FAST: Scalable Registration, Authentication, and Authorization for FHIR Ecosystem Participants

March 9, 2021
FAST Security Tiger Team

• The ONC FHIR at Scale Taskforce’s Security Tiger Team was formed in late 2018 to investigate scalability issues for FHIR and identify potential solutions.

• The Tiger Team identified the Unified Data Access Profiles (UDAP) for Dynamic client registration, client authentication, client authorization, and Tiered OAuth as building blocks to be used by implementers to address the issues above and enhance the overall scalability of the FHIR ecosystem.

Project Scope

• From PSS:

“The aim of this project is to expand upon the existing work by UDAP.org within the HL7 consensus process to produce a more complete set of implementation guides targeted at implementers of both client and server systems using FHIR for data exchange, standardizing how implementers integrate the UDAP profiles identified by the FAST Security Tiger Team into existing OAuth 2.0 and OpenID Connect workflows.”
Anticipated Deliverables

• Among other things, the deliverables of this project would provide the FHIR community with detailed instructions to implement the following:
  - integration of existing public key infrastructure mechanisms with registration, authentication, and authorization processes to establish robust trust networks with reusable credentials to identify actors
  - trusted dynamic client registration
  - client app submissions of self-assertions, third party certifications, or other endorsements to servers, and vice-versa
  - client app assertions of additional information for a given session so that resource holders can more finely scope access tokens, including information related to consent or purpose of use
  - increase security and assurance in identity of all actors by using asymmetric cryptographic methods for authentication, including specific protocols to support network-wide revocation of credentials
  - dynamic federation of user credentials to facilitate reuse of credentials and single sign-on
HL7 Calendar

• Connectathon May 2021
• Ballot for STU1 September 2021
Existing UDAP IG’s

- https://www.udap.org/udap-ig-consumer-facing-health-apps.html
- https://www.udap.org/udap-ig-b2b-health-apps.html
- Google docs will be created for collaboration, commentary and review
- Consumer-Facing IG is currently referenced in CARIN Consumer Directed Payer Data Exchange (CARIN IG for Blue Button®) 1.0.0 - STU1
Meeting Frequency

• 2nd and 4th Tuesdays, 11AM-12PM Pacific/2PM-3PM Eastern
What is UDAP?

**UDAP JWT-Based Client Authentication**: Increase security by using asymmetric cryptography to authenticate client applications

**UDAP Trusted Dynamic Client Registration**: Identify and dynamically register trusted client applications, streamlining app management

**UDAP Tiered OAuth**: Reusable identities via scalable, dynamic, cross organizational user authentication

**UDAP JWT-Based Authorization Assertions**: Extensible JWT-based client authorization grants and

**UDAP Certifications & Endorsements**: Other Trusted Informational Assertions
Swimlane Overview

**Trusted Dynamic Client Registration & Token Request**

**Participant’s Client App**

UDAP Dynamic Client Registration request (signed with client’s certificate-backed key)

- Client submits:
  - Client name
  - Redirect URIs?
  - Token Endpoint Auth Method
  - Grant type client credentials

**Registration Endpoint**

- Authorization and/or Authentication JWT using `client_id` (signed with same key)
  - e.g. UDAP JWT-Based Client Authentication

- Access Token

**Authz & Token Endpoints**

- Policy Engine <rules>

Policy Engine <rules>

Art credit: adapted from ONC FAST communications collateral
Swimlane for Technical Deep Dive

Token Request – authorization code flow

Implementer’s
Client App

User’s
Browser

Authorization Endpoint

Token Endpoint

- Implements the authorization code flow

User's Browser

App sends user to authz endpoint
GET https://{authz}?response_type=code &client_id={client}&state=123&scope=s1+s2 &aud={fhirBase}&redirect_uri={redirect}

Server redirects user to app
302 Found
Location: https://{redirect}?
code={auth_code}&state=123

Client constructs authentication JWT and requests token using JWT and auth code per UDAP JWT-Based Client Authentication

Authorization Endpoint

- Policy Engine - Validate JWT & certificate chain
- Validate auth code
- Optional purpose assertion

POST https://{token}
grant_type=authorization_code&code={auth_code} &client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&client_assertion={JWT_goes_here}&udap=1

Server returns access token for use with FHIR endpoint
200 OK
{"access_token": "random_UUID_or_other_token_issued_by_AS", "token_type": "Bearer", "expires_in": 3600 }
Swimlane for Technical Deep Dive

**Token Request – client credentials flow**

**Implementer’s Client App**
- Client constructs authentication JWT and requests token using
  JWT per UDAP JWT-Based Client Authentication

**User’s Browser**
- App sends user to authz endpoint
  GET https://[authz]?response_type=code
  &client_id={client}&state=123&scope=s1+s2
  &aud={fhirBase}&redirect_uri={redirect}
- Server redirects user to app
  302 Found
  Location: https://[redirect]
  ?code={auth_code}&state=123

**Authorization Endpoint**
- Authz Engine
  - Validates audience, client ID, and redirect URI
  - Interacts with user
  - Authenticates user
  - Obtains user authorization so app can use requested scopes

**Token Endpoint**
- POST https://[token]
  grant_type=client_credentials&scope=s1+s2
  &client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&client_assertion={JWT_goes_here}&udap=1

**Policy Engine**
- Validate JWT & certificate chain
- Validate scopes
- Optional purpose assertion

**Server returns access token for use with FHIR endpoint**
- 200 OK
- {"access_token": "random_UUID_or_other_token_issued_by_AS",
  "token_type": "Bearer", "expires_in": 3600}
UDAP Tiered OAuth with optional Trusted Dynamic Client Registration

Requestor Actor

Client App

User’s Browser

Responder Actor

Authorization Endpoint

App sends User to authorization endpoint

Responder uses OpenID Connect to authenticate User

Authentication Response

Server redirects User Back to App

App receives code

Responder Actor

OIDC Endpoints

Requestor Actor

OIDC Endpoints

Authenticate/Authorize (CC2)

Role/Context Identification (CC9)
UDAP Tiered OAuth with optional Trusted Dynamic

Requestor Actor  
Client App

User’s Browser

App sends User to authorization endpoint

"idp": "https://myidp.com/"

1

Responder Actor
Authorization Endpoint

Responder gets metadata and validates trust

Responder dynamically registers with OIDC IdP if not previously registered;

Registration Response (including client_id)

2

Requestor Actor
OIDC Endpoints

3

Authenticate/Authorize (CC2)

Role/Context Identification (CC9)

User interacts with IdP to complete Authentication

4

Responder redirects user agent to IdP’s Authorization Endpoint to begin OIDC Auth

5

Authenticate/Authorize (CC2)

Role/Context Identification (CC9)

If successful, App receives code that can be exchanged for ID Token

6

Responder redirects user agent if additional interactions with user are required

User completes additional interaction

7

Server redirects User Back to App (success or failure)

If successful, App receives code that can be exchanged for ID Token