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 a quick UMA explainer?
News from the last year, in context

2015

Mar ‘15: UMA V1.0 ratified as Recommendations

Dec ‘15: UMA V1.0.1 ratified as Recommendations

2016

Specs refactored, over 100 issues closed, lots of implementation input received, Disposition of Comments doc written...

2017

Jul ‘17: 1st Public Comment/Review period ends

Sep ‘17: 2nd Public Comment/Review period ends

2018

Jan ‘18: Final Recommendations published

Feb ‘18: Charter update

Jan ‘18: Draft UMA Business Model Report published

May ‘18: Keycloak joins UMA2 vendors Gluu and ForgeRock
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 new UMA business model defines how the UMA protocol enables a license-based model for controlling access rights to personal digital assets.

- Maps legal party roles to technical entity roles
- Use licenses and contracts as legal devices
- Extends the “ends” to model many business relationships
On the docket for (the rest of) 2018

• Complete the business model and capture business scenarios
• Perform a business model POC
• Consider submitted UMA2 extensions
• Maintain UMA2 as required
• Promote UMA2 interoperability
OAuth, OIDC, and UMA2: breaking it down

Find links to UMA2 specs and swimlanes at http://tinyurl.com/umawg/
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

Client

Authorization server

Resource server

Authorizes (consents) at runtime 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
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
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 runtime 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
Authorization Grant
Authorization Request
Authorization Grant
Authorization Grant
Access Token
Access Token
Protected Resource
Protected Resource
Resource server
Resource owner
Client

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

Authorization Request → Authorization Grant → Authorization Grant → Authorization Grant → Access Token → Protected Resource

Resource owner

Client

Authorization server

Resource server

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

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

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
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**
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** = Identity provider ("OpenID provider")

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

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

- Resource owner
- Client
- Authorization server
- Resource server
User-Managed Access is for cross-party sharing

UMA brings next-gen delegation and consent to OAuth

Diagram:
- Resource owner
- Requesting party
- Client
- Authorization server
- Resource server
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

- **Requesting party**
- **Client**
- **Authorization server**

**UX**
- **Share**
- **Opt in**

**Ahead of time**
- **At run time**

**Resource server**

**Resource owner**

**Resource server**

**Resource server**
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

UX

Authorization server

Resource server

Resource server

Requesting party

Client
User-Managed Access is for cross-party sharing

UMA brings next-gen delegation and consent to OAuth

UX

- Share
- Monitor
- Opt in
- Approve

Authorization server

- Requesting party
- Client

Resource server

Ahead of time
Anytime
At run time
After the fact

UX

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

UMA brings next-gen delegation and consent to OAuth

Resource owner

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

UX

Share  Monitor  Withdraw  Opt in  Approve

Requesting party

Authorization server

Resource server

Authorized party

Client

Resource server

Resource server

Resource server
## Key benefits of UMA to service providers

<table>
<thead>
<tr>
<th>Benefit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True security of delegated access</td>
<td>Scalability of resource permissioning</td>
</tr>
<tr>
<td>Scalability of resource permissioning</td>
<td>API-first protection strategy</td>
</tr>
<tr>
<td>API-first protection strategy</td>
<td>Fosters control for compliance and trust</td>
</tr>
</tbody>
</table>

- **True security of delegated access**: Ensures secure access to resources.
- **Scalability of resource permissioning**: Allows for efficient management of permissions across resources.
- **API-first protection strategy**: Prioritizes protection strategies within an API framework.
- **Fosters control for compliance and trust**: Enhances trust and control over compliances regulations.
Key benefits of UMA to consumers

- Constrained party-to-party delegation
- Granting consent without external influence
- Centralized monitoring and management
- Control of consents at a fine grain
Questions?
Thank you!
Join us!

Eve Maler, WG chair

evemaler@forgerock.com | @xmlgrrl

15 May 2018

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