UDAP Unified Data Access Profiles

Implementation Guide for Registration and Authorization of Consumer Facing Health Apps

DRAFT 2020-08-11

This implementation guide describes how to extend OAuth 2.0 and the HL7 SMART App Launch Framework using UDAP workflows for consumer-facing apps that implement the authorization code flow. This guide covers automating the client application registration process and increasing security using asymmetric cryptography to authenticate ecosystem participants. This guide also provides a grammar for communicating metadata critical to healthcare information exchange.

The requirements described in this guide are intended to align with the proposed solutions of the ONC FHIR at Scale Task Force's Security Tiger Team, the security model outlined in the draft Carequality FHIR IG, and implementation guide drafts incorporating UDAP workflows under development by CARIN and the Da Vinci Project.

Note: The proposed URL for publication of this draft is:
www.udap.org/udap-ig-consumer-facing-health-apps.html

1 JSON Web Token requirements

JSON Web Tokens (JWTs) shall conform to the mandatory requirements of RFC 7519. All JWTs defined in this guide are JSON Web Signatures and shall additionally conform to the mandatory requirements of RFC 7515. All JWTs SHALL be serialized using JWS Compact Serialization as per Section 7.2 of RFC 7515.

Signature algorithm identifiers used in this guide are defined in Section 3.1 of RFC 7518. Implementations supporting the UDAP workflows defined in this guide SHALL support RS256. In addition to the algorithm required by the referenced UDAP specifications, this guide also permits the use of ES256 and ES384. Implementations SHOULD support ES256, and MAY support ES384.

All JWTs defined in this guide SHALL contain a Javascript Object Signing and Encryption (JOSE) header conforming to the following requirements:

JWT Header Values

alg	required	A string containing the JWA algorithm used for signing the JWT. For example: "RS256"
x5c	required	An array of one or more strings containing the X.509 certificate or certificate chain [RFC5280], with the leaf certificate corresponding to the key used to digitally sign the JWT. Each string in the array is a base64-encoded DER representation of the certificate, with the leaf certificate appearing as the first element (or only) of the array. See https://tools.ietf.org/html/rfc7515#section-4.1.6

2 Discovery

2.1 Discovery of Endpoints

A FHIR Server SHALL make its Authorization Server's authorization, token, and registration endpoints, and associated metadata, available for discovery by client applications. Servers SHALL allow access to the following metadata URLs to unregistered client applications and without requiring client authentication, where [baseURL] represents the base FHIR URL for the FHIR server:

1. A CapabilityStatement at [baseURL]/metadata. The CapabilityStatement SHALL include:
    a. the OAuth 2.0 URIs Extension on the rest.security element of the as per Section 3.1 of the HL7 SMART App Launch Framework, and list the "authorize", "token", and "register" components.
    b. the following codes in the rest.security.service list: http://hl7.org/fhir/restful-security-service|SMART-on-FHIR and (if UDAP workflows are supported) http://fhir.udap.org/CodeSystem/capability-rest-security-service|UDAP.
2. SMART configuration metadata at [baseURL]/.well-known/smart-configuration. The metadata SHALL be structured as per Section 4 of the HL7 SMART App Launch Framework, and include the respective endpoint URLs for the "authorization_endpoint", "token_endpoint", and "registration_endpoint" keys, and "private_key_jwt" in the array of endpoint authentication methods for the "token_endpoint_auth_methods" key.
3. UDAP metadata as defined below at [baseURL]/.well-known/udap, structured as a JSON object as per section 1 of UDAP Dynamic Client Registration and discussed further in Section 2.2.

2.2 Required UDAP Metadata

The metadata returned from the UDAP metadata endpoint defined above SHALL represent the server's capabilities with respect to the UDAP workflows described in this guide. If no UDAP workflows are supported, the server SHALL return a 404 Not Found response to the metadata request. For elements that are represented by JSON arrays, clients SHALL interpret an empty array value to mean that the corresponding capability is NOT supported by the server.

udap_versions_supported	required	A fixed array with one string element: ["1"]
udap_certifications_supported	recommended	An array of zero or more certification URIs supported by the Authorization Server, e.g.: ["https://www.example.com/udap/profiles/example-certification"]
udap_certifications_required	recommended	An array of zero or more certification URIs required by the Authorization Server, e.g.: ["https://www.example.com/udap/profiles/example-certification"]
grant_types_supported	recommended	An array of one or more grant types supported by the Authorization Server, e.g.: ["authorization_code", "refresh_token"]
scopes_supported	recommended	An array of one or more strings containing scopes supported by the Authorization Server, as defined at http://hl7.org/fhir/smart-app-launch/scopes-and-launch-context/index.html. The server MAY support different

		subsets of these scopes for different client types or entities. E.g.: ["openid", "launch/patient", "system/Patient.read", "system/AllergyIntolerance.read", "system/Procedures.read"]
authorization_endpoint	recommended	A string containing the URL of the Authorization Server's authorization endpoint.
token_endpoint	recommended	A string containing the URL of the Authorization Server's token endpoint if the server supports UDAP JWT-Based Client Authentication.
token_endpoint_auth_methods_supported	recommended	Fixed array with one value: ["private_key_jwt"]
token_endpoint_auth_signing_alg_values_supported	recommended	Array of strings listing one or more JWA algorithm identifiers supported by the Authorization Server for validation of signed JWTs submitted to the token endpoint for client authentication. For example: ["RS256", "ES384"]
registration_endpoint	recommended	A string containing the URL of the Authorization Server's registration endpoint if the server supports UDAP Dynamic Client Registration.

registration_endpoint_jwt_signing_alg_values_supported	recommended	Array of strings listing one or more JWA algorithm identifiers supported by the Authorization Server for validation of signed software statements, certification, and endorsements submitted to the registration endpoint. For example: ["RS256", "ES384"]

3 Client Application Registration

Before FHIR data requests can be made, Client applications operators SHALL register each of their applications with the Authorization Servers identified by the FHIR servers with which they wish to exchange data. Client applications SHALL use the client_id assigned by an Authorization Server in subsequent authorization and token requests to that server.

Authorization Servers SHOULD support dynamic registration as specified in the UDAP Dynamic Client Registration profile DRAFT 2019-05-15 at http://www.udap.org/udap-dynamic-client-registration.html with the additional options and constraints defined in this guide. Confidential clients, i.e. conventional server-based web applications that can maintain a secret, MAY use this dynamic client registration protocol as discussed further in Sections 3.1 through 3.3 to obtain a client_id. Other client types SHOULD follow the manual registration processes for each Authorization Server. Future versions of this guide may add support for dynamic client registration by native device applications or public clients which cannot protect a private key.

3.1 Software Statement

To register dynamically, the client application shall first construct a software statement as per section 2 of UDAP Dynamic Client Registration.

The software statement SHALL contain the required header elements specified in Section 1 of this guide and the JWT claims listed in the table below. The software statement SHALL be signed by the client application operator using the signature algorithm identified in the ‘alg’ header of the software statement and with the private key that corresponds to the public key listed in the client’s X.509 certificate identified in the ‘x5c’ header of the software statement.

Software Statement JWT Claims		
iss	required	Issuer of the JWT -- unique identifying client URI. This SHALL match the value of a uniformResourceIdentifier entry in the Subject Alternative Name extension of the client's certificate included in the 'x5c' JWT header
sub	required	Same as 'iss'. In typical use, the client application will not yet have a client_id from the Authorization Server
aud	required	The Authorization Server's "registration URL" (the same URL to which the registration request will be posted)
exp	required	Expiration time integer for this software statement, expressed in seconds since the "Epoch" (1970-01-01T00:00:00Z UTC). The 'exp' time SHALL be no more than 5 minutes after the value of the 'iat' claim.
iat	required	Issued time integer for this software statement, expressed in seconds since the "Epoch"
jti	required	A nonce string value that uniquely identifies this software statement. This value SHALL NOT be reused by the client app in another software statement or authentication JWT before the time specified in the "exp" claim has passed
client_name	required	A string containing the human readable name of the client application

redirect_uris	conditional	An array of one or more redirection URIs used by the client application. This claim SHALL be present if grant_types includes "authorization_code" and this claim SHALL be absent otherwise. Each URI SHALL use the https scheme.
contacts	required	An array of URI strings indicating how the data holder can contact the app operator regarding the application. The array SHALL contain at least one valid email address using the mailto scheme, e.g. ["mailto:operations@example.com"]
logo_uri	conditional	A URL string referencing an image associated with the client application, i.e. a logo. If grant_types includes "authorization_code", client applications SHALL include this field, and the authorization server MAY display this logo to the user during the authorization process. The URL SHALL use the https scheme and reference a PNG, JPG, or GIF image file, e.g. "https://myapp.example.com/MyApp.png"
grant_types	required	Array of strings, each representing a requested grant type, from the following list: "authorization_code", "refresh_token". The value "refresh_token" SHALL NOT be present in the array unless "authorization_code" is also present
response_types	conditional	Array of strings. If grant_types contains "authorization_code", then this element SHALL have a fixed value of ["code"], and SHALL be omitted otherwise
token_endpoint_auth_method	required	Fixed string value: "private_key_jwt"

scope	required	String containing a space delimited list of scopes requested by the client application for use in subsequent requests. The Authorization Server MAY consider this list when deciding the scopes that it will allow the application to subsequently request

The unique client URI used for the 'iss' claim SHALL match the uriName entry in the Subject Alternative Name extension of the client app operator's X.509 certificate, and SHALL uniquely identify a single client app operator and application over time. The software statement is intended for one-time use with a single OAuth 2.0 server. As such, the 'aud' claim SHALL list the URL of the OAuth Server's registration endpoint, and the lifetime of the software statement ('exp' minus 'iat') SHALL be 5 minutes.

3.2 Example

Example software statement, prior to Base64URL encoding and signature (non-normative, the "." between the header and claims objects is a convenience notation only):


```
{
 "alg": "RS256",
 "x5c": ["MIEF.....remainder omitted for brevity"]
}.{
 "iss": "http://example.com/my-app",
 "sub": "http://example.com/my-app",
 "aud": "https://oauth.example.net/register",
 "exp": 1597186041,
 "iat": 1597186341,
 "jti": "random-value-109a3bd72"
 "client_name": "My Consumer-Facing App",
 "redirect_uris": ["https://myapp.example.com/redirect"],
 "contacts": ["mailto:operations@example.com"],
 "logo_uri": "https://myapp.example.com/MyApp.png",
 "grant_types": ["authorization_code"],
 "response_types": ["code"],
 "token_endpoint_auth_method": "private_key_jwt",
 "scope": ["user/Patient.read", "user/Procedure.read"]
}
```


Request Body

The registration request is submitted by the client to the Authorization Server's registration endpoint.

```
POST /register HTTP/1.1
Host: oauth.example.net
Content-Type: application/json

{
  "software_statement": "...the signed software statement JWT...",
  "certifications": ["...a signed certification JWT…"]
  "udap": "1"
}
```

The Authorization Server SHALL validate the registration request as per Section 4 of UDAP Dynamic Client Registration. This includes validation of the JWT payload and signature, validation of the X.509 certificate chain, and validation of the requested application registration parameters. If a new registration is successful, the Authorization Server SHALL return a registration response with a HTTP 201 response code as per Section 5.1 of UDAP Dynamic Client Registration, including the unique client_id assigned by the Authorization Server to that client app. If a new registration is not successful, e.g. it is rejected by the server for any reason, the Authorization Server SHALL return an error response as per 5.2 of UDAP Dynamic Client Registration.

3.3 Inclusion of Certifications And Endorsements

Authorization Servers MAY support the inclusion of certifications and endorsements by client application operators using the certifications framework outlined in UDAP Certifications and Endorsements for Client Applications. Authorization Servers SHALL ignore unsupported or unrecognized certifications.

Authorization Servers MAY require registration requests to include one or more certifications. If an Authorization Server requires the inclusion of a certain certification, then the Authorization Server SHALL communicate this by including the corresponding certification URI in the udap_certifications_required element of its UDAP metadata.

3.4 Modifying and Cancelling Registrations

The client URI in the Subject Alternative Name of an X.509 certificate uniquely identifies a single application and its operator over time. Thus, a previously registered client application MAY request a modification of its previous registration with an Authorization Server by submitting another registration request to the same Authorization Server's registration endpoint using a certificate with a Subject Alternative Name client URI entry matching the original registration request.

If an Authorization Server receives a valid registration request with a software statement containing the same 'iss' value as an earlier software statement but with a different set of claims or claim values, or with a different (possibly empty) set of optional certifications and endorsements, the server SHALL treat this as a request to modify the registration parameters for the client application by replacing the information from the previous registration request with the information included in the new request. For

example, an Application operator may use this mechanism to update a redirection URI or to remove or update a certification. If the registration modification request is accepted, the Authorization Server SHOULD return the same client_id in the registration response as for the previous registration. If it returns a different client_id, it SHALL cancel the registration for the previous client_id.

If an Authorization Server receives a valid registration request with a software statement that contains an empty grant_types array from a previously registered application, the server SHOULD interpret this as a request to cancel the previous registration. A client application SHALL interpret a registration response that contains an empty grant_types array as a confirmation that the registration for the client_id listed in the response has been cancelled by the Authorization Server.

If the Authorization Server returns the same client_id in the registration response for a modification request, it SHOULD also return an HTTP 200 response code. If the Authorization Server returns a new client_id in the registration response, the client application SHALL use only the new client_id in subsequent transactions with the Authorization Server.

4 Authorization process

Client applications SHALL obtain an access token for access to FHIR resources by following the OAuth 2.0 authorization code grant flow, as extended by the SMART App Launch Framework, and with the additional options and constraints discussed in this section.

4.1 Obtaining an authorization code

Client applications SHALL request an authorization code as per section 7.1.1 of the HL7 SMART App Launch Framework, with the following additional constraints. Client applications are NOT REQUIRED to include a launch scope or launch context requirement scope. Client applications and servers MAY optionally support UDAP Tiered OAuth for User Authentication to allow for cross-organizational or third party user authentication.

Servers SHALL handle and respond to authorization code requests as per section 7.1.2 of the HL7 SMART App Launch Framework.

4.2 Obtaining an access token

Client applications SHALL exchange authorization codes for access tokens as per section 7.1.3 of the HL7 SMART App Launch Framework, with the following additional options and constraints.

4.2.1 Constructing Authentication Token

If the client app has registered to authenticate using a private key rather than a shared client_secret, then the client SHALL use its private key to sign an Authentication Token as described in this section, and

include this JWT in the client_assertion parameter of its token request as described in section 5.1 of UDAP JWT-Based Client Authentication and detailed further in Section 4.2.2 of this guide.

Authentication Tokens submitted by client apps SHALL conform to the general JWT header requirements above and SHALL include the following parameters in the JWT claims defined in Section 4 of UDAP JWT-Based Client Authentication:

Authentication JWT Claims		
iss	required	The unique identifying URI client for this client application and client app operator. This URI SHALL match the value of a uniformResourceIdentifier entry in the Subject Alternative Name extension of the client's certificate included in the 'x5c' JWT header.
sub	required	The application's client_id as assigned by the authorization server during the registration process
aud	required	The FHIR authorization server's token endpoint URL
exp	required	Expiration time integer for this authentication JWT, expressed in seconds since the "Epoch" (1970-01-01T00:00:00Z UTC).
iat	required	Issued time integer for this authentication JWT, expressed in seconds since the "Epoch"
jti	required	A nonce string value that uniquely identifies this authentication JWT. This value SHALL NOT be reused by the client app in another authentication JWT before the time specified in the "exp" claim has passed

The maximum lifetime for an Authentication Token SHALL be 5 minutes, i.e. the value of 'exp' minus the value of 'iat' SHALL NOT exceed 300 seconds. The Authorization Server MAY ignore any unrecognized claims in the Authentication Token. The Authentication Token SHALL be signed and serialized using the JSON compact serialization method.

4.3 Submitting a token request

For client applications authenticating with a shared secret, the client application and server SHALL follow the token request and response protocol in Section 7.1.3 of the HL7 SMART App Launch Framework.

Client applications authenticating with a private key and Authentication Token as per Section 4.2 SHALL submit a POST request to the Authorization Server's token endpoint containing the following parameters as per Section 5.1 of UDAP JWT-Based Client Authentication. Client apps authenticating in this manner SHALL NOT include an HTTP Authorization header or client secret in its token endpoint request. The token request SHALL include the following parameters:

grant_type	required	Fixed value: authorization_code
code	required	The code that the app received from the authorization server
redirect_uri	required	The client application's redirection URI matching the redirect_uri value included in the initial authorization endpoint request
client_assertion_type	required	Fixed value: urn:ietf:params:oauth:client-assertion-type:jwt-bearer
client_assertion	required	The signed Authentication Token JWT
udap	required	Fixed value: 1

Authorization servers receiving token requests containing Authentication Tokens as above shall validate and respond to the request as per Sections 6 and 7 of UDAP JWT-Based Client Authentication.

For all successful token requests, the Authorization Server SHALL issue access tokens with a lifetime no longer than 60 minutes.

5 References

Boston Children's Hospital and Health Level Seven International, "Smart App Launch Implementation Guide", Version 1.0.0, Health Level Seven, November 13, 2018.

Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, RFC Editor, October 2012.

Jones, M., et al., "JSON Web Signature (JWS)", RFC 7515, RFC Editor, May 2015.

Jones, M., et al, "JSON Web Token (JWT)", RFC 7519, RFC Editor, May 2015.

Maas, L., "UDAP Dynamic Client Registration", Draft Specification, UDAP.org, May 15, 2019.

Maas, L., and Maas, J., "UDAP JWT-Based Client Authentication", Draft Specification, UDAP.org, August 14, 2018.

Maas, L., and Maas, J., “UDAP Certifications and Endorsements for Client Applications”, Draft Specification, UDAP.org, May 15, 2019.
Maas, L., and Maas, J., “UDAP Tiered OAuth for User Authentication”, Draft Specification, UDAP.org, July 23, 2018.

6 Authors

Luis C. Maas III, EMR Direct

7 Notices