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

April 13, 2021
Welcome

• New participants
• Tom Loomis – Evernorth/Cigna
• Baird Kaake – Evernorth/Cigna
• Sri Tulasiram – Humana
• Patrick Haren – Cigna/Evernorth
• Rose-Marie Nsahlai – ONC
• Joseph Lamy – eHealth Exchange
• Isaac Vetter – Epic
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.”
Timeline Progress

• ISD comments addressed, PSS updated; e-vote: approved
• TSC (deadline: 4/25/21)
• Connectathon May 2021 – track proposal submitted
  • Training course on 4/26/21 for new first-time participants
  • Sign up open now!
• FHIR IG proposals (like PSS+FHIR details) due by June 20
  • Dana is drafting – FMG is the only HL7 group to review this
• NIB final deadline July 4 – plan to submit by end of April
• Ballot for STU1 September 2021
FHIR Connectathon 27 – May 2021

• Proposal has been added to site:

  https://confluence.hl7.org/display/FHIR/2021-05+Cross+Organization+Application+Access

• Continuation of predecessor track used for testing UDAP workflows with FHIR

• TODO: Refine specific scenarios to test this event
  • Most of the workflows have been tested at previous connectathons
  • Focus on specific workflows in the draft IGs
Existing UDAP IG’s

- [https://www.udap.org/udap-ig-consumer-facing-health-apps.html](https://www.udap.org/udap-ig-consumer-facing-health-apps.html)
- [https://www.udap.org/udap-ig-b2b-health-apps.html](https://www.udap.org/udap-ig-b2b-health-apps.html)

- TODO: port to FHIR IG template
  - [http://build.fhir.org/ig/FHIR/ig-guidance/index.html](http://build.fhir.org/ig/FHIR/ig-guidance/index.html)
  - [https://github.com/HL7/ig-template-fhir](https://github.com/HL7/ig-template-fhir)
  - Markdown conversion
  - Separate into tabs?
  - Two tabs or Two IGs?
    - Substantial overlap
    - Very different use cases
Comparison with Carequality UDAP framework

- Carequality FHIR IG (Version 1.0, December 1, 2020)
  - Aligns with Draft IGs
  - Algs: RS256 (SHALL), ES256 (SHOULD), ES384 + RS384 (MAY)
    - Note: draft IGs do not include RS384
  - Community Certifications: Basic App Certification (self-assertion)
  - Community Authorization Extension Objects: carequality, carequality_user
  - Community Authorization Extension Error Objects: carequality
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.

Trademark Notice: UDAP and the ecosystem gears design are trademarks of UDAP.org and are used with the permission of UDAP.org.

© 2019 Health Level Seven ® International. Licensed under Creative Commons Attribution 4.0 International HL7, Health Level Seven, FHIR and the FHIR flame logo are registered trademarks of Health Level Seven International. Reg. U.S. TM Office.
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

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

**Registration Endpoint**

- Policy Engine <rules>

  - Registration Response

**Authz & Token Endpoints**

- Access Token

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

Token Request – authorization code flow

**Implementer’s Client App**

- **Token Request**
  - User constructs authentication JWT and requests token using JWT and auth code per UDAP JWT-Based Client Authentication

**User’s Browser**

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

**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

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

**Token Endpoint**

- GET https://{redirect}?code={auth_code}&state=123

- 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}
Token Request – client credentials flow

Implementer’s Client App

User’s Browser

Authorization Endpoint

Token Endpoint

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 }

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

Server redirects user to app

302 Found
Location: https://{redirect}?
?code={auth_code}&state=123

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}

Policy Engine

- Validate JWT & certificate chain
- Validate scopes
- Optional purpose assertion

Authz Engine

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

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

GET https://{redirect}?
?code={auth_code}&state=123

User's Browser

Authz Engine

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

Server redirects user to app

302 Found
Location: https://{redirect}?
?code={auth_code}&state=123
UDAP Tiered OAuth with optional Trusted Dynamic Client Registration

Requestor Actor (Client App)

User’s Browser

App sends User to authorization endpoint

Responder Actor

Authorization Endpoint

Responder uses OpenID Connect to authenticate User

Responder Actor

OIDC Endpoints

Authentication Response

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/”

Responder Actor
Authorization Endpoint

Responder gets metadata and validates trust

Registration dynamically registers with OIDC IdP
if not previously registered;

Registration Response (including client_id)

Responder Actor
OIDC Endpoints

authenticate/authorize (CC2)

Authenticate/Authorize (CC2)

Role/Context Identification (CC9)

Authenticate/Authorize (CC2)

Role/Context Identification (CC9)

Requestor Actor

User completes additional interactions with user are required

User interacts with IdP to complete Authentication

IdP provides ID token to responder

Server redirects User Back to App (success or failure)

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

Responder dynamically registers with OIDC IdP

Responder gets metadata and validates trust

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

User interacts with IdP to complete Authentication

IdP provides ID token to responder

User completes additional interaction

Server redirects User Back to App (success or failure)

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