OAuth Dynamic Client Registration

IETF 90 Toronto

Since last we met

- Document reshuffling
  - “Core” and “Metadata” specs merged back into a single spec
  - “Management” spec aligned with new core
- Last call comments
  - Lots of good commentary on making the editorial text read much more clearly
  - No normative changes due to last call comments
  - A handful of changes pending final comments (e.g. jwks / jwks_uri usage)

Open Issue: IPR Attribution

- Document history
  - OIDC Dynamic Registration, UMA Dynamic Registration, and IETF Dynamic Registration all grew up together
  - Ideas came from all three groups
  - OIDC Dyn Reg is published as final
  - UMA Dyn Reg has been dropped in favor of IETF Dyn Reg
- Contention: we don’t mention any of that
- Proposed solution: Historical document attribution
  - Consensus on list to add text to introduction acknowledging OpenID Connect and UMA
  - Informational references to be added for OIDC Dyn Reg final and UMA I-D

Open Issue: “application_type”

- Imported in draft -18 from OpenID Connect Dyn Reg
  - Intended (in OIDC) to differentiate between native applications and web applications and allow AS to apply appropriate rules to each
- Contentions:
  - Was added without workgroup discussion
  - “native” and “web” applications are increasingly ill-defined
  - No registry or mechanism for defining other application types and appropriate rules
  - Value is self-asserted by client, does not add security
- Proposed solution:
  - Adopt recommendations for redirect URIs into security considerations without a formal parameter to dictate rules, tied to grant types and client aspects

A note on existing redirect URI rules

- RFC6749 already says that a redirect URI should be one of the following:
  - TLS-protected (https) if on a remote host
  - Non-TLS-protected (http) if on localhost
  - A non-http URI that is accessible only to the client

Open Issue: client_secret_expires_at

- Value indicates when the client's issued secret will no longer be valid at the AS
- Contention: without the management API, what can the client do with this information?
  - Client could re-register
  - Still makes sense for stateless or timeout-based registrations
- Proposed solution(s):
  - 1: leave it as is, a hint to the client of its state no matter whether the client can do anything about it (Justin's preferred solution)
  - 2: remove it and move it to the management spec (breaking change, as it's a required value right now)

Open Issue: Management API

- Provides a set of control verbs and functions for client lifecycle management
  - Has several implementations in production
- Contention: “maybe we can do better”
- Proposed solution:
  - Publish it as-is as an informational spec
  - … While work potentially continues on something else (if anyone’s got a concrete idea of an alternative)