UMA1 Interop Participants and Solutions

UMA1 Interop Participants and Solutions

Main UMA1 Interop Testing page | Features and Feature Tests | Results

We are planning to do initial interop testing of UMA authorization servers against Roland Hedberg's test suite. Eventually we will test UMA resource servers and clients as well, and conduct cross- matrix testing. In preparation for these later phases, we began to register interop participants on this page; don't read too much into the solution information provided here for now.

Subscribe to the uma-dev mailing list for updates.

Participant information

Participant name and logo	Contact names/emails	Solution full name	Solution abbrev	Roles (AS, RS, C)
Gluu (logo, handle)	Mike Schwartz (mike-at-gluu.org) Yuriy Zabrovarnayy (yuriy-at-gluu.org)	OXAuth	OX	AS, RS, C
		Apache plugin	AP	RS, C
Cloud Identity Limited (logo, handle)	Maciej Machulak (maciej.machulak-at-cloudidentity.co.uk)	NuveAM	CI	AS, RS, C
		Python/Java UMA	PU	RS, C
Roland Hedberg	Roland Hedberg (roland.hedberg-at-adm.umu.se)	PyUMA	RH	AS, RS, C
ZXID.org (logo)	Sampo Kellomäki (sampo-uma14-at-zxid.org)	ZXID w/mod_auth_saml	ZX	AS, RS, C

Solution information: AS role

For the purposes of automating testing, one option is to use a query parameter with the name **_umaauthn** to convey a token string that enables login of an RO or RqP. Any RqP credentials provided in the form of "token strings" below can be used in this fashion.

It is assumed that the C is claims-unaware and will be using the redirect claim profile to redirect the RqP to the AS for login as the sole claims-gathering process.

The "Alice" user can be used as both an RO and an RqP, and the "Bob" user can be used as an RqP. The different RqPs can be used with the same client to test policies that discriminate between RqPs using the same client. Clients "A" and "B" can be easily used to test policies that discriminate between the same RqP using different clients.

Solution: role	Config data URL	RqP method and credentials	C credentials	Supports dynamic client registration?	Other details
OX:AS	https://seed.gluu.org/.well-known/uma-configuration	Alice: Bob:	OAuth Client A: - id: @!1111!0008!FF81!2D39 - secret: 6213e9b9-c46d-4008-8af1-03f918a8ade4 OAuth Client B: - id: @!1111!0008!0068.3E20 - secret: 32c2fb17-409d-48a2-b793-a639c8ac6cb2	RS: yes C: yes	http://www.gluu.org/docs/admin-guide/uma/ http://www.gluu.org/docs/reference/lib/uma/
CI:AS	https://demo.nuveam.com/.well-known/uma-configuration	Alice: Bob:	Client A: Client B:	RS: yes C: yes	Product Website, Online Demo
RH:AS		Alice: Bob:	Client A: Client B:	RS: yes C: yes	
ZX:AS	https://zxidp.org/.well-known/uma-configuration	At AS either use username (form field au) and password (ap) alice:test _uma_authn=YWxpY2U6dGVzdA%3D%3D # protects and accesses resource bob:test _uma_authn=Ym9iOnRlc3Q%3D # accesses resource	Client A: Client B:	RS: yes C: yes	https://zxidp.org/umainfo.html

Solution information: RS role

Any RS participating in interop needs to expose either multiple resource sets (as registered with the AS) or multiple scopes, or both. This enables testing UMA-specific interop around sufficient/insufficient authorization data, permission tickets that match or don't match the requested type of access, etc., while not dictating the specifics of what the API looks like. Each RS participant needs to provide enough information directly in the table below to explain how to access these differential resource sets and scopes, e.g. the URLs, parameters, etc. This way, clients can tell whether the RS was at fault or not if something goes wrong with authorization. It is recommended that each RS document exactly the one or two endpoints/calls/parameters that are sufficient for UMA interop testing purposes, to limit the universe of potential actions that a client can take.

Solution: role	API info	SDK avail?	Login URL and RO creds	Protected resource URL(s) info	Client SDK/library info	Expects dynamic client registration at AS?	Other details
OX:RS	http://www.gluu.org/docs/admin-guide/uma/ http://www.gluu.org/docs/reference/lib/uma/	Java	https://seed.gluu.org/oxuma-rs/	https://seed.gluu.org/oxuma-rs/ws/phone CRUD: 1. GET (view phones) : protected by "view" or "all" scope 2. DELETE (remove phone) : protected by "remove" or "all" scope 3. PUT (create/update phone) : protected by "add" or "all" scope Scopes: http://seed.gluu.org/oxuma-rs/ws/scope/add http://seed.gluu.org/oxuma-rs/ws/scope/view http://seed.gluu.org/oxuma-rs/ws/scope/all http://seed.gluu.org/oxuma-rs/ws/scope/remove			
CI:RS	https://nuvepds.appspot.com/about/api	Python and Java	https://nuvepds.appspot.com (sign in with your social profile)	https://nuvepds.appspot.com/about/api		Optional	
RH:RS	Uses "pbryan" (http-json-resource)		https://xenosmilus.umdc.umu.se:8777/login.html (user:alice, password:krall)	Base URL for alice's resources: https://xenosmilus.umdc.umu.se:8777/json/alice	Available in Python and Java (sample at https://nuvepdsclient.appspot.com/) – where?		Supports webfinger. Supports acct and http identifier urls.
ZX:RS	https://zxidp.org/umainfo.html	libzxid (C/C++, PHP, Perl, Java, Apache httpd module)	https://zxidp.org/idpuma?o=umalogin (test:test)	https://zxidp.org/idpuma?o=umaprotected	?	?	

Solution information: C role

As noted above, it is assumed that the C is claims-unaware and will be using the redirect claim profile to redirect the RqP to the AS for login as the sole claims-gathering process for assessing policy. There are currently (V0.9) not even any optional feature tests for claim profiles anyway, so we're not testing claims gathering at this stage.

Solution:role	App type	Other details
OX:C	https://seed.gluu.org/oxuma-rp/	
CI:C	https://nuvepdsclient.appspot.com/	Sign in using social profile or pass a token
RH:C		
ZX:C	https://zxidp.org/idpuma?o=umatestres	https://zxidp.org/umainfo.html