User-Managed Access (UMA) Working Group

Eve Maler, WG chair

eve.maler@forgerock.com | @xmlgrrl

15 May 2018

http://tinyurl.com/umawg/ | @UMAWG
First: Shall we do an UMA explainer?
News from the last year, in context

- **Mar ‘15:** UMA V1.0 ratified as Recommendations
- **Dec ‘15:** UMA V1.0.1 ratified as Recommendations
- **Jul ‘17:** 1st Public Comment/Review period ends
- **Sep ‘17:** 2nd Public Comment/Review period ends
- **Jan ‘18:** Final Recommendations published
- **Jan ‘18:** Draft UMA Business Model Report published
- **Feb ‘18:** Charter update
- **May ‘18:** Keycloak joins UMA2 vendors Gluu and ForgeRock

Additional notes:
- ** Specs refactored, over 100 issues closed, lots of implementation input received, Disposition of Comments doc written…**
Some use cases/ecosystems involving UMA

• Financial
  • UK Pensions Dashboard Project / OIX / Origo
  • Examining suitability for a set of Open Banking use cases

• IoT
  • “ACE actors” architecture identifies requirements for authorization to an RqP

• Healthcare
  • Profiled in Health Relationship Trust (HEART) at OpenID Foundation
  • Part of the new OpenMedReady framework, along with HEART
The UMA business model defines how the UMA protocol enables a license-based model for controlling access rights to personal digital assets.
Key benefits of UMA to service providers

True security of delegated access

Scalability of resource permissioning

API-first protection strategy

Fosters control for compliance and trust
### Key benefits of UMA to consumers

<table>
<thead>
<tr>
<th>Benefit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constrained party-to-party delegation</td>
<td>Allows controlled access and management of data.</td>
</tr>
<tr>
<td>Granting consent without external influence</td>
<td>Enables granting of consent independently of outside influence.</td>
</tr>
<tr>
<td>Centralized monitoring and management</td>
<td>Provides a centralized approach to monitoring and managing access controls.</td>
</tr>
<tr>
<td>Control of consents at a fine grain</td>
<td>Offers fine-grained control over access permissions.</td>
</tr>
</tbody>
</table>
OAuth, OIDC, and UMA2: breaking it down
OAuth is for constrained delegation to apps

It has helped to kill the “password anti-pattern”
OAuth is for constrained delegation to apps
It has helped to kill the “password anti-pattern”
OAuth is for constrained delegation to apps
It has helped to kill the “password anti-pattern”

Resource owner

App gets consent based on the API scopes it requested: it has its own identity distinct from the RO’s

Authorization server

Client

Authorizes (consents) at runtime after authenticating, at the AS

Resource server
OAuth is for constrained delegation to apps
It has helped to kill the “password anti-pattern”

Authorizes (consents) at run time after authenticating, at the AS
App gets consent based on the API scopes it requested; it has its own identity distinct from the RO’s
Standard OAuth endpoints for authorization and access token issuance
OAuth is for constrained delegation to apps
It has helped to kill the “password anti-pattern”

Resource owner

Client

Authorization server

Resource server

Authorizes (consents) at run time after authenticating, at the AS

App gets consent based on the API scopes it requested; it has its own identity distinct from the RO’s

Standard OAuth endpoints for authorization and access token issuance

Some number of API endpoints that deliver the data or other value-add
OAuth is for constrained delegation to apps
It has helped to kill the “password anti-pattern”
OAuth is for constrained delegation to apps
It has helped to kill the “password anti-pattern”

Authorization Request (A)
Authorization Grant (B)
Authorization Grant (C)
Access Token (D)
Access Token (E)
Protected Resource

Resource owner
Client
Authorization server
Resource server

This can come with a refresh token for renewal without the RO’s intervention

Authorizes (consents) at run time after authenticating, at the AS
App gets consent based on the API scopes it requested; it has its own identity distinct from the RO’s
Standard OAuth endpoints for authorization and access token issuance
Some number of API endpoints that deliver the data or other value-add
OAuth is for constrained delegation to apps
It has helped to kill the “password anti-pattern”

- Authorization Request
- Authorization Grant
- Authorization Grant (B)
- Authorization Grant (D)
- Authorization Grant (E)
- Access Token
- Access Token
- Access Token
- Access Token
- Protected Resource
- Resource server
- Resource server
- Resource owner

The RO can revoke the token to withdraw authorization (consent)

Authorizes (consents) at run time after authenticating, at the AS

App gets consent based on the API scopes it requested; it has its own identity distinct from the RO’s

Standard OAuth endpoints for authorization and access token issuance

Some number of API endpoints that deliver the data or other value-add

This can come with a refresh token for renewal without the RO’s intervention

Authorizes (consents) at run time after authenticating, at the AS

App gets consent based on the API scopes it requested; it has its own identity distinct from the RO’s

Standard OAuth endpoints for authorization and access token issuance

Some number of API endpoints that deliver the data or other value-add

This can come with a refresh token for renewal without the RO’s intervention
OpenID Connect does modern-day federation
It is an OAuth-protected identity API, plus a bit more
OpenID Connect does modern-day federation

It is an OAuth-protected identity API, plus a bit more

- Resource owner = Federation user
- Client
- Authorization server
- Resource server
OpenID Connect does modern-day federation
It is an OAuth-protected identity API, plus a bit more

- Resource owner = Federation user
- Client = Relying party
- Authorization server
- Resource server
OpenID Connect does modern-day federation

It is an OAuth-protected identity API, plus a bit more

- **Resource owner** = Federation user
- **Client** = Relying party
- **Authorization server** = Identity provider ("OpenID provider")
- **Resource server**

---

**Legend:**
- UMA
- KI
OpenID Connect does modern-day federation
It is an OAuth-protected identity API, plus a bit more

Resource owner = Federation user

Client = Relying party

Token endpoint typically delivers an “ID token” similar to a SAML assertion

Authorization server = Identity provider (“OpenID provider”)

Resource server
OpenID Connect does modern-day federation
It is an OAuth-protected identity API, plus a bit more

Resource owner = Federation user

Client = Relying party

Authorization server = Identity provider ("OpenID provider")

Resource server

Token endpoint typically delivers an "ID token" similar to a SAML assertion

Standard UserInfo endpoint can be called with an access token to look up identity claims
User-Managed Access is for cross-party sharing

UMA brings next-gen delegation and consent to OAuth
User-Managed Access is for cross-party sharing
UMA brings next-gen delegation and consent to OAuth
User-Managed Access is for cross-party sharing

UMA brings next-gen delegation and consent to OAuth
User-Managed Access is for cross-party sharing

UMA brings next-gen delegation and consent to OAuth
User-Managed Access is for cross-party sharing
UMA brings next-gen delegation and consent to OAuth

At run time

UX

Opt in

Authorization on server

Resource server

Requesting party

Client

Resource server

Resource server
User-Managed Access is for cross-party sharing

UMA brings next-gen delegation and consent to OAuth

- Ahead of time
- At run time

UX

Share

Opt in

Authorisation on server

Resource server

Resource server

Resource server

Requesting party

Client

- Resource server
- Resource server
- Resource server

kantara initiative
User-Managed Access is for cross-party sharing
UMA brings next-gen delegation and consent to OAuth

- **Ahead of time**: Share
- **At run time**: Opt in
- **After the fact**: Approve

**Resource server**

**Authorisation on server**

**Client**

**Requesting party**

**UX**

**Resource owner**
User-Managed Access is for cross-party sharing

UMA brings next-gen delegation and consent to OAuth

UX
- Share
- Monitor
- Opt in
- Approve

Resource owner

Requesting party

Client

Authorisation on server

Ahead of time
- Anytime
- At run time
- After the fact

Resource server

Resource server

Resource server
User-Managed Access is for cross-party sharing
UMA brings next-gen delegation and consent to OAuth

 UX  Share  Monitor  Withdraw  Opt in  Approve

Ahead of time  Anytime  Anytime  At run time  After the fact

Authorisation

on server

Resource
server

Resource
server

Resource
server

Requesting party

Client

Resource owner
Questions?
Thank you!
Join us!

Eve Maler, WG chair

eve.maler@forgerock.com | @xmlgrrl

15 May 2018

http://tinyurl.com/umawg | @UMAWG