A White Paper on

Access Control with FHIR

Ryan M. Harrison
Henry Mayen Velasquez
Matthew McCall

Amida Technology Solutions, Inc.
December 2021

Summary

HL7 FHIR (Fast Healthcare Interoperability Resources) is a "platform specification" for health data exchange. An important enabler of data exchange is access control, specifically AuthN (authentication: who are you?) and AuthZ (authorization: what are you allowed to do?).

We provide an overview of SMART on FHIR, the de facto standard for role-based access control (RBAC) with FHIR. Since there is no standard for attribute-based access control (ABAC) with FHIR, we discuss available implementations.

Although we briefly recap the most essential topics in footnotes, this white paper assumes an introductory understanding of RBAC versus ABAC access control and FHIR.

Role-Based Access Control with "SMART on FHIR"

Access control is comprised of two actions, *authentication* (AuthN) and *authorization* (AuthZ). Authentication is the process by which a system verifies the identity of an individual or resource requesting access. Authorization is the process to ensure that authenticated requests access the appropriate resources.

Role-based access control (RBAC) defines a set of roles and permissions. Users are assigned roles — e.g., "system administrator," "team lead," or "member."[1] Each role is assigned a set of permissions — e.g., read and write access to a resource.

Using RBAC in practice requires a system of record for users, as well as a way to encode permissions and roles, which is what Substitutable Medical Applications, Reusable Technologies (SMART) is designed to provide. In this context, "SMART on FHIR" is a marketing term that refers to Open ID (AuthN) and OAuth 2.0 (AuthZ) with FHIR resources.[2]

To confuse matters, like with the term "HL7," the meaning of SMART changes depending on context. HL7 may refer to the Health Level Seven *organization*, the HL7 v2 messaging *specification*, or, less commonly, other specific projects such as HL7 v3 messaging, HL7 FHIR, or HL7 CDA. In the same way, SMART may refer to the specification for using OpenID and OAuth 2.0 with FHIR resources

[1] Best practice is to assign roles based on the principle of least privilege to accomplish well-defined business functions.
[2] OpenID Connect (OIDC) is the term for an OpenID authentication (AuthN) layer on OAuth 2.0; therefore, "SMART on FHIR" could also be considered OpenID Connect with FHIR resources.

(our usage), the specific FHIR implementation guide (SMART Application Launch Framework Implementation Guide[3]), the non-profit organization "SMART Health IT project," or the collaboration between the SMART Health IT project and the Argonaut project, which drives adoption of the specification among electronic health record (EHR) vendors and the app community.

OAuth 2.0 defines a protocol for the exchange of tokens. JSON Web Tokens (JWTs[4]) are a specific encoding for tokens. JWTs are a convenient token format for distributed web applications because they can be verified by client applications — i.e., they require no additional server-side verification or look-up. A JWT has three parts: header, payload, and signature.

The header contains the cryptographic signing algorithm (`alg`)[5] and token type (`typ`). The payload contains claims, most importantly authorization permissions (`scope`) and expiry time (`exp`). The signature allows a client to independently verify that the JWT, and thereby the JWT's claims, have not been tampered with.[6]

JWT scope Claim

The JWT specification does not specify how the `scope` claim is encoded.[7] Here, the innovations of SMART on FHIR are 1) separating "patient" and "user" namespaces, 2) tying the `scope` authorization permissions directly to FHIR resource types, and 3) standardizing the `scope` format for FHIR applications.

The SMART on FHIR Implementation Guide distinguishes between "patient-specific" and "user-level" interactions.[8] A permission prefixed with `patient/` concerns one and only one patient, while a permission prefixed with `user/` concerns all administrative functions to which that user has access. For example, viewing one's own personal health record would use a `patient/` prefixed permission, while viewing a listing of all patients on a nurses' ward would use a `user/` prefixed permission.

SMART on FHIR restricts access within `patient/` and `user/` by resource type. For example, `patient/Patient.write` allows write access to the Patient resource type for a specific patient. This would include the `POST`, `PUT`, `PATCH`, and `DELETE` REST verbs. Meanwhile, `user/Observation.read` allows read access to the Observation resource type for all observations the given user has access to. This includes the `GET` and `HEAD` REST verbs.

Taken together, the `scope` standardization of permissions takes the form:

```
patient/:resourceType.(read|write|*)
user/:resourceType.(read|write|*)
```

[3] http://hl7.org/fhir/smart-app-launch/
[4] https://tools.ietf.org/html/rfc7519
[5] The most common `alg` for JWTs are HS256 (HMAC with SHA-256) and RS256 (RSA PKCS#1 signature with SHA-256, with a minimum recommended key size of 2048 bits); the former requires a shared secret (usually base64 encoded), while the latter is an asymmetric public/private key.
[6] While the shared secret (HMAC) or public key (RSA) are required to *verify* the signature, anyone can *read* the header and payload content of a JWT. For use cases where the payload must be kept private, see JSON Web Encryption (JWE).
[7] The `scope` claim is used by convention, and is not one of the seven registered claims (https://tools.ietf.org/html/rfc7519#section-4.1).
[8] http://hl7.org/fhir/smart-app-launch/scopes-and-launch-context/index.html#patient-specific-scopes

… where `:resourceType` is a single FHIR resource. Commonly scoped FHIR resources include Claim, ExplanationOfBenefit (EOB), Patient, Practitioner, Observation and Organization.

Example with SMART on FHIR access_token

JWTs are generic and can be used for all the common token types: `access_token`, `id_token` (identity token), and `refresh_token`. Here, we demonstrate a worked example of a SMART on FHIR `access_token` used for authorization.

Base64 serialized `access_token`, with the format `header.payload.signature`:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwczovL2Voci5leGFtcGxlLmNvbS9maGlyIiwi
aWF0IjoxNTYzMjI3NzQ5LCJleHAiOjE1OTQ3NjM3NDksImF1ZCI6Im15c21hcnRvbmZoaXJhcHAuZXhhbXBsZS5jb
20iLCJzdWIiOiJteXVzZXJAbXlzbWFydG9uZmhpcmFwcC5leGFtcGxlLmNvbSIsInNjb3BlIjoib3BlbmlkIGZoaX
JVc2VyIHBhdGllbnQvKi5yZWFkIn0.BXs6kt7nrMxGEkOGUtUeZt34smU6Mdt1VjHzrX2QoUg
```

With the de-serialized contents:[9]

```
Header
{
  "typ": "JWT",
  "alg": "HS256"
}
Payload
{
  "iss": "https://ehr.example.com/fhir",
  "iat": 1563227749,
  "exp": 1594763749,
  "aud": "mysmartonfhirapp.example.com",
  "sub": "myuser@mysmartonfhirapp.example.com",
  "scope": "openid fhirUser patient/*.read"
}
Signature
{
  HMACSHA256(base64UrlEncode(header) + "."
  +base64UrlEncode(payload), secret <SMARTonFHIR>)
}
```

This allows access to an `id_token` (`scope: openid fhirUser`) and all patient data (`scope: patient/*.read`).

[9] Try for yourself at jwt.io.

Limitations of SMART on FHIR

It is important to remember that the original Argonaut use cases for SMART on FHIR were connecting patient- and clinician-centric applications to EHRs. That initial scope has been clarified to include launch profiles (launched from within an EHR or standalone), `online_access`, and `offline_access` (allows `refresh_token` requests).[10]

The needs of payers and the healthcare industry's transition toward value-based care (VBC) contracting are outside the scope of SMART on FHIR. VBC falls within the remit of the HL7 Da Vinci project,[11] whereas other payer-specific use cases may not have a designated project or working group. Payer use cases that overlap the patient- and practitioner-specific workflows envisaged by SMART on FHIR may be supported, but this must be evaluated on a case-by-case basis.

These overlapping use cases require interaction with one record (personal health record) or a relatively small number of records (practitioner). Therefore, bulk transfer — e.g., provider-to-provider or provider-to-payer transactions with millions of records — are outside the scope of SMART on FHIR.

While simple queries are supported via `GET` requests with query parameters on resources,[12] and across resources via the Search resource type,[13] there is not yet a standard for complex queries with conditionals within FHIR or SMART on FHIR.[14]

SMART on FHIR relies upon RBAC at the FHIR resource level of detail. It allows permissioning by resource type — e.g., `patient/Patient.read` vs. `patient/AllergyIntolerance.read`. FHIR has a security model that includes designations for business-, individual-, and patient-sensitive data at the resource level. Neither FHIR nor SMART on FHIR specifies how to use these security designations. Further, SMART on FHIR does not allow restrictions based on attribute within a resource type — e.g., `Patient.maritalStatus`. Nor does it allow conditional restrictions by either resource or attribute — e.g., allow Medication if `Medication.identifier` is not protected.[15]

[10] SMART on FHIR is a major building block in the emerging ecosystem for clinical workflow integration. Third-party implementation guides for Clinical Decision Support (CDS) hooks and FHIRcast build upon SMART on FHIR to allow real-time in-EHR messaging and synchronization across applications within a clinician workflow.
[11] http://www.hl7.org/about/davinci/index.cfm
[12] http://hl7.org/fhir/patient.html#search
[13] http://hl7.org/fhir/search.html
[14] The authors are unaware of a method for querying FHIR servers with CQL directly. FHIR search does not support Clinical Quality Language (CQL) queries; however, there is a reference implementation that uses CQL (and FHIRPath, basically XML XPath for FHIR) on FHIR bundles (https://github.com/cqframework/cql-exec-fhir).
[15] Protections are often used on medications prescribed for HIV, mental health and substance use disorders, because the medication implies a specific (potentially stigmatizing) condition. For example, from an unprotected medication record of Zubsolv, one could imply that the patient is in a drug treatment program for opioid addiction.

Attribute-Based Access Control

Attribute-based access control (ABAC) grants rights to the user via policies. Policies are almost always at the attribute level and usually allow conditional statements — e.g., allow <X> if <Y>). eXtensible Access Control Markup Language (XACML) is the de facto standard policy language. Because XACML is cumbersome to write directly, policies are in practice authored with a GUI editor[16] or transpiled from pseudocode such as Abbreviated Language for Authorization (ALFA).[17]

To the best of our knowledge, there is no reference implementation for ABAC with FHIR.[18] FHIR does provide the Security resource type and SecurityLabel concept, which could be used in an ABAC control flow. Indeed, the FHIR R4 documentation for SecurityLabel explicitly states, "Local agreements and implementation profiles for the use [of] security labels should describe how the security labels connect to the relevant consent and policy statements."[19]

Additionally, an AuditEvent resource type[20] is available to maintain an access control log, which is a feature of all mainstream ABAC solutions. Both Security and AuditEvent are at trial use in FHIR R4.

FHIR usually relies on an implementation guide to specify CodableConcepts and ValueSets specific to an application or localization. For example, the US Core implementation guide provides resource profiles with localization to US Office of the National Coordinator for Health IT (ONC) guidelines.[21] Security differs in that it relies directly upon the Healthcare Privacy and Security Classification System (HCS) for the five supported security labels. We suspect this choice was made because, while the label categories are enumerated by HCS, their highly localized use would make an implementation guide rather narrow in scope.

As of R4, security labels are ascribed in the metadata of resources.

```
"resource": {
  "id": "1",
  "meta": {
    "security": [{system, code, display}, ...]
  }
}
```

[16] XACML editors from WSO2 and Axiomatics are proprietary. The open source UMU XACML Editor has not been updated since 2013 (https://sourceforge.net/projects/umu-xacmleditor/).
[17] ABAC vendor Axiomatics donated ALFA, originally a proprietary language, to the OASIS XACML Technical Committee in 2014. Unfortunately, while the ALFA standard is technically "open," implementations to author and transpile ALFA have restrictive licenses imposed by Axiomatics. For example, the Eclipse plugin for ALFA is restricted to non-commercial use (https://www.axiomatics.com/alfa/).
[18] In 2017, the US Substance Abuse and Mental Health Services Administration (SAMHSA) developed the open-source Consent2Share (https://github.com/bhits/consent2share) patient consenting application using ONC Data Segmentation for Privacy (DS4P) guidelines. The documentation for a piece of middleware (https://github.com/bhits-dev/context-handler), which connects the Policy Enforcement Point (PEP) and Policy Decision Point (PDP), says, "As an alternative, this API can also be configured to retrieve XACML policies from a FHIR server." The Consent2Share application was last updated in September 2018 and does not appear to be actively maintained. Our presumption is that Consent2Share is restricted to resource-level policies. In any case, the implementation details of how the application handles FHIR resource-level access control could be instructive when architecting an ABAC solution for FHIR.
[19] https://www.hl7.org/fhir/security-labels.html
[20] https://www.hl7.org/fhir/auditevent.html
[21] Specifically, the ONC US Core Data for Interoperability (USCDI) and ONC Common Clinical Data Set (CCDS).

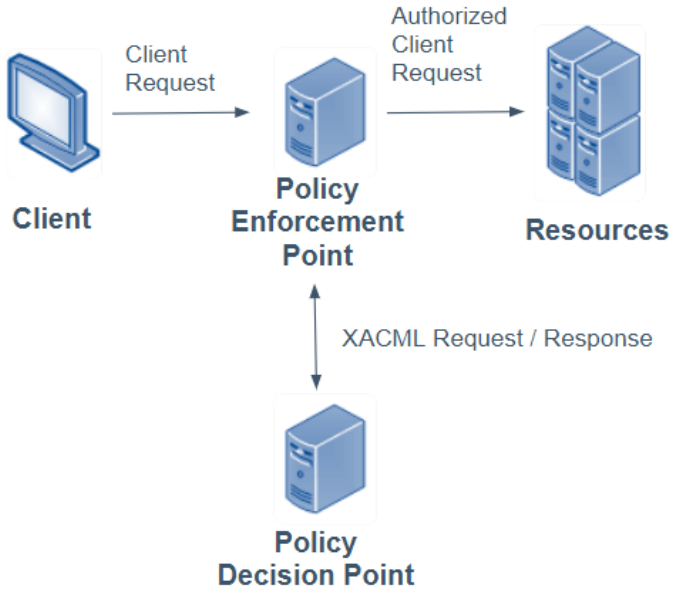

***Figure 1** – ABAC schematic that highlights the relationship between the client application, Policy Enforcement Point (PEP), Policy Decision Point (PDP), and resource server. The Policy Information Point (PIP) and Policy Administration Point (PAP) are not shown.*

Indeed, SecurityLabel can only be applied to the entire resource, not to individual attributes. For example, the Medication resource for the antidepressant Lexapro could have a SecurityLabel applied — e.g., Sensitivity Category, MH (mental health information sensitivity). This use of SecurityLabel is appropriate because each of a patient's medications are separate resources.

However, because most FHIR resource attributes do not allow arbitrary metadata (for example, in the Medication resource type, there is nowhere within `Medication.identifier` or `Medication.code` to append `meta.security`), there is currently no supported way to encode attribute-level security labels.

```
{
  "resourceType": "Medication",
  "identifier": [{ Identifier }], // Business identifier
  "code": { CodeableConcept },    // Taxonomy codes
  …
}
```

As of FHIR R4, SecurityLabel has no facility for conditionality — e.g., allow access IF.

Broadly, ABAC on FHIR could be encoded within FHIR objects or outside FHIR objects. Within FHIR objects, the most obvious paths forward are encoding security labels for attributes within `resource.meta` or using a new FHIR resource type. Security labels within the FHIR resource metadata are advantageous because they would be sent with the resource by default at exchange. For example, a hypothetical `Patient.meta.security.attributes.gender` could encode a Sensitivity Category (GENDER: gender and sexual orientation information sensitivity) specific to gender identity, rather than covering the entire Patient resource. The downside of this approach is

that it requires updating the base FHIR specification, or authoring one or more custom profiles or extensions.

For encoding outside FHIR objects, the most obvious path forward is a FHIR-specific PIP within an ABAC implementation, which allows reuse of the PEP and PAP (Figure 1). Indeed, a literature review reveals this approach in the 2017 paper "Attribute Based Access Control for Healthcare Resources."[22] Briefly, the authors describe their custom integrations with the WSO2 Identity Server, specifically their writing a PIP with XACML encodings of FHIR resource attributes. The advantage and downside of this approach is that it operates outside the FHIR specification — i.e., the policy would not be communicated in a FHIR exchange.

Hybrid RBAC/ABAC approaches are also possible. For example, Janki et al. describe a hybrid approach using a LoopBack proxy server between the client application and the FHIR resource server (HAPI FHIR) that enforces resource-level security labels at the proxy.[23]

About Amida

Amida is a software company focused on enterprise data management, cybersecurity, and digital platform strategies. We design, develop, and deploy systems that enable the secure and reliable exchange of sensitive information. Amida builds open-source solutions that collect and prepare data from a variety of sources – independent of structure, format, provenance, and schema – for applications like business intelligence, predictive analytics, and downstream transactions. We are especially well-known for open data architectures and production services that are scalable, efficient, modular, and secure. Our software engineers and data scientists have extensive experience in data modeling, governance, interoperability and exchange, and visualization, especially in health IT.

Amida's founding team co-conceived and led the design, implementation, and production deployment of the Blue Button personal health record at the Department of Veterans Affairs (VA), and they supported its development and deployment at the Centers for Medicare and Medicaid Services (CMS) and the Department of Defense (DOD) Military Health System. They co-conceived and led the creation of the Joint Legacy Viewer, a clinician portal used by hundreds of thousands of VA and DoD healthcare providers every day. This portal is the cornerstone of both agencies EHR modernization efforts. They also led the design and prototype construction (the "Virtual Regional Office") for the service-connected disability claims platform, which is still in enterprise service today.

[22] A non-peer reviewed master's thesis published in 2015, Ruiz's "FHIR: Cell-Level Security and Real Time Access with Accumulo" predates this work (https://mountainscholar.org/handle/10976/166562).
[23] "Authorization solution for full stack FHIR HAPI access." 2017. DOI:10.1109/nc.2017.8263266