Internet-Draft
Intended status: Standards Track
Expires: November 3, 2012

C. Mortimore, Ed.
Salesforce
M. Jones
Microsoft
B. Campbell
Ping
Y. Goland
Microsoft

May 2, 2012

OAuth 2.0 Assertion Profile
draft-ietf-oauth-assertions-03

Abstract

This specification provides a general framework for the use of assertions as client credentials and/or authorization grants with OAuth 2.0. It includes a generic mechanism for transporting assertions during interactions with a token endpoint, as wells as rules for the content and processing of those assertions. The intent is to provide an enhanced security profile by using derived values such as signatures or HMACs, as well as 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 flow and assertion content. Actual use requires implementation of a companion protocol binding specification. Additional profile documents provide standard representations in formats such as SAML and JWT.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 3, 2012.

Table of Contents

1. Requirements Notation and 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.

2. Overview

The OAuth 2.0 Authorization Protocol [I-D.ietf-oauth-v2] provides a method for making authenticated HTTP requests to a resource using an access token. 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 representing authorization by the resource owner or privileged administrator. Several authorization grant types are defined to support a wide range of client types and user experiences. OAuth also allows for the definition of new extension grant types to support additional clients or to provide a bridge between OAuth and other trust frameworks. Finally, OAuth allows the definition of additional authentication mechanisms to be used by clients when interacting with the authorization server.

In scenarios where security is at a premium one wants to avoid sending secrets across the Internet, even on encrypted connections. Instead one wants to send values derived from the secret that prove to the receiver that the sender is in possession of the secret without actually sending the secret. Typically the way derived values are created is by generating an assertion that is then either HMAC'ed or digitally signed using an agreed upon secret. By validating the HMAC or digital signature on the assertion, the receiver can prove to themselves that the entity that generated the assertion was in possession of the secret without actually communicating the secret directly.

This specification provides a general framework for the use of assertions as client credentials and/or authorization grants with OAuth 2.0. It includes a generic mechanism for transporting assertions during interactions with a token endpoint, as wells as rules for the content and processing of those assertions. The intent is to provide an enhanced security profile by using derived values such as signatures or HMACs, as well as 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 flow and assertion content. Actual use requires implementation of a companion protocol binding specification. Additional profile documents provide standard representations in formats such as SAML and JWT.

3. Authentication vs. Authorization

This specification provides a model for using assertions for authentication of an OAuth client during interactions with an Authorization Server, as well as the use of assertions as authorization grants. It is important to note that the use of assertions for client authentication is orthogonal and separable from using assertions as an authorization grant and can be used either in combination or in isolation. In addition, in scenarios when assertion based authentication and authorization are used in combination, the assertion format and processing may be redundant; under such circumstances, the protocol may be optimized to present a single assertion.

4. Transporting Assertions

This section defines generic HTTP parameters for transporting assertions during interactions with a token endpoint.

4.1. Using Assertions for Client Authentication

In scenarios where one wants to avoid sending secrets, one wants to send values derived from the secret that prove to the receiver that the sender is in possession of the secret without actually sending the secret.

For example, a client can establish a secret using an out-of-band mechanism with a resource server. As part of this out-of-band communication the client and resource server agree that the client will authenticate itself using an assertion with agreed upon parameters that will be signed by the provisioned secret. Later on, the client might send an access token request to the token endpoint for the resource server that includes an authorization code, as well as a "client_assertion" that is signed with the previously agreed key and parameters. The "client_assertion" proves to the token endpoint the identity of the client and the authorization code provides the link to the end-user authorization.

The following section defines the use of assertions as client credentials as an extension of Section 2.3 of OAuth 2.0 [I-D.ietf-oauth-v2]. When using assertions as client credentials,

the client includes the assertion and related information using the following HTTP request parameters:

client_id OPTIONAL. The client identifier as described in Section 2.2 of OAuth 2.0 [I-D.ietf-oauth-v2]. When present, the "client_id" MUST identify the client to the authorization server.

client_assertion_type REQUIRED. The format of the assertion as defined by the authorization server. The value MUST 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. The serialization MUST be encoded for transport within HTTP forms. It is RECOMMENDED that base64url be used.

The following non-normative example demonstrates a client authenticating using an assertion during an Authorization Code Access Token Request as defined in Section 4.1.3 of OAuth 2.0 [I-D.ietf-oauth-v2] (with 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=i1WsRn1uB1&
client_id=s6BhdRkqt3&
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.

4.1.1. Error Responses

If an assertion is invalid for any reason or if more than one client authentication mechanism is used, the Authorization Server MUST construct an error response as defined in OAuth 2.0 [I-D.ietf-oauth-v2]. 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":"Multiple Credentials Not Allowed"
}
```

4.2. Using Assertions as Authorization Grants

An assertion can be used to request an access token when a client wishes to utilize an existing trust relationship. This may be done through the semantics of (and a digital signature/HMAC calculated over) the assertion, and expressed to the authorization server through an extension authorization grant type. The processes by which authorization is previously granted, and by which the client obtains the assertion prior to exchanging it with the authorization server, are out of scope.

The following defines the use of assertions as authorization grants as an extension of OAuth 2.0 [I-D.ietf-oauth-v2], Section 4.5. 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 MUST be an absolute URI.

assertion REQUIRED. The assertion being used as an authorization grant. Specific serialization of the assertion is defined by profile documents. The serialization MUST be encoded for transport within HTTP forms. It is RECOMMENDED that base64url be used.

scope OPTIONAL. The requested scope as described in Section 3.3 of OAuth 2.0 [I-D.ietf-oauth-v2]. When exchanging assertions for access tokens, the authorization for the token has been previously granted through some other mechanism. As such, the requested scope SHOULD be equal or lesser than the scope originally granted to the authorized accessor. If the scope parameter and/or value is omitted, the scope SHOULD be treated as equal to the scope originally granted to the authorized accessor. The Authorization Server SHOULD limit the scope of the issued access token to be equal or lesser than the scope originally granted to the authorized accessor.

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

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

client_id=s6BhdRkqt3&
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 tokens that exceed that lifetime 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 limited 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.

4.2.1. Error Responses

If an assertion is not valid or has expired, the Authorization Server MUST construct an error response as defined in OAuth 2.0 [I-D.ietf-oauth-v2]. 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"
}
```

5. Assertion Content and Processing

This section provides a general content and processing model for the use of assertions in OAuth 2.0 [I-D.ietf-oauth-v2].

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 The unique identifier for the entity that issued the assertion. Generally this is the entity that holds the keying material used to generate the assertion. In some use cases Issuers may be either OAuth Clients (when assertions are self-asserted) or a Security Token Service (STS) (an entity capable of issuing, renewing, transforming and validating of security tokens).

Principal A unique identifier for the subject of the assertion. When using assertions for client authentication, the Principal SHOULD be the "client_id" of the OAuth client. When using assertions as an authorization grant, the Principal MUST identify an authorized accessor for whom the access token is being requested (typically the resource owner, or an authorized delegate).

Audience A URI that identifies the party intended to process the assertion. The audience SHOULD be the URL of the Token Endpoint as defined in Section 3.2 of OAuth 2.0 [I-D.ietf-oauth-v2].

Issued At The time at which the assertion was issued. While the serialization may differ by assertion format, this is always 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, this is always 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 that that entity or any other entity will 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 MUST identify the entity that issued the assertion as recognized by the Authorization Server. If an assertion is self-asserted, the Issuer SHOULD be the "client_id".

- The assertion SHOULD contain a Principal. The Principal MUST identify an authorized accessor for whom the access token is being requested (typically the resource owner, or an authorized delegate). When the client is acting on behalf of itself, the Principal SHOULD be the "client_id".

- The assertion MUST contain an Audience that identifies the Authorization Server as the intended audience. The Authorization Server MUST verify that it is an intended audience for the assertion. The Audience SHOULD be the URL of the Authorization Server's Token Endpoint.

- The assertion MUST contain an Expires At entity that limits the time window during which the assertion can be used. The authorization server MUST verify that the expiration time has not passed, subject to allowable clock skew between systems. The authorization server SHOULD 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 assertion MAY contain an Assertion ID. An Authorization Server MAY dictate that Assertion ID is mandatory.

- The Authorization Server MUST validate the assertion's signature in order to verify the Issuer of the assertion. The algorithm used to validate the assertion, and the mechanism for designating the secret used to generate the assertion, are beyond the scope of this specification.

6. Specific Assertion Format and Processing Rules

The following clarifies the format and processing rules defined in Section 4 and Section 5 for a number of common use cases:

6.1. Client Authentication

When a client authenticates to a token service using an assertion, it SHOULD do so according to Section 4.1. The following format and processing rules apply.

- The "client_assertion_type" HTTP parameter MUST identify the assertion format being used for authentication.

- The "client_assertion" HTTP parameter MUST contain the serialized assertion in a format indicated by the "client_assertion_type" parameter.

- The Principal SHOULD be the "client_id".

- The Issuer of the assertion MUST identify the entity that issued the assertion as recognized by the Authorization Server. If the assertion is self-asserted, the Issuer SHOULD be the "client_id".

- The Audience of the assertion MUST identify the Authorization Server and SHOULD be the URL of the Token Endpoint.

- The Authorization Server MUST validate the assertion's signature in order to verify the Issuer of the assertion.

The following non-normative example demonstrates the use of a client authentication using an assertion during an Authorization Code Access Token Request as defined in Section 4.1.3 of OAuth 2.0 [I-D.ietf-oauth-v2] (with 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=i1WsRn1uB1&
client_id=s6BhdRkqt3&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth
%3Aclient-assertion-type%3Asaml2-bearer&
client_assertion=PHNhb...[omitted for brevity]...ZT4
```

6.2. Client Acting on Behalf of Itself

When a client is accessing resources on behalf of itself, it SHOULD do so in a manner analogous to the Client Credentials flow defined in Section 4.4 of OAuth 2.0 [I-D.ietf-oauth-v2]. 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 4.1, with the addition of a grant_type parameter. The following format and processing rules apply.

- The grant_type HTTP request parameter MUST be "client_credentials".

- The "client_assertion_type" HTTP parameter MUST identify the assertion format.

- The "client_assertion" HTTP parameter MUST contain the serialized assertion as in a format indicated by the "client_assertion_type" parameter.

- The Issuer of the assertion MUST identify the entity that issued the assertion as recognized by the Authorization Server. If the assertion is self-asserted, the Issuer SHOULD be the "client_id". If the assertion was issued by a Security Token Service (STS), the Issuer SHOULD identify the STS as recognized by the Authorization Server.

- The Principal SHOULD be the "client_id".

- The Audience of the assertion MUST identify the Authorization Server and SHOULD be the URL of the Token Endpoint.

- The Authorization Server MUST validate the assertion's signature in order to verify the Issuer of the assertion.

The following non-normative example demonstrates the use of a sample assertion being used for a Client Credentials Access Token Request as defined in Section 4.4.2 of OAuth 2.0 [I-D.ietf-oauth-v2] (with line breaks for display purposes only):

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

client_id=s6BhdRkqt3&
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 SHOULD be treated as using an assertion as an Authorization Grant according to Section 4.2. The following format and processing rules apply.

- The grant_type HTTP request parameter MUST indicate the assertion format.

- The assertion HTTP parameter MUST contain the serialized assertion as in a format indicated by the grant_type parameter.

- The Issuer of the assertion MUST identify the entity that issued the assertion as recognized by the Authorization Server. If the assertion is self-asserted, the Issuer SHOULD be the "client_id". If the assertion was issued by a Security Token Service (STS), the Issuer SHOULD identify the STS as recognized by the Authorization Server.

- The Principal MUST identify an authorized accessor for whom the access token is being requested (typically the resource owner, or an authorized delegate).

- The Audience of the assertion MUST identify the Authorization Server and MAY be the URL of the Token Endpoint.

- The Authorization Server MUST validate the assertion's signature in order to verify the Issuer of the assertion.

The following non-normative example demonstrates the use of a client authenticating using an assertion during an Authorization Code Access Token Request as defined in Section 4.1.3 of OAuth 2.0 [I-D.ietf-oauth-v2] (with line breaks for display purposes only):

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

client_id=s6BhdRkqt3&
grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Asaml2-bearer&
assertion=PHNhbWxwOl...[omitted for brevity]...ZT
```

6.4. Client Acting on Behalf of an Anonymous User

When a client is accessing resources on behalf of an anonymous user, the following format and processing rules apply.

- The grant_type HTTP request parameter MUST indicate the assertion format.

- The assertion HTTP parameter MUST contain the serialized assertion as in a format indicated by the grant_type parameter.

- The Issuer of the assertion MUST identify the entity that issued the assertion as recognized by the Authorization Server. If the assertion is self-asserted, the Issuer SHOULD be the "client_id". If the assertion was issued by a Security Token Service (STS), the Issuer SHOULD identify the STS as recognized by the Authorization Server.

- The Principal SHOULD indicate to the Authorization Server that the client is acting on-behalf of an anonymous user as defined by the Authorization Server. It is implied that authorization is based upon additional criteria, such as additional attributes or claims provided in the assertion. For example, a client may 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.

- The Audience of the assertion MUST identify the Authorization Server and MAY be the URL of the Token Endpoint.

- The Authorization Server MUST validate the assertion's signature in order to verify the Issuer of the assertion.

7. Security Considerations

Authorization Providers concerned with preventing replay attacks may choose to implement using replay detection using a combination of the Assertion ID and Issued At/Expires At attributes. Previously processed assertions MAY be de-duped and 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.

Authorization Servers SHOULD consider the amount of information exposed in error responses, and the risk associated with exposing details of specific processing errors. In addition, Authorization Servers SHOULD prevent timing attacks related to cryptographic processing of the assertion.

There are no additional security considerations beyond those described within OAuth 2.0 [I-D.ietf-oauth-v2], Section 11.

8. IANA Considerations

8.1. assertion Parameter Registration

The following is the parameter registration request, as defined in The OAuth Parameters Registry of The OAuth 2.0 Authorization Protocol [I-D.ietf-oauth-v2], for the "assertion" parameter:

- Parameter name: assertion

- Parameter usage location: client authentication, token request
- Change controller: IETF
- Specification document(s): [[this document]]

8.2. client_assertion Parameter Registration

The following is the parameter registration request, as defined in The OAuth Parameters Registry of The OAuth 2.0 Authorization Protocol [I-D.ietf-oauth-v2], for the "client_assertion" parameter:

- Parameter name: "client_assertion"
- Parameter usage location: client authentication, token request
- Change controller: IETF
- Specification document(s): [[this document]]

8.3. client_assertion_type Parameter Registration

The following is the parameter registration request, as defined in The OAuth Parameters Registry of The OAuth 2.0 Authorization Protocol [I-D.ietf-oauth-v2], for the "client_assertion_type" parameter:

- Parameter name: "client_assertion_type"
- Parameter usage location: client authentication, token request
- Change controller: IETF
- Specification document(s): [[this document]]

9. Normative References

[I-D.ietf-oauth-v2] Hammer-Lahav, E., Recordon, D., and D. Hardt, "The OAuth 2.0 Authorization Framework", draft-ietf-oauth-v2-26 (work in progress), May 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Appendix A. Acknowledgements

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

Appendix B. Document History

[[to be removed by RFC editor before publication as an RFC]]

draft-ietf-oauth-assertions-03

- updated reference to draft-ietf-oauth-v2 from -25 to -26

draft-ietf-oauth-assertions-02

- Added text about limited lifetime ATs and RTs per http://www.ietf.org/mail-archive/web/oauth/current/msg08298.html.
- Changed the line breaks in some examples to avoid awkward rendering to text format. Also removed encoded '=' padding from a few examples because both known derivative specs, SAML and JWT, omit the padding char in serialization/encoding.
- Remove section 7 on error responses and move that (somewhat modified) content into subsections of section 4 broken up by authn/authz per http://www.ietf.org/mail-archive/web/oauth/current/msg08735.html.
- Rework the text about "MUST validate ... in order to establish a mapping between ..." per http://www.ietf.org/mail-archive/web/oauth/current/msg08872.html and http://www.ietf.org/mail-archive/web/oauth/current/msg08749.html.
- Change "The Principal MUST identify an authorized accessor. If the assertion is self-issued, the Principal SHOULD be the client_id" in 6.1 per http://www.ietf.org/mail-archive/web/oauth/current/msg08873.html.
- Update reference in 4.1 to point to 2.3 (rather than 3.2) of oauth-v2 (rather than self) http://www.ietf.org/mail-archive/web/oauth/current/msg08874.html.
- Move the "Section 3 of" out of the xref to hopefully fix the link in 4.1 and remove the client_id bullet from 4.2 per

http://www.ietf.org/mail-archive/web/oauth/current/msg08875.html.

- Add ref to Section 3.3 of oauth-v2 for scope definition and remove some then redundant text per http://www.ietf.org/mail-archive/web/oauth/current/msg08890.html.

- Change "The following format and processing rules SHOULD be applied" to "The following format and processing rules apply" in sections 6.x to remove conflicting normative qualification of other normative statements per http://www.ietf.org/mail-archive/web/oauth/current/msg08892.html.

- Add text the client_id must id the client to 4.1 and remove similar text from other places per http://www.ietf.org/mail-archive/web/oauth/current/msg08893.html.

- Remove the MUST from the text prior to the HTTP parameter definitions per http://www.ietf.org/mail-archive/web/oauth/current/msg08920.html.

- Updated examples to use grant_type and client_assertion_type values from the OAuth SAML Assertion Profiles spec.

Authors' Addresses

Chuck Mortimore (editor)
Salesforce.com

Email: cmortimore@salesforce.com

Michael B. Jones
Microsoft

Email: mbj@microsoft.com

Brian Campbell
Ping Identity Corp.

Email: brian.d.campbell@gmail.com

Yaron Y. Goland
Microsoft

Email: yarong@microsoft.com