Internet Engineering Task Force (IETF)
Request for Comments: 7521
Category: Standards Track
ISSN: 2070-1721

B. Campbell
Ping Identity
C. Mortimore
Salesforce
M. Jones
Y. Goland
Microsoft
May 2015

Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants

Abstract

This specification provides a framework for the use of assertions with OAuth 2.0 in the form of a new client authentication mechanism and a new authorization grant type. Mechanisms are specified for transporting assertions during interactions with a token endpoint; general processing rules are also specified.

The intent of this specification is to provide a common framework for OAuth 2.0 to interwork with other identity systems using assertions and to provide alternative client authentication mechanisms.

Note that this specification only defines abstract message flows and processing rules. In order to be implementable, companion specifications are necessary to provide the corresponding concrete instantiations.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7521.

Table of Contents

1. Introduction

An assertion is a package of information that facilitates the sharing of identity and security information across security domains. Section 3 provides a more detailed description of the concept of an assertion for the purpose of this specification.

OAuth 2.0 [RFC6749] is an authorization framework that enables a third-party application to obtain limited access to a protected HTTP resource. In OAuth, those third-party applications are called clients; they access protected resources by presenting an access token to the HTTP resource. Access tokens are issued to clients by an authorization server with the (sometimes implicit) approval of the resource owner. These access tokens are typically obtained by exchanging an authorization grant, which represents the authorization granted by the resource owner (or by a privileged administrator). Several authorization grant types are defined to support a wide range of client types and user experiences. OAuth also provides an extensibility mechanism for defining additional grant types, which can serve as a bridge between OAuth and other protocol frameworks.

This specification provides a general framework for the use of assertions as authorization grants with OAuth 2.0. It also provides a framework for assertions to be used for client authentication. It provides generic mechanisms for transporting assertions during interactions with an authorization server's token endpoint as well as general rules for the content and processing of those assertions. The intent is to provide an alternative client authentication mechanism (one that doesn't send client secrets) and to facilitate the use of OAuth 2.0 in client-server integration scenarios, where the end user may not be present.

This specification only defines abstract message flows and processing rules. In order to be implementable, companion specifications are necessary to provide the corresponding concrete instantiations. For instance, "Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants" [RFC7522] defines a concrete instantiation for Security Assertion Markup Language (SAML) 2.0 Assertions and "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants" [RFC7523] defines a concrete instantiation for JWTs.

Note: The use of assertions for client authentication is orthogonal to and separable from using assertions as an authorization grant. They can be used either in combination or separately. Client assertion authentication is nothing more than an alternative way for a client to authenticate to the token endpoint and must be used in conjunction with some grant type to form a complete and meaningful

protocol request. Assertion authorization grants may be used with or without client authentication or identification. Whether or not client authentication is needed in conjunction with an assertion authorization grant, as well as the supported types of client authentication, are policy decisions at the discretion of the authorization server.

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Throughout this document, values are quoted to indicate that they are to be taken literally. When using these values in protocol messages, the quotes must not be used as part of the value.

3. Framework

An assertion is a package of information that allows identity and security information to be shared across security domains. An assertion typically contains information about a subject or principal, information about the party that issued the assertion and when was it issued, and the conditions under which the assertion is to be considered valid, such as when and where it can be used.

The entity that creates and signs or integrity-protects the assertion is typically known as the "Issuer", and the entity that consumes the assertion and relies on its information is typically known as the "Relying Party". In the context of this document, the authorization server acts as a relying party.

Assertions used in the protocol exchanges defined by this specification MUST always be integrity protected using a digital signature or Message Authentication Code (MAC) applied by the issuer, which authenticates the issuer and ensures integrity of the assertion content. In many cases, the assertion is issued by a third party, and it must be protected against tampering by the client that presents it. An assertion MAY additionally be encrypted, preventing unauthorized parties (such as the client) from inspecting the content.

Although this document does not define the processes by which the client obtains the assertion (prior to sending it to the authorization server), there are two common patterns described below.

In the first pattern, depicted in Figure 1, the client obtains an assertion from a third-party entity capable of issuing, renewing, transforming, and validating security tokens. Typically, such an entity is known as a "security token service" (STS) or just "token service", and a trust relationship (usually manifested in the exchange of some kind of key material) exists between the token service and the relying party. The token service is the assertion issuer; its role is to fulfill requests from clients, which present various credentials, and mint assertions as requested, fill them with appropriate information, and integrity-protect them with a signature or message authentication code. WS-Trust [OASIS.WS-Trust] is one available standard for requesting security tokens (assertions).

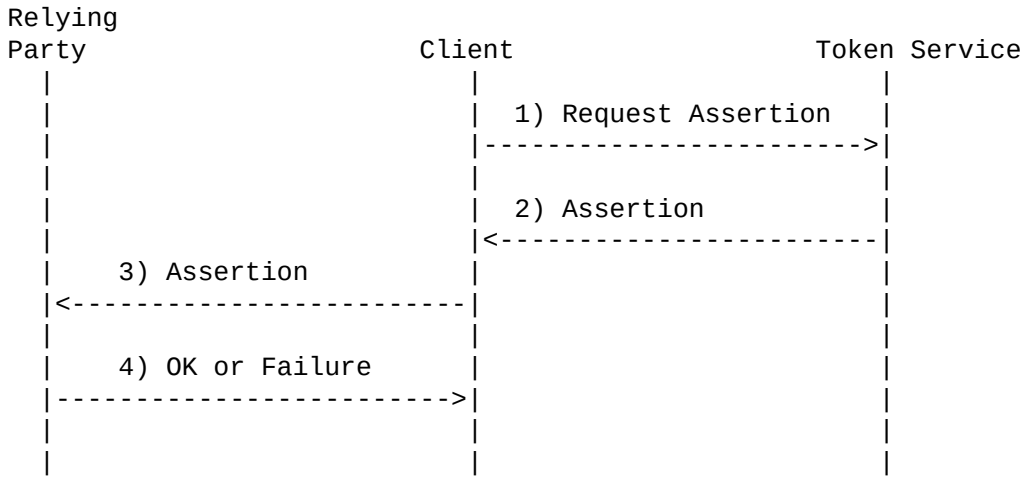

Figure 1: Assertion Created by Third Party

In the second pattern, depicted in Figure 2, the client creates assertions locally. To apply the signatures or message authentication codes to assertions, it has to obtain key material: either symmetric keys or asymmetric key pairs. The mechanisms for obtaining this key material are beyond the scope of this specification.

Although assertions are usually used to convey identity and security information, self-issued assertions can also serve a different purpose. They can be used to demonstrate knowledge of some secret, such as a client secret, without actually communicating the secret directly in the transaction. In that case, additional information included in the assertion by the client itself will be of limited value to the relying party, and for this reason, only a bare minimum of information is typically included in such an assertion, such as information about issuing and usage conditions.

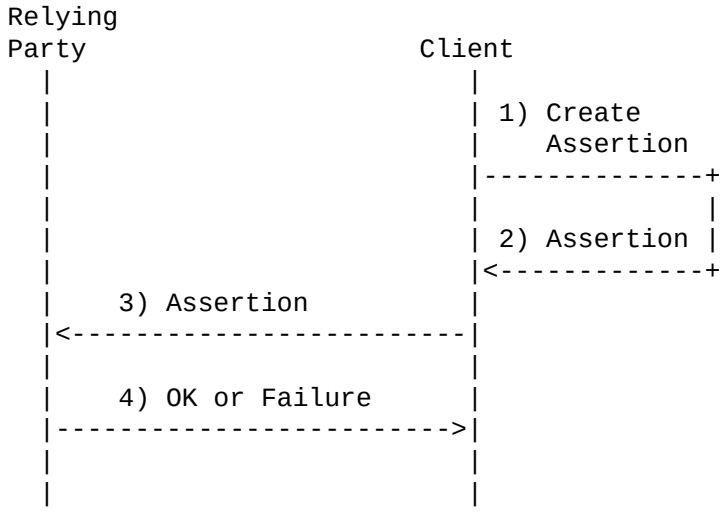

Figure 2: Self-Issued Assertion

Deployments need to determine the appropriate variant to use based on the required level of security, the trust relationship between the entities, and other factors.

From the perspective of what must be done by the entity presenting the assertion, there are two general types of assertions:

1. Bearer Assertions: Any entity in possession of a bearer assertion (the bearer) can use it to get access to the associated resources (without demonstrating possession of a cryptographic key). To prevent misuse, bearer assertions need to be protected from disclosure in storage and in transport. Secure communication channels are required between all entities to avoid leaking the assertion to unauthorized parties.

2. Holder-of-Key Assertions: To access the associated resources, the entity presenting the assertion must demonstrate possession of additional cryptographic material. The token service thereby binds a key identifier to the assertion, and the client has to demonstrate to the relying party that it knows the key corresponding to that identifier when presenting the assertion.

The protocol parameters and processing rules defined in this document are intended to support a client presenting a bearer assertion to an authorization server. They are not directly suitable for use with holder-of-key assertions. While they could be used as a baseline for a holder-of-key assertion system, there would be a need for

additional mechanisms (to support proof-of-possession of the secret key), and possibly changes to the security model (e.g., to relax the requirement for an Audience).

4. Transporting Assertions

This section defines HTTP parameters for transporting assertions during interactions with a token endpoint of an OAuth authorization server. Because requests to the token endpoint result in the transmission of clear-text credentials (in both the HTTP request and response), all requests to the token endpoint MUST use Transport Layer Security (TLS), as mandated in Section 3.2 of OAuth 2.0 [RFC6749].

4.1. Using Assertions as Authorization Grants

This section defines the use of assertions as authorization grants, based on the definition provided in Section 4.5 of OAuth 2.0 [RFC6749]. When using assertions as authorization grants, the client includes the assertion and related information using the following HTTP request parameters:

grant_type
: REQUIRED. The format of the assertion as defined by the authorization server. The value will be an absolute URI.

assertion
: REQUIRED. The assertion being used as an authorization grant. Specific serialization of the assertion is defined by profile documents.

scope
: OPTIONAL. The requested scope as described in Section 3.3 of OAuth 2.0 [RFC6749]. When exchanging assertions for access tokens, the authorization for the token has been previously granted through some out-of-band mechanism. As such, the requested scope MUST be equal to or less than the scope originally granted to the authorized accessor. The authorization server MUST limit the scope of the issued access token to be equal to or less than the scope originally granted to the authorized accessor.

Authentication of the client is optional, as described in Section 3.2.1 of OAuth 2.0 [RFC6749], and consequently, the "client_id" is only needed when a form of client authentication that relies on the parameter is used.

The following example demonstrates an assertion being used as an authorization grant (with extra line breaks for display purposes only):

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Asaml2-bearer&
assertion=PHNhbWxwOl...[omitted for brevity]...ZT4
```

An assertion used in this context is generally a short-lived representation of the authorization grant, and authorization servers SHOULD NOT issue access tokens with a lifetime that exceeds the validity period of the assertion by a significant period. In practice, that will usually mean that refresh tokens are not issued in response to assertion grant requests, and access tokens will be issued with a reasonably short lifetime. Clients can refresh an expired access token by requesting a new one using the same assertion, if it is still valid, or with a new assertion.

An IETF URN for use as the "grant_type" value can be requested using the template in [RFC6755]. A URN of the form urn:ietf:params:oauth:grant-type:* is suggested.

4.1.1. Error Responses

If an assertion is not valid or has expired, the authorization server constructs an error response as defined in OAuth 2.0 [RFC6749]. The value of the "error" parameter MUST be the "invalid_grant" error code. The authorization server MAY include additional information regarding the reasons the assertion was considered invalid using the "error_description" or "error_uri" parameters.

For example:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error":"invalid_grant",
  "error_description":"Audience validation failed"
}
```

4.2. Using Assertions for Client Authentication

The following section defines the use of assertions as client credentials as an extension of Section 2.3 of OAuth 2.0 [RFC6749]. When using assertions as client credentials, the client includes the assertion and related information using the following HTTP request parameters:

client_assertion_type
: REQUIRED. The format of the assertion as defined by the authorization server. The value will be an absolute URI.

client_assertion
: REQUIRED. The assertion being used to authenticate the client. Specific serialization of the assertion is defined by profile documents.

client_id
: OPTIONAL. The client identifier as described in Section 2.2 of OAuth 2.0 [RFC6749]. The "client_id" is unnecessary for client assertion authentication because the client is identified by the subject of the assertion. If present, the value of the "client_id" parameter MUST identify the same client as is identified by the client assertion.

The following example demonstrates a client authenticating using an assertion during an access token request, as defined in Section 4.1.3 of OAuth 2.0 [RFC6749] (with extra line breaks for display purposes only):

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=n0esc3NRze7LTCu7iYzS6a5acc3f0ogp4&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth
%3Aclient-assertion-type%3Asaml2-bearer&
client_assertion=PHNhbW...[omitted for brevity]...ZT
```

Token endpoints can differentiate between assertion-based credentials and other client credential types by looking for the presence of the "client_assertion" and "client_assertion_type" parameters, which will only be present when using assertions for client authentication.

An IETF URN for use as the "client_assertion_type" value may be requested using the template in [RFC6755]. A URN of the form urn:ietf:params:oauth:client-assertion-type:* is suggested.

4.2.1. Error Responses

If an assertion is invalid for any reason or if more than one client authentication mechanism is used, the authorization server constructs an error response as defined in OAuth 2.0 [RFC6749]. The value of the "error" parameter MUST be the "invalid_client" error code. The authorization server MAY include additional information regarding the reasons the client assertion was considered invalid using the "error_description" or "error_uri" parameters.

For example:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error":"invalid_client"
  "error_description":"assertion has expired"
}
```

5. Assertion Content and Processing

This section provides a general content and processing model for the use of assertions in OAuth 2.0 [RFC6749].

5.1. Assertion Metamodel

The following are entities and metadata involved in the issuance, exchange, and processing of assertions in OAuth 2.0. These are general terms, abstract from any particular assertion format. Mappings of these terms into specific representations are provided by profiles of this specification.

Issuer
: A unique identifier for the entity that issued the assertion. Generally, this is the entity that holds the key material used to sign or integrity-protect the assertion. Examples of issuers are OAuth clients (when assertions are self-issued) and third-party security token services. If the assertion is self-issued, the Issuer value is the client identifier. If the assertion was issued by a security token service (STS), the Issuer should identify the STS in a manner recognized by the authorization server. In the absence of an application profile specifying otherwise, compliant applications MUST compare Issuer values using the Simple String Comparison method defined in Section 6.2.1 of RFC 3986 [RFC3986].

Subject
: A unique identifier for the principal that is the subject of the assertion.

  * When using assertions for client authentication, the Subject identifies the client to the authorization server using the value of the "client_id" of the OAuth client.

  * When using assertions as an authorization grant, the Subject identifies an authorized accessor for which the access token is being requested (typically, the resource owner or an authorized delegate).

Audience
: A value that identifies the party or parties intended to process the assertion. The URL of the token endpoint, as defined in Section 3.2 of OAuth 2.0 [RFC6749], can be used to indicate that the authorization server is a valid intended audience of the assertion. In the absence of an application profile specifying otherwise, compliant applications MUST compare the Audience values using the Simple String Comparison method defined in Section 6.2.1 of RFC 3986 [RFC3986].

Issued At
: The time at which the assertion was issued. While the serialization may differ by assertion format, it is REQUIRED that the time be expressed in UTC with no time zone component.

Expires At
: The time at which the assertion expires. While the serialization may differ by assertion format, it is REQUIRED that the time be expressed in UTC with no time zone component.

Assertion ID
: A nonce or unique identifier for the assertion. The Assertion ID may be used by implementations requiring message de-duplication for one-time use assertions. Any entity that assigns an identifier MUST ensure that there is negligible probability for that entity or any other entity to accidentally assign the same identifier to a different data object.

5.2. General Assertion Format and Processing Rules

The following are general format and processing rules for the use of assertions in OAuth:

- The assertion MUST contain an Issuer. The Issuer identifies the entity that issued the assertion as recognized by the authorization server. If an assertion is self-issued, the Issuer MUST be the value of the client's "client_id".

- The assertion MUST contain a Subject. The Subject typically identifies an authorized accessor for which the access token is being requested (i.e., the resource owner or an authorized delegate) but, in some cases, may be a pseudonymous identifier or other value denoting an anonymous user. When the client is acting on behalf of itself, the Subject MUST be the value of the client's "client_id".

- The assertion MUST contain an Audience that identifies the authorization server as the intended audience. The authorization server MUST reject any assertion that does not contain its own identity as the intended audience.

- The assertion MUST contain an Expires At entity that limits the time window during which the assertion can be used. The authorization server MUST reject assertions that have expired (subject to allowable clock skew between systems). Note that the authorization server may reject assertions with an Expires At attribute value that is unreasonably far in the future.

- The assertion MAY contain an Issued At entity containing the UTC time at which the assertion was issued.

- The authorization server MUST reject assertions with an invalid signature or MAC. The algorithm used to validate the signature or message authentication code and the mechanism for designating the secret used to generate the signature or message authentication code over the assertion are beyond the scope of this specification.

6. Common Scenarios

The following provides additional guidance, beyond the format and processing rules defined in Sections 4 and 5, on assertion use for a number of common use cases.

6.1. Client Authentication

A client uses an assertion to authenticate to the authorization server's token endpoint by using the "client_assertion_type" and "client_assertion" parameters as defined in Section 4.2. The Subject of the assertion identifies the client. If the assertion is self-issued by the client, the Issuer of the assertion also identifies the client.

The example in Section 4.2 shows a client authenticating using an assertion during an access token request.

6.2. Client Acting on Behalf of Itself

When a client is accessing resources on behalf of itself, it does so in a manner analogous to the Client Credentials Grant defined in Section 4.4 of OAuth 2.0 [RFC6749]. This is a special case that combines both the authentication and authorization grant usage patterns. In this case, the interactions with the authorization server should be treated as using an assertion for Client Authentication according to Section 4.2, while using the "grant_type" parameter with the value "client_credentials" to indicate that the client is requesting an access token using only its client credentials.

The following example demonstrates an assertion being used for a client credentials access token request, as defined in Section 4.4.2 of OAuth 2.0 [RFC6749] (with extra line breaks for display purposes only):

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth
%3Aclient-assertion-type%3Asaml2-bearer&
client_assertion=PHNhbW...[omitted for brevity]...ZT
```

6.3. Client Acting on Behalf of a User

When a client is accessing resources on behalf of a user, it does so by using the "grant_type" and "assertion" parameters as defined in Section 4.1. The Subject identifies an authorized accessor for which the access token is being requested (typically, the resource owner or an authorized delegate).

The example in Section 4.1 shows a client making an access token request using an assertion as an authorization grant.

6.3.1. Client Acting on Behalf of an Anonymous User

When a client is accessing resources on behalf of an anonymous user, a mutually agreed-upon Subject identifier indicating anonymity is used. The Subject value might be an opaque persistent or transient pseudonymous identifier for the user or be an agreed-upon static value indicating an anonymous user (e.g., "anonymous"). The authorization may be based upon additional criteria, such as additional attributes or claims provided in the assertion. For example, a client might present an assertion from a trusted issuer asserting that the bearer is over 18 via an included claim. In this case, no additional information about the user's identity is included, yet all the data needed to issue an access token is present.

More information about anonymity, pseudonymity, and privacy considerations in general can be found in [RFC6973].

7. Interoperability Considerations

This specification defines a framework for using assertions with OAuth 2.0. However, as an abstract framework in which the data formats used for representing many values are not defined, on its own, this specification is not sufficient to produce interoperable implementations.

Two other specifications that profile this framework for specific assertions have been developed: [RFC7522] uses SAML 2.0 Assertions and [RFC7523] uses JSON Web Tokens (JWTs). These two instantiations of this framework specify additional details about the assertion encoding and processing rules for using those kinds of assertions with OAuth 2.0.

However, even when profiled for specific assertion types, agreements between system entities regarding identifiers, keys, and endpoints are required in order to achieve interoperable deployments. Specific items that require agreement are as follows: values for the Issuer and Audience identifiers, supported assertion and client authentication types, the location of the token endpoint, the key used to apply and verify the digital signature or MAC over the assertion, one-time use restrictions on assertions, maximum assertion lifetime allowed, and the specific Subject and attribute requirements of the assertion. The exchange of such information is explicitly out of the scope of this specification. Deployments for particular trust frameworks, circles of trust, or other uses cases will need to agree

among the participants on the kinds of values to be used for some abstract fields defined by this specification. In some cases, additional profiles may be created that constrain or prescribe these values or specify how they are to be exchanged. The "OAuth 2.0 Dynamic Client Registration Core Protocol" [OAUTH-DYN-REG] is one such profile that enables OAuth Clients to register metadata about themselves at an authorization server.

8. Security Considerations

This section discusses security considerations that apply when using assertions with OAuth 2.0 as described in this document. As discussed in Section 3, there are two different ways to obtain assertions: either as self-issued or obtained from a third-party token service. While the actual interactions for obtaining an assertion are outside the scope of this document, the details are important from a security perspective. Section 3 discusses the high-level architectural aspects. Many of the security considerations discussed in this section are applicable to both the OAuth exchange as well as the client obtaining the assertion.

The remainder of this section focuses on the exchanges that concern presenting an assertion for client authentication and for the authorization grant.

8.1. Forged Assertion

Threat:
: An adversary could forge or alter an assertion in order to obtain an access token (in the case of the authorization grant) or to impersonate a client (in the case of the client authentication mechanism).

Countermeasures:
: To avoid this kind of attack, the entities must assure that proper mechanisms for protecting the integrity of the assertion are employed. This includes the issuer digitally signing the assertion or computing a MAC over the assertion.

8.2. Stolen Assertion

Threat:
: An adversary may be able obtain an assertion (e.g., by eavesdropping) and then reuse it (replay it) at a later point in time.

Countermeasures:

The primary mitigation for this threat is the use of secure communication channels with server authentication for all network exchanges.

An assertion may also contain several elements to prevent replay attacks. There is, however, a clear trade-off between reusing an assertion for multiple exchanges and obtaining and creating new, fresh assertions.

Authorization servers and resource servers may use a combination of the Assertion ID and Issued At/Expires At attributes for replay protection. Previously processed assertions may be rejected based on the Assertion ID. The addition of the validity window relieves the authorization server from maintaining an infinite state table of processed Assertion IDs.

8.3. Unauthorized Disclosure of Personal Information

Threat:

The ability for other entities to obtain information about an individual, such as authentication information, role in an organization, or other authorization-relevant information, raises privacy concerns.

Countermeasures:

To address this threat, two cases need to be differentiated:

First, a third party that did not participate in any of the exchange is prevented from eavesdropping on the content of the assertion by employing confidentiality protection of the exchange using TLS. This ensures that an eavesdropper on the wire is unable to obtain information. However, this does not prevent legitimate protocol entities from obtaining information that they are not allowed to possess from assertions. Some assertion formats allow for the assertion to be encrypted, preventing unauthorized parties from inspecting the content.

Second, an authorization server may obtain an assertion that was created by a third-party token service and that token service may have placed attributes into the assertion. To mitigate potential privacy problems, prior consent for the release of such attribute information from the resource owner should be obtained. OAuth itself does not directly provide such capabilities, but this consent approval may be obtained using other identity management protocols or user consent interactions; it may also be obtained in an out-of-band fashion.

For the cases where a third-party token service creates assertions to be used for client authentication, privacy concerns are typically lower, since many of these clients are Web servers rather than individual devices operated by humans. If the assertions are used for client authentication of devices or software that can be closely linked to end users, then privacy protection safeguards need to be taken into consideration.

Further guidance on privacy friendly protocol design can be found in [RFC6973].

8.4. Privacy Considerations

An assertion may contain privacy-sensitive information and, to prevent disclosure of such information to unintended parties, should only be transmitted over encrypted channels, such as TLS. In cases where it is desirable to prevent disclosure of certain information to the client, the assertion (or portions of it) should be encrypted to the authorization server.

Deployments should determine the minimum amount of information necessary to complete the exchange and include only such information in the assertion. In some cases, the Subject identifier can be a value representing an anonymous or pseudonymous user, as described in Section 6.3.1.

9. IANA Considerations

This section registers three values, as listed in the subsections below, in the IANA "OAuth Parameters" registry established by RFC 6749 [RFC6749].

9.1. "assertion" Parameter Registration

- Name: assertion
- Parameter Usage Location: token request
- Change Controller: IESG
- Specification Document(s): RFC 7521

9.2. "client_assertion" Parameter Registration

- Name: client_assertion
- Parameter Usage Location: token request
- Change Controller: IESG
- Specification Document(s): RFC 7521

9.3. "client_assertion_type" Parameter Registration

- Name: client_assertion_type
- Parameter Usage Location: token request
- Change Controller: IESG
- Specification Document(s): RFC 7521

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <http://www.rfc-editor.org/info/rfc6749>.

10.2. Informative References

[OASIS.WS-Trust] Nadalin, A., Ed., Goodner, M., Ed., Gudgin, M., Ed., Barbir, A., Ed., and H. Granqvist, Ed., "WS-Trust", February 2009, <http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html>.

[OAUTH-DYN-REG]
Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", Work in Progress, draft-ietf-oauth-dyn-reg-29, May 2015.

[RFC6755] Campbell, B. and H. Tschofenig, "An IETF URN Sub-Namespace for OAuth", RFC 6755, DOI 10.17487/RFC6755, October 2012, <http://www.rfc-editor.org/info/rfc6755>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <http://www.rfc-editor.org/info/rfc6973>.

[RFC7522] Campbell, B., Mortimore, C., and M. Jones, "Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7522, DOI 10.17487/RFC7522, May 2015, <http://www.rfc-editor.org/info/rfc7522>.

[RFC7523] Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7523, DOI 10.17487/RFC7523, May 2015, <http://www.rfc-editor.org/info/rfc7523>.

Acknowledgements

The authors wish to thank the following people who have influenced or contributed to this specification: Paul Madsen, Eric Sachs, Jian Cai, Tony Nadalin, Hannes Tschofenig, the authors of the OAuth WRAP specification, and the members of the OAuth working group.

Authors' Addresses

Brian Campbell
Ping Identity

EMail: brian.d.campbell@gmail.com

Chuck Mortimore
Salesforce.com

EMail: cmortimore@salesforce.com

Michael B. Jones
Microsoft

EMail: mbj@microsoft.com
URI: http://self-issued.info/

Yaron Y. Goland
Microsoft

EMail: yarong@microsoft.com