentication Request from the Identity Exchange.

6. The Identity Provider logs in the End-User or verifies whether the End-User is logged in, depending on the request parameters in the request. The Identity Provider may require additional interactions with the End-User in order to meet the level of assurance specified by the `acr` value in the request.

7. The Authentication Response is returned to the Identity Exchange Client. The Authorization Server issues a `code` adding the following query parameters to the query component of the Identity Exchange Client's registered Redirection URI using the `application/x-www-form-urlencoded` format. For example (with line wraps within values for display purposes only):

```
HTTP/1.1 302 Found
  Location: https://idexchange.gov.au/cb?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=xf5ifjsldkj
```

8. The Identity Exchange validates the Authentication Response from the Identity Provider.

9. The Identity Exchange makes a Token Request to the Identity Provider to exchange the Authorization Code for an ID Token and Access Token. The Identity Exchange includes a signed JWT Bearer Token.

```
POST /token HTTP/1.1
    Host: idp.gov.au
    Content-Type: application/x-www-form-urlencoded

    grant_type=authorization_code&
    code=SplxlOBeZQQYbYS6WxSbIA&
    redirect_uri=https%3A%2F%2Fidexchange.gov.au%2Fcb&
    client_id=m6BhdRkqt9&
    client_assertion_type=
    urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer&
    client_assertion=PHNhbWxwOl ... ZT
```

10. The Identity Provider validates the Token Request from the Identity Exchange. The Identity Provider authenticates the Identity Exchange Client by validating the signature on the JWT using the Identity Exchange Client's registered public key.

11. The Identity Provider return a successful Token Response to the Identity Exchange. The response includes an ID token, Access Token, and may include a Refresh Token. The ID token is a JWT that is signed by the Identity Provider. The ID Token includes a unique `sub` identifier (a pairwise identifier) for the End-User at the Identity Provider, the requested claims (attributes), and the `acr` (level of assurance) for the authentication.

    For example (with line wraps within values for display purposes only):

```
HTTP/1.1 200 OK
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache

  {
   "access_token": "SlAV32hkKG",
   "token_type": "Bearer",
   "refresh_token": "8xLOxBtZp8",
   "expires_in": 3600,
   "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
     yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
     NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZ
     fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
     AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
     Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
     NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
     QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
     K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
     XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
  }
```

12. The Identity Exchange validates the Token Response from the Identity Provider.

13. The Identity Exchange validates the ID Token. The Identity Exchanges validates the signature on the ID Token using the public key for the Identity Provider.

14. The Identity Exchange extracts the subject identifier from the ID Token. The Identity Exchange determines if it already has a pairwise identifier for the subject for the Relying Party that initiated the Authentication Request in step 1. If a pairwise identifier does not exist, Identity Exchange creates a pairwise identifier for the subject for the Relying Party.

15. The Identity Exchange extract the claims from the ID Token. The Identity Exchange gets consent from the End-User to share attributes with the Relying Part in accordance for the policy requirements for these attributes.

16. The Authentication Response is returned to the Relying Client. The Authorization Server issues a `code` adding the following query parameters to the query component of the Identity Exchange Client's registered Redirection URI using the `application/x-www-form-urlencoded` format. For example (with line wraps within values for display purposes only):

```
HTTP/1.1 302 Found
  Location: https://client.example.org/cb?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj
```

17. The Relying Party validates the Authentication Response from the Identity Exchange.

18. The Relying Party makes a Token Request to the Identity Exchange to exchange the Authorization Code for an ID Token and Access Token. The Relying Party includes a signed JWT Bearer Token.

```
POST /token HTTP/1.1
    Host: idexchange.gov.au
    Content-Type: application/x-www-form-urlencoded

    grant_type=authorization_code&
    code=SplxlOBeZQQYbYS6WxSbIA&
    redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb&
    client_id=s6BhdRkqt3&
    client_assertion_type=
    urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer&
    client_assertion=PHNhbWxwOl ... ZT
```

19. The Identity Exchange returns a successful Token Response to the Relying Party. The response includes an ID token, Access Token, and may include a Refresh Token. The ID token is a JWT that is signed by the Identity Exchange. The ID Token includes a unique `sub` identifier (a pairwise identifier) for the End-User at the Relying Party, the requested claims (attributes), and the requested `acr` (level of assurance) for the authentication.

    For example (with line wraps within values for display purposes only):

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
 "access_token": "SlAV32hkKG",
 "token_type": "Bearer",
 "refresh_token": "8xLOxBtZp8",
 "expires_in": 3600,
 "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
   yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
   NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZ
   fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
   AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
   Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
   NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
   QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
   K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
   XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
}
```

20. The Relying Party validates the Token Response from the Identity Exchange.

21. The Relying Party validates the ID Token. The Relying Party validates the signature on the ID Token using the public key for the Identity Exchange.

22. The Relying Party extracts the subject identifier (`sub`), from the ID Token, level of assurance (`acr`), and identity claims from the ID Token.

A.3.1.1 Additional Notes on Examples

- The OIDC interactions for the following steps are specified in the Relying Party to Identity Exchange Profile:
    - Steps 1,2 and 16 to 22.
- The OIDC interactions for the following steps are specified in the Identity Exchange to Identity Provider Profile:
    - Steps 4 to 15.
- The example assumes all required attributes are included in the ID Tokens. The client may retrieve additional attributes using the relevant UserInfo Endpoint.

A.3.2 Native Application Example

1. The Relying Party Client initialises the authentication process by generating a PKCE code verifier and code challenge. The code verifier is a 32 Byte high entropy cryptographic random string. The code challenge is a SHA256 hash of the code verifier.

   For example:

```
code_verifier=LuHyDyxbDiGJsZVsoPdlyPnUV1dhI7jSXL4BcMjt98g
code_challenge=gvOOe2Mnroq78ABp085BsstZYOIH17I1hlQvsXA5pnw
```

2. The Relying Party Client constructs the Authentication Request and sends it to the Identity Exchange Authorization Endpoint using HTTPS.

   The Authentication Request includes the `scope` parameter to specify the required identity claims, and an `acr` value to specify the required level of assurance. The request also includes the `code_challenge` generated in step 1, and the `code_challenge_method` which will always be `S256` (SHA256).

   The following is the non-normative example request that would be sent by the User Agent to the Authorization Server (with line wraps within values for display purposes only). The native application should use the system browser for the platform that it is installed on.

```
GET /authorize?
  response_type=code
  &scope=openid%20profile%20email%20phone
  &client_id=s6BhdRkqt3
  &state=af0ifjsldkj
  &code_challenge=gvOOe2Mnroq78ABp085BsstZYOIH17I1hlQvsXA5pnw
  &code_challenge_method=S256
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb HTTP/1.1
Host: idexchange.gov.au
```

3. The Identity Exchange logs the request from the Relying Party validates and generates a unique audit id for the request (`tdif_audit_id`), all subsequent actions in the Identity Exchange are logged using this identifier. The Identity Exchange validates the Authentication Request from the Relying Party and stores the code challenge locally.

4. 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.

5. The Identity Exchange constructs an Authentication Request and sends it to the Authorization Endpoint of End-User's selected Identity Provider using HTTPS. In this request, the Identity Exchange is now acting as a Relying Party Client.

   The following is a non-normative example HTTP 302 redirect response by the Client, which triggers the User Agent to make an Authentication Request to the Authorization Endpoint (with line wraps within values for display purposes only):

```
HTTP/1.1 302 Found
  Location: https://idp.gov.au/authorize?
    response_type=code
    &scope=openid%20tdif_core%20email%20phone
    &client_id=m6BhdRkqt9
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fidexchange.gov.au%2Fcb
    &acr_values=urn%3Aidp.gov.au%3Atdif%3Aacr%3Aip3%3Acl2%20
urn%3Aid.gov.au%3Atdif%3Aacr%3Aip3%3Acl3%20
urn%3Aid.gov.au%3Atdif%3Aacr%3Aip4%3Acl3
```

   The following is the non-normative example request that would be sent by the User Agent to the Authorization Server in response to the HTTP 302 redirect response by the Client above (with line wraps within values for display purposes only):

```
GET /authorize?
    response_type=code
    &scope=openid%20tdif_core%20email%20phone
    &client_id=m6BhdRkqt9
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fidexchange.gov.au%2Fcb
    &acr_values=urn%3Aid.gov.au%3Atdif%3Aacr%3Aip3%3Acl2%20
urn%3Aid.gov.au%3Atdif%3Aacr%3Aip3%3Acl3%20
urn%3Aid.gov.au%3Atdif%3Aacr%3Aip4%3Acl3 HTTP/1.1
  Host: idp.gov.au
```

6. The Identity Provider validates the Authentication Request from the Identity Exchange.

7. The Identity Provider logs in the End-User or verifies whether the End-User is logged in, depending on the request parameters in the request. The Identity

Provider may require additional interactions with the End-User in order to meet the level of assurance specified by the `acr` value in the request.

8. The Authentication Response is returned to the Identity Exchange Client. The Authorization Server issues a `code` adding the following query parameters to the query component of the Identity Exchange Client's registered Redirection URI using the `application/x-www-form-urlencoded` format. For example (with line wraps within values for display purposes only):

```
HTTP/1.1 302 Found
  Location: https://idexchange.gov.au/cb?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj
```

9. The Identity Exchange validates the Authentication Response from the Identity Provider.

10. The Identity Exchange makes a Token Request to the Identity Provider to exchange the Authorization Code for an ID Token and Access Token. The Identity Exchange includes a signed JWT Bearer Token.

```
POST /token HTTP/1.1
    Host: idp.gov.au
    Content-Type: application/x-www-form-urlencoded

    grant_type=authorization_code&
    code=SplxlOBeZQQYbYS6WxSbIA&
    redirect_uri=https%3A%2F%2Fidexchange.gov.au%2Fcb&
    client_id=m6BhdRkqt9&
    client_assertion_type=
    urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer&
    client_assertion=PHNhbWxwOl ... ZT
```

11. The Identity Provider validates the Token Request from the Identity Exchange. The Identity Provider authenticates the Identity Exchange Client by validating the signature on the JWT using the Identity Exchange Client's registered public key.

12. The Identity Provider return a successful Token Response to the Identity Exchange. The response includes an ID token, Access Token, and may include a Refresh Token. The ID token is a JWT that is signed by the Identity Provider. The ID Token includes a unique `sub` identifier (a pairwise identifier) for the End-User at the Identity Provider, the requested claims (attributes), and the `acr` (level of assurance) for the authentication.

For example (with line wraps within values for display purposes only):

```
HTTP/1.1 200 OK
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache

  {
   "access_token": "SlAV32hkKG",
   "token_type": "Bearer",
   "refresh_token": "8xLOxBtZp8",
   "expires_in": 3600,
   "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
     yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
     NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZ
     fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
     AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
     Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
     NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
     QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
     K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
     XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
  }
```

13. The Identity Exchange validates the Token Response from the Identity Provider.

14. The Identity Exchange validates the ID Token. The Identity Exchange validates the signature on the ID Token using the public key for the Identity Provider.

15. The Identity Exchange extracts the subject identifier from the ID Token. The Identity Exchange determines if it already has a pairwise identifier for the subject for the Relying Party that initiated the Authentication Request in step 1. If a pairwise identifier does not exist, Identity Exchange creates a pairwise identifier for the subject for the Relying Party.

16. The Identity Exchange extract the claims from the ID Token. The Identity Exchange gets consent from the End-User to share attributes with the Relying Part in accordance for the policy requirements for these attributes.

    The Authentication Response is returned to the Relying Client. The Authorization Server issues a `code` adding the following query parameters to the query component of the Identity Exchange Client's registered Redirection URI using the `application/x-www-form-urlencoded format`. For example (with line wraps within values for display purposes only):

```
HTTP/1.1 302 Found
  Location: https://client.example.org/cb?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj
```

17. The Relying Party validates the Authentication Response from the Identity Exchange.

18. The Relying Party makes a Token Request to the Identity Exchange to exchange the Authorization Code for an ID Token and Access Token. The Relying Party includes the code verifier generated in step 1 in the request.

```
POST /token HTTP/1.1
  Host: idexchange.gov.au
  Content-Type: application/x-www-form-urlencoded

    code_verifier=LuHyDyxbDiGJsZVsoPdlyPnUV1dhI7jSXL4BcMjt98g&
    client_id=s6BhdRkqt3&
    grant_type=authorization_code&
    code=SplxlOBeZQQYbYS6WxSbIA&
    redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

19. The Identity Exchange validates the code verifier against the code challenge it received earlier as part of the authentication request (step 3), and if successful returns a successful Token Response to the Relying Party. The response includes an ID token, Access Token, and may include a Refresh Token. The ID token is a JWT that is signed by the Identity Exchange. The ID Token includes a unique sub identifier (a pairwise identifier) for the End-User at the Relying Party, the requested claims (attributes), the acr (level of assurance) for the authentication and the tdif_audit_id.

    For example (with line wraps within values for display purposes only):

```
HTTP/1.1 200 OK
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache

  {
   "access_token": "SlAV32hkKG",
   "token_type": "Bearer",
   "refresh_token": "8xLOxBtZp8",
   "expires_in": 3600,
   "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
     yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
     NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZ
     fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
```

```
    AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
    Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
    NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
    QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
    K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
    XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
}
```

20. The Relying Party validates the Token Response from the Identity Exchange.

21. The Relying Party validates the ID Token. The Relying Party validates the signature on the ID Token using the public key for the Identity Exchange.

22. The Relying Party extracts the subject identifier (`sub`), from the ID Token, level of assurance (`acr`), and identity claims from the ID Token.

A.3.2.1 Additional Notes on Examples

- The OIDC interactions for the following steps are specified in the Relying Party to Identity Exchange Profile:
    - Steps 1 to 3 and 15 to 22.
- The OIDC interactions for the following steps are specified in the Identity Exchange to Identity Provider Profile:
    - Steps 5 to 14.
- The example assumes all required attributes are included in the ID Tokens. The client may retrieve additional attributes using the relevant UserInfo Endpoint.