OpenID Connect Conformance Profiles

OpenID Connect Working Group, OpenID Foundation

February 17, 2015

# Introduction

This document defines a set of profiles of the OpenID Connect specifications to be used for certifying implementations conforming to those profiles. This document also lists the features that must be supported by implementations certified as conforming to each profile and lists the tests used to test those features.

Many but not all of the features are able to be tested using the self-certification test procedures established by the OpenID Connect working group and the OpenID Foundation. The testing procedures for these features are described in the [Conformance Testing Procedures](http://openid.net/certification/testing).

# Overview of Conformance Profiles

This section briefly describes each of the currently defined conformance profiles. When we publish summaries of conformance self-certification results, these will be the columns in the certification results table and implementations will be the rows.

This section describes only the initial certification profiles included in the phase 1 launch of the OpenID Certification program in April 2015. Possible additional future profiles are described in a later section.

## OpenID Provider Conformance Profiles

### Basic OpenID Provider

Basic OpenID Providers implement the features needed by Basic Relying Parties – essentially, those that use the features described in the [OpenID Connect Basic Client Implementer’s Guide 1.0](http://openid.net/specs/openid-connect-basic-1_0.html) (although the actual profile will be based on [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html)). These features include the Mandatory to Implement Features for All OpenID Providers described in Section 15.1 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html).

### Implicit OpenID Provider

Implicit OpenID Providers implement the features needed by Implicit Relying Parties – those that use the features described in the [OpenID Connect Implicit Client Implementer’s Guide 1.0](http://openid.net/specs/openid-connect-implicit-1_0.html), excluding the Self-Issued OP features described in Section 4 (although the actual profile will be based on [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html)). These features include the Mandatory to Implement Features for All OpenID Providers described in Section 15.1 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html).

### Hybrid OpenID Provider

Hybrid OpenID Providers implement the features needed by Hybrid Relying Parties – those that use the features described in Section 3.3 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html).

### OpenID Provider Publishing Configuration Information

OpenID Providers Publishing Configuration Information publish their discovery information at provider configuration endpoints, as described in Sections 3 and 4 of [OpenID Connect Discovery 1.0](http://openid.net/specs/openid-connect-discovery-1_0.html). They also rotate their signing keys in the manner described in Section 10.1 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html).

### Dynamic OpenID Provider

Dynamic OpenID Providers implement the Mandatory to Implement Features for Dynamic OpenID Providers described in Section 15.2 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html). Note that conforming to the Dynamic OpenID Provider profile also means that the implementation will conform to the Basic OpenID Provider, Implicit OpenID Provider, and OpenID Provider Publishing Configuration Information profiles and implement the OP features of the [OpenID Connect Discovery 1.0](http://openid.net/specs/openid-connect-discovery-1_0.html) and [OpenID Connect Dynamic Client Registration 1.0](http://openid.net/specs/openid-connect-registration-1_0.html) specifications.

## Relying Party Conformance Profiles

### Basic Relying Party

Basic Relying Parties implement the features described in the [OpenID Connect Basic Client Implementer’s Guide 1.0](http://openid.net/specs/openid-connect-basic-1_0.html) (although the actual profile will be based on [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html)).

### Implicit Relying Party

Implicit Relying Parties implement the features described in the [OpenID Connect Implicit Client Implementer’s Guide 1.0](http://openid.net/specs/openid-connect-implicit-1_0.html), excluding the Self-Issued OP features described in Section 4 (although the actual profile will be based on [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html)).

### Hybrid Relying Party

Hybrid Relying Parties implement the features described in Section 3.3 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html).

### Relying Party Using Configuration Information

OpenID Relying Parties Using Configuration Information obtain info about the OpenID Providers that they use from provider configuration endpoints, as described in Sections 3 and 4 of [OpenID Connect Discovery 1.0](http://openid.net/specs/openid-connect-discovery-1_0.html). They also support OP signing key rotation in the manner described in Section 10.1 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html).

### Dynamic Relying Party

Dynamic Relying Parties implement the features of the Basic Relying Party, Implicit Relying Party, and Relying Party Using Configuration Information profiles. In addition to this, they implement the RP features of the [OpenID Connect Discovery 1.0](http://openid.net/specs/openid-connect-discovery-1_0.html) and [OpenID Connect Dynamic Client Registration 1.0](http://openid.net/specs/openid-connect-registration-1_0.html) specifications. It is recommended that Dynamic Relying Parties also seek certification as Relying Parties with Self-Issued OpenID Provider Support.

# Conformance Profile Definitions

## OpenID Provider Conformance Profile Definitions

The following table specifies the protocol features included in the OpenID Provider conformance profiles defined above. It also names the tests in the OpenID Provider test suite at <http://op.certification.openid.net/> that are used to test those features.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Conformance Feature Information** | | | **OP Conformance Profiles** | | | | |
| **Feature Name** | **Conformance Test Name** | **Test ID** | **Basic** | **Implicit** | **Hybrid** | **Config** | **Dynamic** |
| **Response Type & Response Mode** |  |  |  |  |  |  |  |
| Support code response\_type | Request with response\_type=code | OP-Response-code | y |  |  |  |  |
| Reject request without response\_type | Authorization request missing the response\_type parameter | OP-Response-Missing | y | y | y |  |  |
| Support id\_token response\_type | Request with response\_type=id\_token | OP-Response-id\_token |  | y |  |  |  |
| Support id\_token token response\_type | Request with response\_type=id\_token token | OP-Response-id\_token+token |  | y |  |  |  |
| Support code id\_token response\_type | Request with response\_type=code id\_token | OP-Response-code+id\_token |  |  | y |  |  |
| Support code token response\_type | Request with response\_type=code token | OP-Response-code+token |  |  | y |  |  |
| Support code id\_token token response\_type | Request with response\_type=code id\_token token | OP-Response-code+id\_token+token |  |  | y |  |  |
| **ID Token** |  |  |  |  |  |  |  |
| ID Token has iss claim |  | IdToken.verify() | y | y | y |  |  |
| ID Token has sub claim |  | IdToken.verify() | y | y | y |  |  |
| ID Token has aud claim |  | IdToken.verify() | y | y | y |  |  |
| ID Token has iat claim |  | IdToken.verify() | y | y | y |  |  |
| If left to itself is the OP signing the ID Token and with what | If left to itself is the OP signing the ID Token and with what | OP-IDToken-Signature | y | y | y |  |  |
| Asymmetric ID Token signature with RS256 | Asymmetric ID Token signature with RS256 | OP-IDToken-RS256 | y unless uses none | y | y |  |  |
| ID Token has kid claim | IDToken has kid | OP-IDToken-kid | y | y | y |  |  |
| ID Token has nonce when requested for code flow | ID Token has nonce when requested for code flow | OP-IDToken-nonce-code | y |  |  |  |  |
| ID Token has auth\_time claim when max\_age in request | Requesting ID Token with max\_age=1 seconds Restriction | OP-IDToken-max\_age=1 | y | y | y |  |  |
| Support max\_age request parameter when max age reached | Requesting ID Token with max\_age=1 seconds Restriction | OP-IDToken-max\_age=1 | y | y | y |  |  |
| Support max\_age request parameter when max age not reached | Requesting ID Token with max\_age=1000 seconds Restriction | OP-IDToken-max\_age=1000 | y | y | y |  |  |
| Unsecured ID Token signature with none | Unsecured ID Token signature with none | OP-IDToken-none | y if uses none |  |  | y if uses none | y if uses none |
| ID Token has nonce when requested for non-code flows | Request with nonce, verifies it was returned in id\_token | OP-IDToken-nonce-noncode |  | y | y |  |  |
| ID Token has at\_hash when ID Token and Access Token returned from Authorization Endpoint | ID Token has at\_hash when ID Token and Access Token returned from Authorization Endpoint | OP-IDToken-at\_hash |  | y | y |  |  |
| ID Token has c\_hash when ID Token and Authorization Code returned from Authorization Endpoint | ID Token has c\_hash when ID Token and Authorization Code returned from Authorization Endpoint | OP-IDToken-c\_hash |  |  | y |  |  |
| **UserInfo Endpoint** |  |  |  |  |  |  |  |
| Has UserInfo Endpoint | UserInfo Endpoint Access with GET and bearer\_header | OP-UserInfo-Endpoint | y | y | y |  |  |
| UserInfo Endpoint access with header method | UserInfo Endpoint Access with POST and bearer\_header | OP-UserInfo-Header | y | y | y |  |  |
| UserInfo Endpoint access with form-encoded body method | UserInfo Endpoint Access with POST and bearer\_body | OP-UserInfo-Body | y | y | y |  |  |
| UserInfo has sub claim |  | OpenIDSchema.verify() | y | y | y |  |  |
| Can provide signed UserInfo response with RS256 | RP registers userinfo\_signed\_response\_alg to signal that it wants signed UserInfo returned | OP-UserInfo-RS256 |  |  |  |  | y |
| **nonce Request Parameter** |  |  |  |  |  |  |  |
| Support requests without nonce when using the code flow | Login no nonce, code flow | OP-nonce-NoReq-code | y |  |  |  |  |
| Reject requests without nonce unless using the code flow | Reject requests without nonce unless using the code flow | OP-nonce-NoReq-noncode |  | y | y |  |  |
| **scope Request Parameter** |  |  |  |  |  |  |  |
| Support openid scope | UserInfo Endpoint Access with GET and bearer\_header | OP-UserInfo-Endpoint | no err | no err | no err |  |  |
| Support profile scope | Scope Requesting profile Claims | OP-scope-profile | no err | no err | no err |  |  |
| Support email scope | Scope Requesting email Claims | OP-scope-email | no err | no err | no err |  |  |
| Support address scope | Scope Requesting address Claims | OP-scope-address | no err | no err | no err |  |  |
| Support phone scope | Scope Requesting phone Claims | OP-scope-phone | no err | no err | no err |  |  |
| Support scope value requesting all basic claims | Scope Requesting all Claims | OP-scope-All | no err | no err | no err |  |  |
| **display Request Parameter** |  |  |  |  |  |  |  |
| Support display value page | Request with display=page | OP-display-page | no err | no err | no err |  |  |
| Support display value popup | Request with display=popup | OP-display-popup | no err | no err | no err |  |  |
| **prompt Request Parameter** |  |  |  |  |  |  |  |
| Support prompt value login | Request with prompt=login | OP-prompt-login | y | y | y |  |  |
| Support prompt value none | Request with prompt=none when not logged in | OP-prompt-none-NotLoggedIn | y | y | y |  |  |
| Support prompt value none | Request with prompt=none when logged in | OP-prompt-none-LoggedIn | y | y | y |  |  |
| **Misc Request Parameters** |  |  |  |  |  |  |  |
| Support max\_age request parameter | Requesting ID Token with max\_age=1 seconds Restriction | OP-IDToken-max\_age=1 | y | y | y |  |  |
| Ignores not understood query parameter in Authentication Request | Request with extra query component | OP-Req-NotUnderstood | y | y | y |  |  |
| Support id\_token\_hint request parameter | Using prompt=none with user hint through id\_token\_hint | OP-Req-id\_token\_hint | SHOULD | SHOULD | SHOULD |  |  |
| Support login\_hint request parameter | Providing login\_hint | OP-Req-login\_hint | no err | no err | no err |  |  |
| Support ui\_locales request parameter | Providing ui\_locales | OP-Req-ui\_locales | no err | no err | no err |  |  |
| Support claims\_locales request parameter | Providing claims\_locales | OP-Req-claims\_locales | no err | no err | no err |  |  |
| Support acr\_values request parameter | Providing acr\_values | OP-Req-acr\_values | no err | no err | no err |  |  |
| **OAuth Behaviors** |  |  |  |  |  |  |  |
| OAuth state request value returned in response |  | VerifyState() | y | y | y |  |  |
| Reject second use of Authorization Code | Trying to use access code twice should result in an error | OP-OAuth-2nd | OAuth MUST |  | OAuth MUST |  |  |
| Second use of Authorization Code revokes previously issued Access Token | Trying to use access code twice should result in revoking previous issued tokens | OP-OAuth-2nd-Revokes | OAuth SHOULD |  | OAuth SHOULD |  |  |
| Reject second use of Authorization Code | Trying to use access code twice with 30 seconds in between must result in an error | OP-OAuth-2nd-30s | OAuth MUST |  | OAuth MUST |  |  |
| **redirect\_uri** |  |  |  |  |  |  |  |
| Reject redirect\_uri not matching a registered redirect\_uri | The sent redirect\_uri does not match the registered | OP-redirect\_uri-NotReg | y | y | y |  |  |
| Reject request without redirect\_uri when multiple registered | Reject request without redirect\_uri when multiple registered | OP-redirect\_uri-Missing | y | y | y |  |  |
| Preserves query parameter in redirect\_uri | Request with redirect\_uri with query component | OP-redirect\_uri-Query | y | y | y |  |  |
| Preserves query parameter in registered redirect\_uris | Registration where a redirect\_uri has a query component | OP-redirect\_uri-RegQuery |  |  |  |  | y |
| Reject redirect\_uri when query parameter does not match | Rejects redirect\_uri when Query Parameter Does Not Match | OP-redirect\_uri-BadQuery | y | y | y |  |  |
| Reject registration of redirect\_uris with fragment | Registration where a redirect\_uri has a fragment | OP-redirect\_uri-RegFrag |  |  |  |  | y |
| **Client Authentication** |  |  |  |  |  |  |  |
| Support client authentication to Token Endpoint using HTTP Basic with POST | Access token request with client\_secret\_basic authentication | OP-ClientAuth-Basic-Dynamic | y |  | y |  |  |
|  | Access token request with client\_secret\_basic authentication | OP-ClientAuth-Basic-Static | y |  | y |  |  |
| Support client authentication to Token Endpoint using form-encoded client credentials in POST body | Access token request with client\_secret\_post authentication | OP-ClientAuth-SecretPost-Dynamic | y |  | y |  |  |
|  | Access token request with client\_secret\_post authentication | OP-ClientAuth-SecretPost-Static | y |  | y |  |  |
| **Discovery** |  |  |  |  |  |  |  |
| Publish openid-configuration discovery information | Publish openid-configuration discovery information | OP-Discovery-Config |  |  |  | y | y |
| Config has issuer |  | ProviderConfigurationResponse.verify() |  |  |  | y | y |
| Discovered issuer matches openid-configuration path prefix |  | ProviderConfigurationResponse.verify() |  |  |  | y | y |
| Discovered issuer matches ID Token iss value |  | IdToken.verify() |  |  |  | y | y |
| Config has authorization\_endpoint |  | CheckEndpoint() |  |  |  | y | y |
| Config has token\_endpoint |  | CheckEndpoint() |  |  |  | y unless only Implicit | y |
| Config has userinfo\_endpoint |  | CheckEndpoint() |  |  |  | y unless self-issued | y |
| Config has jwks\_uri | Verify that jwks\_uri and claims\_supported are published | OP-Discovery-Values |  |  |  | y unless only none | y |
| Keys in OP JWKs well formed | Keys in OP JWKs well formed | OP-Discovery-JWKs |  |  |  | y unless only none | y |
| Config has scopes\_supported |  | CheckScopeSupport() |  |  |  | y | y |
| Config has response\_types\_supported |  | ProviderConfigurationResponse.verify() |  |  |  | y | y |
| Config has subject\_types\_supported |  | ProviderConfigurationResponse.verify() |  |  |  | y | y |
| Config has id\_token\_signing\_alg\_values\_supported |  | ProviderConfigurationResponse.verify() |  |  |  | y unless only none | y |
| Config has claims\_supported | Verify that jwks\_uri and claims\_supported are published | OP-Discovery-Values |  |  |  | y | y |
| All OP endpoints use https |  | VerifyOPEndpointsUseHTTPS() |  |  |  | y | y |
| Can Discover Identifiers using E-Mail Syntax | Can Discover Identifiers using E-Mail Syntax | OP-Discovery-WebFinger-Email |  |  |  |  | y |
| Support WebFinger discovery | Can Discover Identifiers using URL Syntax | OP-Discovery-WebFinger |  |  |  |  | y |
| **Dynamic Client Registration** |  |  |  |  |  |  |  |
| Config has registration\_endpoint | Verify that registration\_endpoint is published | OP-Registration-Endpoint |  |  |  |  | y |
| Enables dynamic registration | Client registration Request | OP-Registration-Dynamic |  |  |  |  | y |
| Support using Sector Identifier for pairwise sub values |  |  | SHOULD | SHOULD | SHOULD |  | no err |
| Displays logo\_uri in login page | Registration with logo\_uri | OP-Registration-logo\_uri | SHOULD | SHOULD | SHOULD |  |  |
| Displays policy\_uri in login page | Registration with policy\_uri | OP-Registration-policy\_uri | SHOULD | SHOULD | SHOULD |  |  |
| Displays tos\_uri in login page | Registration with tos\_uri | OP-Registration-tos\_uri | SHOULD | SHOULD | SHOULD |  |  |
| Uses keys registered with jwks value | Uses Keys Registered with jwks Value | OP-Registration-jwks |  |  |  |  | y |
| Uses keys registered with jwks\_uri value | Uses Keys Registered with jwks\_uri Value | OP-Registration-jwks\_uri |  |  |  |  | y |
| Reject Sector Identifier not containing registered redirect\_uri values | Incorrect registration of sector\_identifier\_uri | OP-Registration-Sector-Bad |  |  |  |  | y |
| **Key Rollover** |  |  |  |  |  |  |  |
| Can rollover OP signing key | Can Rollover OP Signing Key | OP-Rollover-OP-Sig |  |  |  | y | y |
| Support RP signing key rollover | Request access token, change RSA signing key and request another access token | OP-Rollover-RP-Sig |  |  |  |  | y |
| **request\_uri Request Parameter** |  |  |  |  |  |  |  |
| Support request\_uri request parameter | Support request\_uri Request Parameter | OP-request\_uri-Support | no err | no err | no err |  | y |
| Support request\_uri request parameter with unsecured request | Support request\_uri Request Parameter with unSigned Request | OP-request\_uri-Unsigned | no err | no err | no err |  | y |
| Support request\_uri request parameter with signed request | Support request\_uri Request Parameter with Signed Request | OP-request\_uri-Sig |  |  |  |  | y |
| **request Request Parameter** |  |  |  |  |  |  |  |
| Support request request parameter | Support request Request Parameter | OP-request-Support | no err | no err | no err |  |  |
| Support request request parameter with unsecured request | Support request Request Parameter with unSigned Request | OP-request-Unsigned | no err | no err | no err |  |  |
| **claims Request Parameter** |  |  |  |  |  |  |  |
| Support claims request parameter | Claims Request with Essential name Claim | OP-claims-essential | no err | no err | no err |  |  |

## Relying Party Conformance Profile Definitions

The following table specifies the protocol features included in the Relying Party conformance profiles defined above. A future version of this table will also name the tests in the Relying Party test suite at <http://rp.certification.openid.net/> that are used to test those features.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Conformance Feature Information** | **RP Conformance Profiles** | | | | |
| **Feature Name** | **Basic** | **Implicit** | **Hybrid** | **Config** | **Dynamic** |
| **Response Type & Resonse Mode** |  |  |  |  |  |
| Can make request with code response\_type | y |  |  |  |  |
| Can make request with id\_token response\_type |  | y |  |  |  |
| Can make request with id\_token token response\_type |  | y |  |  |  |
| **ID Token** |  |  |  |  |  |
| Reject ID Token with invalid iss claim | y | y | y |  |  |
| Reject ID Token without sub claim | y | y | y |  |  |
| Reject ID Token with invalid aud claim | y | y | y |  |  |
| Reject ID Token without iat claim | y | y | y |  |  |
| Accept ID Token without kid claim if only one JWK supplied in jwks\_uri | optional | y | y |  |  |
| Reject ID Token without kid claim if multiple JWKs supplied in jwks\_uri | optional | rejection allowed | rejection allowed |  |  |
| Reject invalid at\_hash when ID Token and Access Token returned from Authorization Endpoint |  | y | y |  |  |
| Reject invalid c\_hash when ID Token and Authorization Code returned from Authorization Endpoint |  |  | y |  |  |
| Reject invalid asymmetric ID Token signature with rs256 | optional | y | y |  |  |
| Can request and use unsecured ID Token signature | optional |  |  | use optional | use optional |
| **UserInfo Endpoint** |  |  |  |  |  |
| Accesses UserInfo Endpoint with header method | y | y | y |  |  |
| Accesses UserInfo Endpoint with form-encoded body method | alt to hdr mthd | alt to hdr mthd | alt to hdr mthd |  |  |
| Does not access UserInfo Endpoint with query parameter method | y | y | y |  |  |
| Reject UserInfo with invalid sub claim | y | y | y |  |  |
| Can request and use signed UserInfo response |  |  |  | use optional | use optional |
| **nonce Request Parameter** |  |  |  |  |  |
| Sends nonce request parameter unless using code flow |  | y | y |  |  |
| Reject ID Token with invalid nonce when nonce valid sent | y | y | y |  |  |
| **scope Request Parameter** |  |  |  |  |  |
| Scope openid present in all requests | y | y | y |  |  |
| Can request UserInfo claims with scope values | use optional | use optional | use optional |  |  |
| **Client Authentication** |  |  |  |  |  |
| Can make Access Token request using client\_secret\_basic client authentication | y | y | y |  |  |
| **Discovery** |  |  |  |  |  |
| Uses WebFinger discovery |  |  |  |  | y |
| Can discover identifiers using e-mail syntax |  |  |  |  | y |
| Can discover identifiers using URL syntax |  |  |  |  | y |
| Uses openid-configuration discovery information |  |  |  | y | y |
| Reject discovered issuer not matching openid-configuration path prefix |  |  |  | y | y |
| Reject ID Token with iss not matching discovered issuer |  |  |  | y | y |
| Uses keys discovered with jwks\_uri value |  |  |  | y | y |
| **Dynamic Client Registration** |  |  |  |  |  |
| Uses dynamic registration |  |  |  |  | y |
| Registration has redirect\_uris |  |  |  |  | y |
| Keys in RP JWKs well formed |  |  |  |  | y |
| Uses https for all endpoints unless only using code flow | y | y | y |  |  |
| **Key Rollover** |  |  |  |  |  |
| Support OP signing key rollover |  |  |  | y | y |
| Can rollover RP signing key |  |  |  |  | y |
| **request\_uri Request Parameter** |  |  |  |  |  |
| Can use request\_uri request parameter with unsecured request |  |  |  |  | use optional |
| Can use request\_uri request parameter with signed request |  |  |  |  | use optional |

# Possible Future Conformance Profiles

## Possible Future OpenID Provider Conformance Profiles

### Self-Issued OpenID Provider

Self-Issued OpenID Providers implement the OP features described in Section 7 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html). These OPs must also implement the Mandatory to Implement Features for All OpenID Providers described in Section 15.1 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html).

### OpenID Provider Using Form Post Response Mode

OpenID Providers Using Form Post Response Mode implement the [OAuth 2.0 Form Post Response Mode](http://openid.net/specs/oauth-v2-form-post-response-mode-1_0.html) specification.

### OpenID Provider Issuing Refresh Tokens

OpenID Providers Issuing Refresh Tokens issue and use Refresh Tokens in the manner described in Sections 11 and 12 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html).

### Full OpenID Provider

Full OpenID Providers implement all six of the response\_type values specified in Section 3 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html). They implement the “request”, “request\_uri”, and “claims” request parameters. They support encrypted requests and encrypted responses. They support rotation of RP and OP singing and encryption keys. They support both public and pairwise subject identifiers. They support offline access. They support all the client authentication methods defined in Section 9. These OPs must also implement the Mandatory to Implement Features for All OpenID Providers described in Section 15.1 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html).

## Possible Future Relying Party Conformance Profiles

### Self-Issued Relying Party

Self-Issued Relying Parties implement the RP features described in Section 7 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html).

### Relying Parties Using Form Post Response Mode

Relying Parties Using Form Post Response Mode implement the [OAuth 2.0 Form Post Response Mode](http://openid.net/specs/oauth-v2-form-post-response-mode-1_0.html) specification.

### Relying Party Using Refresh Tokens

Relying Parties Using Refresh Tokens use Refresh Tokens in the manner described in Sections 11 and 12 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html).

### Full Relying Party

Full Relying Parties implement all six of the response\_type values specified in Section 3 of [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html). They implement the “request”, “request\_uri”, and “claims” request parameters. They support encrypted requests and encrypted responses. They support rotation of RP and OP singing and encryption keys. They can request offline access.