4 November 2024

Client ID hint for Authorization Server Metadata Requests

Filip Skokan
Staff Engineer at Okta
https://github.com/panva

RFC8414 - Authorization Server Metadata request

```
>>> GET /.well-known/oauth-authorization-server HTTP/1.1
Host: server.example.com

<<< HTTP/1.1 200 OK
Content-Type: application/json

{
  "issuer":
    "https://server.example.com/",
  "authorization_endpoint":
    "https://server.example.com/authorize",
  "token_endpoint":
    "https://server.example.com/token",
  "token_endpoint_auth_methods_supported":
    ["client_secret_post"],
  "response_types_supported":
    ["code"],
}
```


IETF

RFC8414 - Authorization Server Metadata

Metadata Registry - https://www.iana.org/assignments/oauth-parameters/oauth-parameters.xhtml#authorization-server-metadata

Naturally, metadata presence is used for feature detection, e.g.:

> The server MUST indicate its support for ... by setting the metadata parameter ...

Additions are added in a backwards compatible matter, e.g.:

> If omitted, the default is ...

IETF

Metadata changes can “break” existing integrations

It *shouldn’t*, but it absolutely **can**.

Few examples:

An end_session_endpoint addition opts a client in to use it, customers complain because the UX changed.

code_challenge_method and token_endpoint_auth_methods_supported changed client behaviour.

(it shouldn’t have, those were bad client implementations, but it still happened)

AS Metadata does not support migrations / deprecations

How to deal with migrations to new endpoint versions (or hypothetical gradual deprecation?)

- easy for GET on the authorization_endpoint
- not so much for POST or essentially any other endpoint

Authorization Server Metadata request with a client hint

```
>>> GET /.well-known/oauth-authorization-server HTTP/1.1
Host: server.example.com

<<< HTTP/1.1 200 OK
Content-Type: application/json

{
  "issuer":
    "https://server.example.com/",
  "authorization_endpoint":
    "https://server.example.com/authorize",
  "token_endpoint":
    "https://server.example.com/token",
  "token_endpoint_auth_methods_supported":
    ["client_secret_post"],
  "response_types_supported":
    ["code"],
}
```


IETF

Authorization Server Metadata request with a client hint

```
>>> GET /.well-known/oauth-authorization-server?client_id_hint=s6BhdRkqt3 HTTP/1.1
Host: server.example.com

<<< HTTP/1.1 200 OK
Content-Type: application/json

{
  "issuer":
    "https://server.example.com/",
  "authorization_endpoint":
    "https://server.example.com/v2/authorize",
  "token_endpoint":
    "https://server.example.com/v2/token",
  "authorization_response_iss_parameter_supported": true,
  "token_endpoint_auth_methods_supported":
    ["client_secret_post"],
  "response_types_supported":
    ["code"],
}
```

Discuss

- This would not replace or deprecate RFC8414
- Goal: allow gradual migrations and feature rollouts, opting brittle clients out of unnecessary changes on a per-client basis
- Note: some servers already accommodate client / app identifiers informing the discovery responses
- Query Parameter vs HTTP Header
  - custom headers trigger browser CORS preflights and require CORS configuration
  - What about response caching - is ok or a custom response header and Vary on that to inform upstream CDN caches?

Backup

-

Why client_id_hint instead of client_id?

```
curl -s
"https://login.microsoftonline.com/common/v2.0/.well-known/openid-configuration?client_id=s6BhdRkqt3" | jq
```


```
{
  "error": "invalid_request",
  "error_description": "AADSTS1004008: Required parameter 'jwks_extensions' has not been provided. Trace ID: cecea404-ebff-4614-82e8-f582fd825c00 Correlation ID: 1a769b51-17dc-4d40-8090-0e01aabeea29 Timestamp: 2024-10-31 12:22:14Z",
  "error_codes": [
    1004008
  ],
  "timestamp": "2024-10-31 12:22:14Z",
  "trace_id": "cecea404-ebff-4614-82e8-f582fd825c00",
  "correlation_id": "1a769b51-17dc-4d40-8090-0e01aabeea29"
}
```


IETF