# Argonaut App-Authorization Profile Alignment with HEART Profiles

December 30, 2015

## Introduction

At the December meeting of the Argonaut Steering Committee, a question was raised regarding the alignment of the Argonaut authorization profiles with the profiles being developed by OpenID's Health Relationship Trust (HEART) Working Group. Specifically, the questioner wanted to make sure that the Argonaut profiles are consistent with the HEART "low-level" profile so that if Argonaut wanted to add use-managed access in the future, it would be fairly easy to do so. The purpose of this paper is to respond to this query.

The HEART Working Group states as its purpose:

> "to harmonize and develop a set of privacy and security specifications that enable an individual to control the authorization of access to RESTful health-related data sharing APIs, and to facilitate the development of interoperable implementations of these specifications by others." (HEART Working Group Charter)

The HEART Working Group has developed the following three DRAFT specifications:

1. Health Relationship Trust Profile for OAuth 2.0
2. Health Relationship Trust Profile for OpenID Connect 1.0
3. Health Relationship Trust Profile for User Managed Access 1.0

OpenID Connect and User Managed Access (UMA) are built on OAuth 2.0, which HEART sometimes referred to as the "low-level profile." As such, the OpenID and UMA profiles inherit all requirements in the OAuth 2.0 profile.

In addition, the HEART Working Group has produced a DRAFT specification specifying FHIR-based scopes for HEART OAuth requests entitled Health Relationship Trust Profile for Fast Health Interoperability Resources (FHIR) OAuth 2.0 Scopes.

The Argonaut SMART App Authorization Guide specifies OAuth 2.0 application programming interfaces that enable applications to retrieve FHIR resources, and to request end-user OpenID authentication. Thus it includes aspects of both the HEART OAuth 2.0 and OpenID Connect 1.0 profiles, but does not explicitly include the use case envisioned for UMA. Most importantly, the Argonaut app-authorization profile focuses exclusively on authorization for, and retrieval of, FHIR resources held by Electronic Health Record (EHR) technology. The HEART profiles are not healthcare-specific at all, although the HEART Project currently is working on profiles for accessing FHIR resources.

The Argonaut SMART App Authorization Guide includes the option for an EHR authorization server to ask the individual resource owner (presumably end-user) to approve or disapprove authorization for an app to access the individual's data (a core OAuth 2.0 capability). However;

the use case does not preclude the possibility that the authorization server might access a database of accesses that the individual has pre-approved (i.e., UMA). The access rules and workflow the authorization server uses to mediate an access request is internal to an organization and therefore outside the scope of the Argonaut SMART App Authorization Guide.

## Purpose

The purpose of this analysis is to identify the alignment between the Argonaut SMART App Authorization Guide and the HEART OAuth 2.0 and OpenID Connect profiles. The aim is not to declare one "superior" over the other, but to determine whether a reasonable path exists for the Argonaut profile to be extended to include the HEART functionality.

## Analysis

The table below compares HEART OAuth 2.0, with OpenID Connect, with the Argonaut SMART App Authorization Guide. This analysis is based upon existing HEART and Argonaut documentation, and incorporates insights from principals of both the HEART and Argonaut Projects.

| Topic | HEART | Argonaut App Authorization |
|---|---|---|
| 1. Client types | "Full" (confidential) clients, browser-embedded, direct-access clients | Confidential and public clients seeking to access FHIR resources held by an EHR, and launched from within or outside an EHR |
| Analysis: No basic difference in client types, although Argonaut categorizes clients based on their capability to authenticate themselves, and HEART categorizes clients based on their architecture. However, a fundamental difference between the HEART profiles and the Argonaut profile is that Argonaut focuses on queries for FHIR resources held by an EHR. As such, the Argonaut profile includes a set of additional parameters designed to support embedding apps within EHRs, and launching apps from patient portals – both of which are aimed at communicating the context of the user session from which the app was launched. These launch-context details are outside the scope of the HEART profiles. | | |
| 2. Authorization grant types | Authorization code flow for "full" clients, implicit flow for browser-embedded, and client-credential flow for direct-access | Authorization code flow for both confidential and public clients (authorization-code flow is strongest of the grant types defined in OAuth 2.0) |
| Analysis: This is a fundamental difference. The authorization code flow is the strongest of grant types defined in OAuth 2.0. We would encourage HEART to consider using the | | |

| Topic | HEART | Argonaut App Authorization |
|---|---|---|
| authorization code grants for all client types as a means of addressing certain types of errors that developers are prone to make. | | |
| 3. Refresh tokens | Refresh tokens available only for "full" clients | Online_access and offline_access refresh tokens supported for both confidential and public clients |
| Analysis: The HEART profiles issue refresh tokens only to "full" (confidential) clients. Refresh tokens are useful in enforcing the "minimum necessary" rule in that they enable authorization servers to issue access tokens with short expiration times, with the option to renew them using longer-lived refresh tokens. Also, the Argonaut profile anticipates the need for some apps to be able to access EHR data even when the user no longer is using the app. So we provide for two types of refresh tokens: "online-access," which are valid only while the user is actively using the app, and "offline-access," which enable the client to access data after the user is no longer active on the application (for example, to periodically "ping" an EHR for a lab result). | | |
| 4. Client registration | Dynamic registration recommended. Must include public keys. | Client registration pre-condition (static or dynamic acceptable). |
| Analysis: The only fundamental difference here is that the HEART profiles require that apps register a public key for use in digitally signing JWT authentication tokens. (see #5 below) | | |
| 5. Client authentication | Full and direct clients must authenticate using JWT client authentication, with private key | Confidential clients must authenticate themselves using HTTP Basic (shared secret) |
| Analysis: The HEART profiles require the use of digitally signed JWT tokens for client authentication. The Argonaut profile uses client secrets for authentication. The use cases for the Argonaut app-authorization profile all have the end-user in the loop, making the strength of client-authentication less critical than in the cross-organizational use case, the profile for which we do use JWT authentication. Clearly, this is an area of potential future alignment with the HEART Project. | | |
| 6. Requests to auth end-point | State parameter with at least 128 bits of entropy | State parameter with at least 128 bits of entropy |
| Analysis: No difference. | | |

| Topic | HEART | Argonaut App Authorization |
|---|---|---|
| 7. Redirect URI | Clients using authorization code or implicit grant types must register full redirect URIs | All clients must register full redirect URIs |
| Analysis: Same as #2 above. | | |
| 8. Dynamic registration | Authorization servers must support dynamic client registration, and clients using authorization code or implicit grant types MAY register using dynamic registration | Clients must register with authorization server; method is unspecified |
| Analysis: Argonaut profile leaves the client-registration method up to the organization. | | |
| 9. OAuth 2.0 server profile | All servers must conform to applicable recommendations in Security Considerations sections of RFC6749 (Framework) and recommendations in RFC6819 (Threat Model) | Security Considerations of RFC6749, and recommendations in RFC5819 were primary source documents for security decisions and best practices, but are not explicitly called out as "requirements" |
| Analysis: HEART and Argonaut use the same source for security requirements and considerations. | | |
| 10. OpenID Connect service Discovery | Authorization server must provide OpenID Connect service discovery, which requires the authorization server to publish authorization, token, introspection, and revocation endpoint URLs | Authorization server must publish authorize and token endpoint URLs in FHIR conformance statement; API enables client to request OpenID token and profile by including in scope |
| Analysis: Argonaut uses FHIR resources for both conformance statement (where end-points are defined) and for OpenID user profiles. | | |

| Topic | HEART | Argonaut App Authorization |
|---|---|---|
| 11. Access tokens | Must be digitally signed JWT bearer tokens | Must be bearer tokens whose format is determined by the issuing organization |
| Analysis: An access token is effectively an "internal" communication between the authorization server that issues the token and the resource server to whom the token is presented. As such, the Argonaut Project believes that the choice of format for an access token should be a decision left up to the organization holding the resource. Note that the Argonaut profile does not preclude the use of JWT bearer tokens as access tokens. | | |
| 12. Token lifetimes | Recommends lifetimes for different types of tokens issued to different types of clients:<br>• Clients using authorization code grant type: no more than 1 hour for access tokens, and no more than 24 hours for refresh tokens<br>• Clients using implicit-grant type: no more than 15 minutes<br>• Clients using client credentials: no more than 6 hours | (Only authorization code grant type is supported) Recommends no more than 1-hour lifetime for access tokens and no more than 24 hours for refresh tokens |
| Analysis: The recommended lifetime for access tokens for clients using the authorization code grant type is the same for HEART and Argonaut. | | |
| 13. Token revocation and introspection | Authorization server must support token revocation (RFC7009) and token introspection (RFC7662) | Token revocation and introspection are not addressed |
| Analysis: Token revocation and introspection are internal processes that may be useful to individual organizations, but are not strictly required for interoperability. We would note that the Argonaut approach of issuing longer-lived refresh tokens along with short-expiration access tokens provides authorization servers the opportunity to effectively "revoke" an access token whenever a refresh token is presented to exchange for a new access token. We would welcome feedback from Argonaut implementers regarding their view of these requirements. | | |

| Topic | HEART | Argonaut App Authorization |
|---|---|---|
| 14. Resource requests | Resource servers must support bearer tokens passed in Authentication header and may support form-parameter or query-parameter methods | Resource servers must support bearer tokens passed in Authentication header; use cases address queries for FHIR resources |
| Analysis: The Argonaut profile supports queries for FHIR resources. As such, the profile includes an approach to scoped access control wherein scopes are tied to FHIR resource types (e.g., "patient/Condition.read." HEART's profiles do not impose any limitations on the scopes with which they can be used. A key difference between the two profile is the use of the "aud" (audience) parameter. Although both profiles require that the "aud" parameter be included in the authorization request, for HEART, the "aud" value is the URL of the authorization server's token endpoint, whereas in the Argonaut profile, the "aud" value identifies the | | |
| 15. Scopes | Broad statement regarding scopes for protected resources | Scopes include requests for FHIR resources |
| Analysis: This is simply a difference in focus – Argonaut focuses on queries for FHIR resources. | | |
| 16. Advanced security options | Include client TLS authentication and proof-of-possession tokens as options | Advanced options are not included |
| Analysis: These "options" are not included in the Argonaut profile. | | |
| 17. Security considerations | Transactions over TLS, and security considerations of RFC6749 and RFC6819 | Transactions over TLS, and security considerations of RFC6749 and RFC6819 |
| Analysis: Both profiles use RFC6749 and RFC6819 as their primary sources of security requirements and recommendations. However, a key difference is the use of the "aud" (audience) parameter in the authorization request. The HEART profile does not require the "aud" parameter, but mentions that an authorization server "MAY" use this parameter to specify an array of identifier(s) of protected resources for which a token is valid. The Argonaut profile REQUIRES the use of the "aud" parameter to identify the resource server for which the access token is valid (i.e., the resource server that holds the FHIR resource). The use of the "aud" parameter helps prevent leakage of a valid bearer token to a counterfeit resource server, and is consistent with guidance provided by RFC6819. Note that when the client is launched from an EHR, the EHR passes an "iss" parameter containing the value the | | |

| Topic | HEART | Argonaut App Authorization |
|---|---|---|
| client then passes to the authorization server in the "aud" parameter. Although the OpenID Connect 1.0 Core specification defines the "iss" (issuer) parameter, it is not used in the HEART specifications. | | |
| 18. ID tokens | ID token is digitally signed | ID token is digitally signed |
| Analysis: No difference. | | |
| 19. UserInfo endpoint | Servers must support UserInfo endpoint | User profile is retrieved as FHIR resource |
| Analysis: HEART's authorization server (and OpenID provider) returns end-user profile. Argonaut's clients retrieve user profile as FHIR resource from the resource server. | | |
| 20. Request objects | Clients may send request objects to authorization endpoint | Only requests for FHIR resources are supported |
| Analysis: This is simply a difference in focus – Argonaut focuses on queries for FHIR resources. | | |
| 21. Authentication context | ID tokens must be JWT web tokens and must include acr (authentication context class reference) and amr (authentication methods reference) parameters | ID tokens are as specified in OpenID Connect Core 1.0 specification specification -- in which acr and amr parameters are optional |
| Analysis: Both use same standard – OpenID Connect Core 1.0. | | |

## Conclusion

Both the HEART OAuth 2.0 and OpenID Connect profiles, and the Argonaut SMART App-Authorization Guide specify profiles for using RFC6749, The OAuth 2.0 Authorization Framework, to enable applications to request authorization to access resources, with end-user input into the authorization decision. The principle difference is that the Argonaut profile focuses on authorizing apps to access FHIR resources held by EHR technology, whereas the HEART profiles are more general. Whereas differences exist, none of these differences would preclude the Argonaut Project from building on the existing profile to converge with the HEART Project for future use cases.