Network Working Group D. Catalano, Ed.
Internet-Draft Oracle
Intended status: Standards Track M. Machulak
Expires: January 21, 2015 Cloud Identity
July 20, 2014

User-Managed Access (UMA) Claim Profiles Framework
draft-catalano-oauth-umaclaim-00

Abstract

User-Managed Access (UMA) is a profile of OAuth 2.0. UMA defines how resource owners can control protected-resource access by clients operated by arbitrary requesting parties, where the resources reside on any number of resource servers, and where a centralized authorization server governs access based on resource owner policy. This specification defines a generic framework for building UMA claim profiles that can be used by client applications to obtain the necessary authorization to access protected resources. This revision of the specification is part of V0.9.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

User-Managed Access [UMA] is a profile of OAuth 2.0. UMA defines how resource owners can control protected-resource access by clients operated by arbitrary requesting parties, where the resources reside on any number of resource servers, and where a centralized authorization server governs access based on resource owner policy. This specification defines a generic framework for building UMA claim profiles that can be used by client applications to obtain the necessary authorization to access protected resources.

Using the framework defined in this specification, UMA deployers can add new claim profiles to meet requirements of particular deployments

of UMA. Profiles built on this framework will give both authorization servers and clients certain interoperability and ease of development properties. This specification also provides some sample profiles that build on the framework. Deployers can build on the framework directly or on these sample profiles, as they wish, in order to promote interoperability in their specific environments.

The framework introduces different interaction patterns that the client and authorization server can use, and different roles they can play, in order to gather claims about the requesting party:

o The ?delivery? interaction pattern leverages a ?claims-aware client? that is able to deliver claims about the requesting party (or information about how to get claims) directly to the authorization server. The information delivered can be an identity or claims token, data that aids in discovery of a claims endpoint, etc., depending on the client's role outside of UMA as a federated identity provider, a federated relying party, an application integrated with a native identity repository, etc. The authorization server then plays the role of a ?claims receiver? (and/or activates a ?claims connector? based on the information, for gathering claims itself without requesting party involvement).

o The ?redirect? pattern assumes a ?claims-unaware client? whose only option (other than failing entirely) is to redirect an end-user requesting party to the authorization server. On receiving the end user, the authorization server activates a ?claims connector? for gathering the necessary claims with the user's involvement, using any method or combination of methods. In this role, the authorization server may be a relying party in a federated identity interaction, or it may connect to a directory or other user repository. After the claims-gathering process, the authorization server redirects the user back to the client.

The profiles defined based on both interaction patterns are as follows:

o Delivery:

   * Client delivers a SAML assertion to the authorization server

   * Client delivers OpenID Connect user claims to the authorization server

   * Client delivers custom user claims to the authorization server

   * Client delivers custom and OpenID Connect user claims to the authorization server

- Redirect:

   * Client redirects end-user requesting party to the authorization server

In all cases, it is assumed that the authorization server evaluates the resource owner's policy for a particular resource set based, at least in part, on the supplied claims. An authorization server MAY support any claim profiles defined in this specification, and SHOULD advertise its conformance to tbe profiles it supports in its configuration data.

1.1. 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].

Unless otherwise noted, all the protocol properties and values are case sensitive.

2. Generic Framework for Claim Profiles

When a client asks an authorization server to associate authorization data with a requesting party token (RPT) so that the client can successfully access a resource on behalf of the requesting party operating it, the authorization can respond in three main ways: either it can deny the client's request outright, or it can accede to the request outright, or it can respond that it needs claims in order to assess whether suitability of adding the needed authorization data. The authorization server has an opportunity, when it returns a "need_claims" response, to provide further instructions and hints to the client in this response. This section defines extensions to [UMA] that support these instructions and hints.

The authorization request endpoint in the authorization API presented by the authorization server is extended to accept JSON-encoded claims-related data in the body of the request. Along with the "rpt" and "ticket" properties that already need to be provided, a "claims" property appears in addition.

Common message flow:

1. The client sends the claims type and its claims directly to the AS

```
POST /rpt_authorization HTTP/1.1
        Host: www.nuveam.com
        Authorization: Bearer jwfLG53^sad$#f
        ...

{
    "rpt": "sbjsbhs(/SSJHBSUSSJHVhjsgvhsgvshgsv",
    "ticket": "016f84e8-f9b9-11e0-bd6f-0021cc6004de",
    "claims": [
        {
            "type": "CLAIM_TYPE_AS_STRING",
            "value": {SPECIFIC_SET_OF_CLAIMS_AS_JSON_OBJECT}
        }
    ]
}
```

Importantly, the claims MUST be an array of JSON objects. The type field MUST have a String value indicating the type of claims-related data, while the value field MUST be a JSON object specific to that type of claims-related data.

2. The authorization server informs the client that authorization data has been added

```
HTTP/1.1 201 Created
    Content-Type: application/json;charset=UTF-8

    {
        "rpt":"e6b09a4f434a6a47a65a198652df381a"
    }
```

3. The authorization server informs the client that further claims should be provided to the authorization request endpoint:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
    "need_claims":[
    {
        "type":"CLAIM_TYPE_AS_STRING",
        "name":"",
        "value":""
    }]
}
```

4. The authorization server informs the client that further claims should be provided (the example below is for SAML assertion)

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
    "need_claims":[
    {
        "type":"claim-client-assertion-saml-1.0",
        "name":"",
        "value":""
    }
    ]
}
```

5. The authorization server informs the client that the authorization data cannot be added.

```
HTTP/1.1 403 Forbidden
Content-Type: application/json;charset=UTF-8

{
    "error":"not_authorized_permission",
    "error_description":"Authorization data cannot be added."
}
```

2.1. Client Provides Custom User Attributes

TYPE = "custom"

VALUE = {custom defined}

In the most trivial setting where the AS and the Client are collocated and have an established trust relationship (in particular, the AS trusts information that it receives from the client), then the client can be preconfigured to provide the required information to the AS based on a custom schema. We provide the most trivial example below, where the client application provides a user's identifier (in this case email) to the AS and such identifier is used for policy evaluation.

Example:

```
   POST /rpt_authorization HTTP/1.1
   Host: www.nuveam.com
   Authorization: Bearer jwfLG53^sad$#f
   ...

   {
       "rpt": "sbjsbhs(/SSJHBSUSSJHVhjsgvhsgvshgsv",
       "ticket": "016f84e8-f9b9-11e0-bd6f-0021cc6004de",
       "claims": [
       {
           "type": "ci-nuveam-claims",
           "value": { "email": "bob@company.example.com" }
       }
       ]
   }
```

Another example is where the client provides a richer set of attributes directly to the AS and these attributes are used for policy evaluation. Importantly, it is the AS that decides which attributes are used for policy evaluation and which are not.

Example:

```
POST /rpt_authorization HTTP/1.1
Host: www.nuveam.com
Authorization: Bearer jwfLG53^sad$#f
...

{
    "rpt": "sbjsbhs(/SSJHBSUSSJHVhjsgvhsgvshgsv",
    "ticket": "016f84e8-f9b9-11e0-bd6f-0021cc6004de",
    "claims": [
        {
        "type": "ci-nuveam-claims",
        "value": { "email": "bob@gmail.com", "roles": [ "manager", "admin" ] }
        }
    ]

}
```

We provide an example of a reply below (standard UMA reply):

Example:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "rpt": "sbjsbhs(/SSJHBSUSSJHVhjsgvhsgvshgsv2"
}
```

In case of custom attributes, the client does not necessarily use any specific protocol for obtaining user attributes. It can use a pre-established relationship with the AS to provide the required set of attributes.

2.2. Client Acts as SAML Assertion Conveyor

TYPE = "claim-client-assertion-saml-1.0"

VALUE = {base64-encoded SAML assertion}

In this setting the AS and the Client have a pre-established trust relationship. The client may provide the AS with a SAML assertion that can be used for policy evaluation. We provide an example of the request below.

Example:

```
POST /rpt_authorization HTTP/1.1
Host: www.nuveam.com
Authorization: Bearer jwfLG53^sad$#f
...

{
    "rpt": "sbjsbhs(/SSJHBSUSSJHVhjsgvhsgvshgsv",
    "ticket": "016f84e8-f9b9-11e0-bd6f-0021cc6004de",
    "claims": [
    {
        "type": "claim-client-assertion-saml-1.0",
        "value": {
        "saml_assertion": "PHNhbWxwOl...[omitted for brevity]...ZT"
        }
    }
    ]

}
```

2.3. Client Acts as OpenID Connect Claims Conveyor

TYPE = "claim-client-claims-oidc-1.0"

VALUE = {set of oidc reserved claims}

In this setting the AS and the Client have a pre-established trust relationship. The client may provide the AS with a OpenID Connect user claims that can be used for policy evaluation. We provide an example of the request made by the client to the Authorization Server below.

Example:

```
POST /rpt_authorization HTTP/1.1
Host: www.nuveam.com
Authorization: Bearer jwfLG53^sad$#f
...

{
    "rpt": "sbjsbhs(/SSJHBSUSSJHVhjsgvhsgvshgsv",
    "ticket": "016f84e8-f9b9-11e0-bd6f-0021cc6004de",
    "claims": [
    {
        "type": "claim-client-claims-oidc-1.0",
        "value": {
        "sub": "248289761001"
        "name": "Jane Doe",
        "given_name": "Jane",
        "family_name": "Doe",
        "email": "joedoe@example.com"
        "email_verified": true,
        }
    }
    ]
}
```

2.4. Hybrid Approach: Client Acts as Custom Claims Conveyor and OpenID Connect Claims Conveyor

TYPE = "custom"

VALUE = {custom defined}

TYPE = "claim-client-claims-oidc-1.0"

VALUE = {set of oidc reserved claims}

In this setting the AS and the Client have a pre-established trust relationship. The client may provide the AS with custom claims as well as with OpenID Connect user claims that can be used for policy evaluation. We provide an example of the request below.

Example:

```
POST /rpt_authorization HTTP/1.1
Host: www.nuveam.com
Authorization: Bearer jwfLG53^sad$#f
...
{
    "rpt": "sbjsbhs(/SSJHBSUSSJHVhjsgvhsgvshgsv",
    "ticket": "016f84e8-f9b9-11e0-bd6f-0021cc6004de",
    "claims": [
    {
        "type": "ci-nuveam-claims",
        "value": { "roles": ["manager", "admin" }
            },
    {
        "type": "claim-client-claims-oidc-1.0",
        "value": { "email": "bob@gmail.com" }
            }
    ]
}
```

2.5. Client Redirects Requesting Party to AS

TYPE = "claim-client-redirect-1.0"

VALUE = {value of the scope at AS}

The redirect UMA profile defines a Requesting Party Claims Endpoint that the Authorization Server has to support. This endpoint is advertised in the Authorisation Server Configuration Data as defined by the UMA specification [UMA]. The requesting party claims endpoint is used by the Authorization Server to interact with the requesting party and not with the client application. The authorization server can first verify the identity of the requesting party or it may engage the requesting party in claims gathering flow. For example, the AS may decide based on the authentication process that it has enough information to evaluate a policy or it may require the requesting party to provide further claims, e.g. using an existing identity federation protocol. For example, after landing at this endpoint the requesting party may be further redirected to the source of claims (e.g. SAML IDP or the OpenID Connect Identity Provider).

2.5.1. Requesting Party Claims Endpoint

In redirect UMA profile, the configuration data has to be extended with the following property.

requesting_party_claims_endpoint
REQUIRED. The endpoint URI at which the authorization server interacts with the end-user requesting party to obtain the necessary user-claims that will be used during policy evaluation process.

Example of authorization server configuration extended with requesting party claims endpoint:

```
{
"version":"1.0",
"issuer":"https://example.com",
"pat_profiles_supported":["bearer"],
"aat_profiles_supported":["bearer"],
"rpt_profiles_supported":["bearer"],
"pat_grant_types_supported":["authorization_code"],
"aat_grant_types_supported":["authorization_code"],
"claim_profiles_supported":["openid"],
"dynamic_client_endpoint":"https://as.example.com/dyn_client_reg_uri",
"token_endpoint":"https://as.example.com/token_uri",
"user_endpoint":"https://as.example.com/user_uri",
"resource_set_registration_endpoint":"https://as.example.com/rs/rsrc_uri",
"introspection_endpoint":"https://as.example.com/rs/status_uri",
"permission_registration_endpoint":"https://as.example.com/rs/perm_uri",
"rpt_endpoint":"https://as.example.com/client/rpt_uri",
"authorization_request_endpoint":"https://as.example.com/client/authz_uri",
"requesting_party_claims_endpoint":"https://as.example.com/rp/claims_uri"
}
```

2.5.2. Message Flow

Message flow:

1. Client asks for new authorization data to be added to an existing RPT

```
POST /rpt_authorization HTTP/1.1
        Host: www.nuveam.com
        Authorization: Bearer jwfLG53^sad$#f
        ...

        {
        "rpt": "sbjsbhs(/SSJHBSUSSJHVhjsgvhsgvshgsv",
        "ticket": "016f84e8-f9b9-11e0-bd6f-0021cc6004de"
        }
```

2. AS tells the client to redirect the user to the Requesting Party Claims Endpoint and includes the scope parameter in the value of the response

```
HTTP/1.1 403 Forbidden
    Content-Type: application/json

    {
      "need_claims":[
        {
          "type":"redirect_required",
          "name":"Redirect Required",
          "value":"699faf5bf2869838e992d57756bc6f"
        }
      ]
    }
```

3. Client redirects the user to the Requesting Party Claims Endpoint and includes the scope parameter in the request

```
HTTP/1.1 302 Found
    Location: https://www.nuveam.com/uma/rp_claims?scope=699faf5bf2869838e992
    d57756bc6f&redirect_uri=http%3A%2F%2Fwww.umaapp.com%%2Fredirect&client_id=
    ca4453936fa5fda2110b9e589d61ab37&state=32455ddsafas
```

After the user is redirected to the AS, the claims for the user are gathered according to one of the defined protocols, such as SAML, OpenID Connect or any other protocol implemented by an UMA-compliant Authorisation Server. Furthermore, the AS is free to obtain the information from a local or remote LDAP, Active Directory or any other user datastore (e.g. SQL or NoSQL-based datastore).

4. AS informs the client that new authorization can be added and the client is free to request a new RPT

```
HTTP/1.1 302 Found
    Location: https://www.umaapp.com/redirect?access=granted&state=32455ddsafas
```

5. AS informs the client that authorization data cannot be added

```
HTTP/1.1 302 Found
    Location: https://www.umaapp.com/redirect?access=denied&state=32455ddsafas
```

2.5.3. Examples

In this section, we discuss three examples:

1. User is redirected to an OIDC Provider;

2. User is redirected to a SAML IDP;

3. User's authentication is sufficient for policy evalutation.

2.5.3.1. Authorization Server Acts as OpenID Connect Relying Party

In this claim profile example, the Authorisation Server acts as an OIDC compliant RP. This flow is used in case the policies for a particular resource set use any of the existing reserved OIDC claims. Importantly, it is the AS that determines if OIDC claims should be used for policy evaluation. This information is not shared with the client application.

During this flow the AS acts according to the OpenID Connect protocol and this is outside of the UMA specification.

2.5.3.2. Authorization Server Acts as SAML Relying Party

In this claim profile example, the Authorisation Server acts as an SAML compliant Service Provider. This flow is used in case the policies for a particular resource set require the use of the SAML protocol. Importantly, it is the AS that determines if the SAML protocol should be used for policy evaluation. This information is not shared with the client application.

During this flow the AS acts according to the SAML protocol and this is outside of the UMA specification.

2.5.3.3. Authorization Server pulls Claim from local user store

In this claim profile example and after successful authentication of the RP, the AS can pull the required user attributes from a local user datastore (e.g. LDAP, Active Directory, and other SQL and NoSQL-datastores). This information can be used for policy evaluation.

2.6. IANA Considerations

This document makes no request of IANA.

2.7. Acknowledgments

The current editor of this specification is Domenico Catalano of Oracle. The following people are co-authors:

o Maciej Machulak, Cloud Identity Ltd

o Thomas Hardjono, MIT

o Eve Maler, ForgeRock

Additional contributors to this specification include the Kantara UMA Work Group participants, a list of whom can be found at [UMAnitarians].

2.8. Issues

Issues are captured at the project's GitHub site ([1]).

3. References

3.1. Normative References

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

[UMA] Hardjono, T., Ed., "User-Managed Access (UMA) Profile of OAuth 2.0", December 2013, <http://docs.kantarainitiative.org/uma/draft-uma-core.html>.

3.2. Informative References

[UMAnitarians]
Maler, E., "UMA Participant Roster", July 2014, <http://kantarainitiative.org/confluence/display/uma/Participant+Roster>.

3.3. URIs

[1] https://github.com/xmlgrrl/UMA-Specifications/issues

[2] http://kantarainitiative.org/confluence/display/uma/UMA+1.0+Core+Protocol

Appendix A. Document History

NOTE: To be removed by RFC editor before publication as an RFC.

See [2] for a list of code-breaking and other major changes made to this specification at various revision points.

Authors' Addresses

Domenico Catalano (editor)
Oracle

Email: domenico.catalano@oracle.com

Maciej Machulak
Cloud Identity

Email: maciej.machulak@cloudidentity.co.uk