Security Working Group

Security and Privacy Issues in FHIR Subscription

Mohammad Jafari,
Kathleen Connor, John M. Davis, Christopher Shawn

Version 1.1

December 17 , 2019

(revised for publication on 1/20/2020)

Table of Contents

1 INTRODUCTION

This report presents an overview of the concept of *subscription* in the Fast Healthcare Interoperability Resources (FHIR) standard and provides discussions on the corresponding security and privacy concerns, especially in the context of the *share-with-protection* paradigm. Aligned with the existing specifications, an architecture is also proposed for a FHIR subscription service which addresses the primary security and privacy concerns.

Subscription defines a mechanism for sending event notifications, which may include health information, by a FHIR server to a recipient, such as an organization, an electronic health records (EHR) system, or a mobile or web application. Events which can be the basis for subscriptions are referred to as Topics.

FHIR, the next-generation standard for health information, defines data structures known as *resources* for representing various types of health information, as well as an application programming interface (API) for working with these resources. FHIR API has been primarily developed based on the Representational State Transfer (REST) model in which standard operations of the HyperText Transfer Protocol (HTTP) are leveraged to define standards for reading, searching, creating, updating, and deleting FHIR resources. However, REST follows a pull-based model in which every interaction is initiated by a client and the server can only send data to a client if the client sends a request first.

This pull-based model is not suitable or efficient in use-cases where the client needs to react to an event which takes place on the server by receiving information about that event from the server; in other words, use-cases in which effective implementation depends on a push by the server, without a request from the client.

For example, a calendar app (mobile or web) which tracks and maintains the appointments for a clinician, needs to know when a new patient has been referred for scheduling an appointment. In the pull model, the app (or a service on its behalf) needs to actively and periodically query (poll) the server to see if there are any new referrals. This requires the app to record the state of the appointment requests and compare it with the latest state from the server based on periodical queries, effectively maintaining a cache of the resources, so that it can detect the *event* of a new referral on the app's side based on that information.

If polling is frequent, it will take up bandwidth and computational resources which will burden both the client and the server. This is in addition to the complexity and computational resources required to maintain the cache on the client side. On the other hand, if the polling is infrequent, the app will have a noticeable delay in processing the event; for example, if the app polls the server every hour, it may be out of sync with the server for up to an hour. Such delays may not be acceptable for some use-cases where response time is crucial.

Having a mechanism for the server to push notifications to the app can significantly improve the effective implementation of use-cases like the one discussed above. Rather than having the app dedicate resources to actively querying the server and the server needlessly responding to many such queries while the state has not changed, the server can notify the app once and only when the event of a new appointment request has occurred. The app can, in turn, process the notification once and only when this event occurs.

FHIR Subscription is aimed at resolving this problem by defining a mechanism for clients to subscribe to specific server events (known as Topics) and subsequently receive push notifications

when the event occurs over a pre-defined protocol, such as a REST API callback, a FHIR message [1], or an email.

1.1 Related Technologies

Over the past decade, the need for dynamicity and reactiveness in web applications has led to the concept of *webhooks* which has become an inseparable part of most modern web application frameworks and services. Webhooks provide a call-back mechanism which a server can use to notify a client about an event that happened on the server side. Webhooks are often a REST-style call sent in the form of an HTTP POST request to an endpoint previously registered by the client. The body and the headers of this HTTP call include information about the event, or links the client can follow to find such information. Webhooks are the most common form of subscription delivery in modern web applications.

The REST Hook specifications developed by Zapier [2] utilize the basic concept of webhooks and expand it to define a comprehensive subscription service in which client can self-register to receive webhooks and manage their own subscriptions. These specifications also discuss some of the broad concerns around security and performance.

In the healthcare domain, the specifications for Document Metadata Subscription (DSUB) by Integrating the Healthcare Enterprise (IHE) [3] define the transactions and flows for a subscription system in a Cross-Enterprise Document Sharing (XDS) environment where recipients can register to receive notifications about the availability of documents based on specific criteria of interest. These specifications are based on the Organization for the Advancement of Structured Information Standards (OASIS) Web Services Base Notification [4], an older set of specifications which define a subscription and notification system in the context of traditional Service-Oriented Architecture (SOA) based on Simple Object Access Protocol (SOAP).

The evolving FHIR specifications for subscriptions follow the lessons learned and the broad concepts from these preceding specifications.

2 ANATOMY OF A SUBSCRIPTION SERVICE

As shown in the class diagram in Figure 1, a Subscription is an entity that records the association between a recipient and a Topic (event) which denotes that the recipient must receive notifications when the Topic event occurs. Subscriptions may have further restrictions such as a period of validity (end date and time).

Subscriptions are created and managed by clients. The client creating a subscription is associated with the recipient receiving the subsequent notifications. Often, the client and recipient are the same entity.

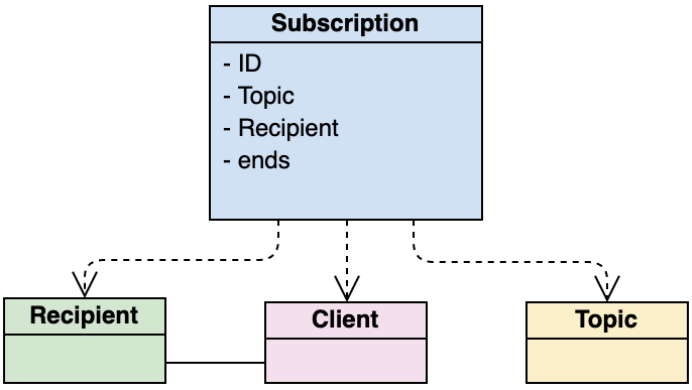

Figure 1: Subscription class diagram

Providing subscription-based notifications requires the FHIR server to implement a set of mechanisms collectively referred to as a Subscription Service. A Subscription Service enables a Client (which may be a recipient) to create Subscriptions to Topics on behalf of Recipients, and ensures that the server subsequently delivers notifications to the Recipient every time the respective Topic event occurs. The components of such a service and its clients are discussed in this section.

2.1 Subscription Topics

Subscription Topics are well-defined events to which a client can request to subscribe. The straightforward examples are the standards events (i.e., create, update, and delete) on specific FHIR resource types. A Subscription Service must provide potential clients with a mechanism for retrieving a list of all available Topics for subscription. Moreover, the Subscription Service may also allow clients to define custom Topics narrowing down a general Topic by configuring its parameters.

In the current[1] proposal for FHIR subscriptions, a FHIR resource named Topic is used to represent subscription Topics. Clients can obtain a list of subscription Topics in a FHIR Server using the FHIR search API. Clients can also define narrow and refined events by creating new Topic resources by adding specific filters in the form of FHIR search queries. For example, the general Topic of "update events to Immunization resources" can be narrowed down by adding

[1] As of the time of writing, this proposal is in the Argonaut branch of FHIR specifications and has not be merged into the main build yet:
- http://build.fhir.org/branches/argonaut-subscription/subscription.html
- http://build.fhir.org/branches/argonaut-subscription/subscriptiontopic.html

more parameters, such as “update events to Immunization resources administered at care facility X”).

The Topic resource also allows a FHIR server to provide opaque Topics for events with complex logic that cannot be expressed in the form of FHIR standard events and query parameters. Such Topics provide a textual description for the event they capture and allow clients to use them in subscriptions based on the assumption that the client and server have a mutual understanding of the semantics of the Topic.

2.2 Subscriptions Management

Since manual setup and maintenance of subscriptions is inefficient and unscalable, the Subscription Service must provide clients (with sufficient authorization) the tools and the interface for setting up and managing their subscriptions.

In the current FHIR specifications, this is done by the Subscription resource. By creating, updating, or deleting resources of this type, clients can create subscriptions, activate or deactivate them, change their parameters, or remove them entirely. Clients can also use the FHIR search API to search their active and past subscriptions.

The Subscription resource enables defining and updating the details of a subscription by specifying or changing the following parameters:

- The Topic of the subscription specifying the event that should trigger a notification,
- The identity of the recipient that should receive the notification,
- The notification protocol and delivery mechanism, such as webhooks or email,
- An expiration date at which point the subscription should be deactivated, archived, or deleted,
- Frequency of delivery, such as instantaneous notifications or a summary by some frequency, and
- Payload of the notification and its format; for example, whether or not the resource content should be included in the notification. If resource content are included, the preferred format (e.g., JavaScript Object Notation (JSON) or Extensible Markup Language (XML)) may also be specified.

2.3 Event Monitor

Event Monitor is the component in charge of observing the FHIR server’s internal functions and determining whether a noteworthy event –from the perspective of the subscription system– has occurred. Noteworthy events are any event associated with valid subscription Topics supported by the FHIR server.

The Event Monitor is a component deeply tied to the internal functions of the FHIR server such as the web server, the core business logic implementation, and the database. How to determine whether a noteworthy event has occurred is deeply dependent on these implementation details and beyond the scope of the FHIR specifications.

2.4 Notification Delivery

Delivering notifications when the event associated with the Topic occurs is the final piece in the Subscription Service. Triggered by the Event Monitor once an event of interest has occurred, the Notification Delivery component creates, packages, and sends a notification to all corresponding subscription recipients in suitable format and over the configured protocol.

The Notification Delivery component provides the connective tissue and the interface between the internal implementations-dependent parts of the Subscription Service (i.e., the Event Monitor) and the standardized parts of the FHIR server, such as the Subscription resource and the delivery mechanism to the recipients.

This component provides the implementation for all the various notification protocols supported by the Subscription Server and permitted in the Subscription resource, such as webhooks, FHIR messaging, or email.

Depending on the implementation decisions, the Notification Delivery component may maintain a state for the notification delivery and provide a window of re-attempts in case of a failure. This is important in use-cases where notifications are crucial and there needs to be a level of tolerance for cases such as momentary errors or unavailability. The FHIR specifications also define an attribute in the Subscription resource for recording error status or messages. This helps clients to inspect and troubleshoot errors in their Subscriptions.

2.5 Notification Processing

The recipient is in charge of receiving and consuming notifications it is subscribed to. While not within the boundaries of the FHIR server's subscription service, this step is important for the correct implementation of any use-case that relies on subscriptions. The recipient must provide a correct implementation of the notification protocol registered by its subscriptions and ensure the availability and health of the receiving endpoints. Moreover, the recipient must have the capabilities to parse and process the content of the notification if the subscription is configured to include payloads.

Some notification mechanisms, such as HTTP POST (e.g. webhooks or FHIR messaging), require a public API endpoint accessible to the FHIR server; this essentially compels the recipient to run an HTTP server. Such servers must comply with all the general applicable privacy and security requirements a discussion of which is beyond the scope of this report.

3 SECURITY AND PRIVACY CONSIDERATIONS

This section discusses the major security and privacy considerations in a subscription system. Note that the general security and privacy requirements applicable to all open HTTP servers, or any other generic requirements specific to a delivery mechanism which are orthogonal the subscription use-case are not discussed in this report.

3.1 Authorization for Subscription Management

Subscription management operations, such as creating new subscriptions or modifying or terminating them, must be restricted to authorized clients and the FHIR server must ensure that the client requesting any of these function is sufficiently authorized.

In the current and upcoming FHIR specifications, subscription management is performed via standard operations (create, read/search, update, and delete) on the relevant FHIR resources (i.e., Subscription and Topic). Thus, the existing FHIR access control technology can be easily leveraged for enforcing authorization in these cases by controlling access to these two resources.

However, it is still noteworthy that, unlike most FHIR resources, which are simply passive containers of medical information, subscription-related resources lead to active consequences in the behavior of the FHIR server over a potentially long period of time; therefore, it is advisable to distinguish subscription resources from other resources and require explicit clearances for clients requesting subscription management operations. Section 4 provides an example of how to achieve this using OAuth authorization scopes.

3.2 Recipient's Consent

In the request to set up a subscription, the client asks the FHIR server to send notifications to a recipient. In this transaction, the request comes from the client but the notifications will go to the recipient. While it is often the case that the client and the recipient are the same entity or controlled by the same organization, it is also possible for the client and the recipient to be entirely separate entities. For example, a patient-controlled app can request to set up a subscription for sending updates to a clinician's system.

Thus, the FHIR server must ensure that the recipient is aware of, and consents to receiving notification pushes before accepting and activating the subscription. An obvious example for this is the confirmation step in email subscriptions that ensures the owner of the email address is aware of, and accepts receiving the notifications.

Similarly, HTTP-based subscriptions (e.g., webhooks or FHIR messaging) need the verifications to ensure recipients are not subscribed to unwanted notifications by malicious or misconfigured clients. Also, from the FHIR server's perspective, sending notifications is a costly operation which takes up computational and communication resources, so, the FHIR server also needs to ensure that it is not wasting its resources on unwanted notifications.

The simplest form of confirming the recipient's consent is to look into the recipient's response to the first notification. An HTTP OK is interpreted as the recipient's consent to receiving the notifications. Although this provides *some* level of control by the recipient over receiving notifications, it is not quite sufficiently granular about the details of the subscription unless the first notification includes more details (e.g., a link to the Subscription resource that is the origin of the notification). In order to ensure robustness against momentary errors, the first notification may be reattempted a number of times to ensure that the non-OK response from the recipient is deliberate.

A more comprehensive mechanism for a recipient's consent is discussed in Section 3.6.

3.3 Authorization Throughout Subscription Lifetime

While a subscription can remain active over a long period of time, the access control system of the FHIR server only verifies the client's authorization at the moment of creating or updating the subscription. This leaves a gap in ensuring that the subscription *remains* authorized throughout its lifetime, as the consequences of an active subscription go far beyond the instant at which it is set up. For example, the FHIR server must ensure that the expiry date of the subscription is on or before the expiry date of the client's authorization scopes. In other words, a client must not be able to bypass the temporal limits of its granted permissions by creating subscriptions that will remain active beyond the client's authorization period. For example, if an app is authorized by a patient to access some of the patient's information for a month, the app must not be able to create a subscription that remains active more than a month.

One of the immediate problems in this regard is the complexity caused by short-lived OAuth access tokens. OAuth access tokens are often issued to last for very short periods of time in the order of magnitude of minutes or seconds. In cases where a client needs access for a longer period of time, a long-lived refresh token is issued by the OAuth server; the client can use the refresh token to request a fresh access token once an old one expires, without repeating the full authorization process [5].

When a client presents an access token to the server for creating a subscription, that access token usually expires within minutes, so, if the server does not allow the subscription to remain active beyond that, all subscriptions will expire within minutes of creation and have to be renewed periodically. This is impractical and in essence contrary to the point of having subscriptions – which is to serve notifications to a recipient without frequent requests from a client. Thus, the need for subscriptions to be long-lived is somehow inherent in the core concept of subscriptions.

One solution is to rely on explicit scopes that grant subscriptions, or at least indicate that the client possesses a refresh token. For example, in some implementations of OAuth, granting the scope `offline` indicates that the client has also been issued a refresh token and, therefore, is able to acquire a new token once the current short-lived access token expires. Or, more explicitly, a special FHIR scope (e.g., `subscribe`) can be used to indicate that the client is allowed to create subscriptions without a limit on their expiry. In cases where the refresh token has an expiry date, or the OAuth server decides that the client's subscriptions should be bound to a limited time, an extension to the scope can note the expiry date. For example, `offline;expires_in:3600` or `subscribe;expires_in:3600` denote that the scope is granted only for a limited amount of time. The server must enforce such restrictions on the creation of, and updates to, subscriptions.

Another complexity may ensue if the OAuth server allows changing the scopes associated with a client's token, without forcing re-authorization. For example, consider the case of a patient granting an app access to their information including creating subscriptions up to a year. Now if the OAuth server allows the patient to change this scope (without revoking the old one and forcing new authorization), the patient can, for example, change the limit on creating subscriptions to six months. While this update to scopes is reflected in all the subsequent access tokens presented by that client to the FHIR server, there is no way for the FHIR server to adjust active subscriptions, previously created by that client to abide by the new restrictions. In fact, the FHIR server may not even become aware of this change if the client does not interact with the FHIR server after the change takes place.

Changing a client's scopes is very similar to the case of revoking a client's authorization, and in fact, revocation could be thought of as taking away all the granted scopes to the client. So, this case is further discussed in Section 3.4 alongside revocation.

3.4 Revocation of Authorization

Refresh tokens are usually issued either without an expiration date (i.e. in perpetuity), or with expiration dates long into future. Therefore, if a client's access needs to be terminated for any reason its existing refresh token must be revoked. This usually takes place by interacting with a user interface provided by the OAuth server with which the original granter of access (e.g., the patient) review all granted authorizations and can choose to revoke them. There are also specifications for revoking refresh tokens using an API, which allow a client to programmatically revoke a refresh token [6].

After the revocation is recorded, the OAuth server will immediately cease issuing access tokens based on the revoked refresh token. This will ensure that, after a brief window of time in which short-lived access tokens may still be valid, no new access tokens will be granted to the client and the client's access to the server will be terminated.

Although this flow works well for the authorization of regular transactions, it is not adequate for subscriptions. A subscription created and activated based on a access token (backed by a valid refresh token), will continue to send notifications and share information with the recipient even after the revocation of the original scopes, based on which the subscription was set up. This is not acceptable considering that the FHIR server is expected to terminate all such subscriptions when the client loses its authorization.

The current standard OAuth 2.0 specifications do not provide a mechanism for actively notifying partner servers about the revocation of refresh tokens. So, the FHIR server has no way to know about such events. Some OAuth implementations, however, provide a notification mechanism to which partner servers can subscribe and receive alerts when a refresh token is revoked (e.g., [7]). Note that since clients never share refresh tokens directly with a resource server, the revocation notification must include the client's identifier, so that the resource server can identity which client's access is being revoked. After receiving such a notification, the FHIR server is able to find all the subscriptions associated with that client and deactivate or delete them.

This notification mechanism can also solve the problem of scope changes in a similar manner. When a client's granted scopes change, the OAuth server can send a notification to the FHIR servers detailing the changes. This notification can trigger the process of adjusting the subscriptions tied to the client whose granted scopes has been modified, in accordance with the new scopes, which could lead to modification or deletion of some subscriptions.

Note that as an implementation point, the FHIR server must record the client identifier with any subscription created or modified by that client, and retain this association in order to be able to make adjustments based on the revocation or modification of the client's scopes.

3.5 Security Labeling and Privacy-Preserving Services

FHIR specifications allow clients to configure a subscription to send the complete content of the corresponding resources. This, essentially, results in a form of exchange where healthcare information are sent to recipients periodically in the course of fulfilling an active subscription. Like other forms of exchange, sharing information via a subscription, should be subject to the labeling requirements and the fine-grained processing by the Security Labeling (SLS) and Privacy-Preserving Services (PPS).

Invoking the SLS will ensure that all outgoing data is labeled with appropriate security labels, including handling instructions, which could be specific to the recipient. The PPS sifts through the outgoing data and removes resources for which the recipient's clearances are not sufficient. Other forms of privacy-preserving functions may entail resource content transformations such as applying de-identification algorithms.

3.6 Notification Authenticity

Notifications inform the recipient about an event on the server side and usually play an important role in the recipient's healthcare workflow. When there is a payload in the notification, the medical information in the payload will also be used in making clinical decisions. Therefore, it is important for the recipient to determine the authenticity and trustworthiness of the notification and enclosed data to ensure it has actually originated from the expected FHIR server.

A standard solution to this problem is to use digital signatures and have the server digitally sign the notification messages. This can be set up either at the application level by explicit signing of the payload and headers, or by the transport layer, by using the Transport Layer Security (TLS) with client authentication enabled.[2]

Since using digital signatures usually brings about the complexity of the public-key infrastructure, simpler solutions are often used to verify the authenticity of webhooks, based on a shared secret between the FHIR server and the recipient. The shared secret enables using symmetric signatures (e.g., based on a keyed hash function) to ensure the authenticity of the data. The secret can be set by the client when setting up the subscription and it can be rotated via periodic updates to reduce the risk of compromise. Note that the recognition of this secret by the recipient also implies the recipient's consent in receiving the notification; if the recipient does not agree to receiving the notifications, the signature of the data (which relies on the secret) would not seem valid to the recipient. Since the client is the one setting up the secret with the subscription service, the recognition of the secret by the recipient, also authenticates the relationship between the client and the recipient.

Using OAuth to control access to the recipient's notification endpoint is another solution. This requires that in the course of creating a subscription (or any subsequent changes) the FHIR server follow the OAuth protocol (with a designated OAuth server) to obtain a token for contacting the recipient's endpoint. Since the OAuth flow incorporates the recipient's consent, this solves the recipient's consent problem as well as the authenticity of the notifications. The recipient's designated OAuth server can be set by the client as an attribute of the subscription at the time of creation.

Although using OAuth is a more comprehensive solution, it involves more moving parts and therefore more configuration and more points of failure. Ultimately, this solution is more complicated to configure and enact, especially if the FHIR server is issued refresh tokens, which compels the FHIR server to communication with the OAuth server to obtain an access token before sending every notification to every recipient –an overhead that could nearly double the computational and communication resources necessary for handling subscriptions.

[2] Note that when a notification is sent to a Recipient, the FHIR server is the one sending an HTTP request and therefore takes the role of the HTTP "client."

4 PROPOSED ARCHITECTURE

This section discusses the proposed architecture for implementing a FHIR Subscription Service, in line with the current specifications and by considering the security and privacy requirements discussed in Section 3. The entities and components in this architecture correspond with those discussed in Section 2. This architecture is based on using OAuth 2.0 as the authorization system which assumes an OAuth 2.0 server to which clients are onboarded (via OAuth registration) and on which the FHIR server has been configured to rely.

The rest of this section discusses the details of three major flows of this architecture for creating or updating subscriptions, delivering notifications, and revoking or changing the authorization scopes of a client.

4.1 Creating and Updating Subscriptions

The starting point for a subscription is a client's request to create a subscription or update an existing one. Figure 2 shows the flow for creating or updating a subscription as discussed in the rest of this section.

The client is assumed to have acquired an access token from the OAuth server following one of the OAuth flows.

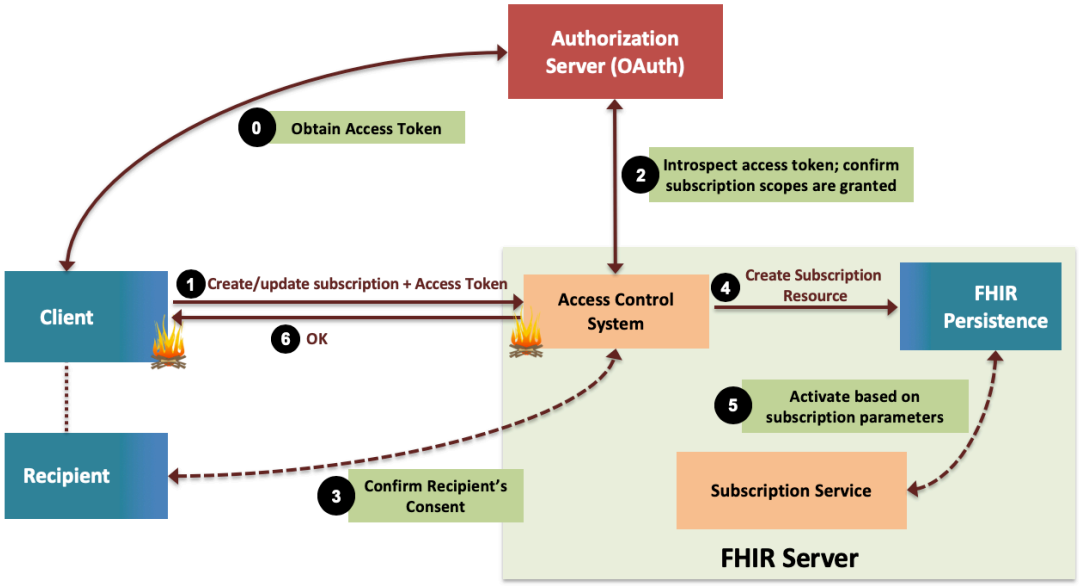

Figure 2: The Flow for Creating or Updating New Subscription

4.1.1 Client Request

The flow starts by the client's request to the FHIR Server to create or update a subscription. This request includes the client's access token as well as the attributes of the subscription based on the FHIR specifications for the Subscription resource and the FHIR standard API (often REST). The client may have previously created custom FHIR Topic resources to be referenced in this request.

4.1.2 Request Authorization

The client's request is captured by the FHIR Server's Access Control System (ACS). The ACS examines the access token presented in the request and extracts the authorization scopes associated

with the access token. Unless the access token is self-sufficiently structured, this requires running the OAuth Introspection protocol to inquire about the validity of the token as well as the assigned scopes from the issuing OAuth server.

The ACS ensures that the client has been granted sufficient scopes for creating or updating subscriptions and rejects the request otherwise.

Based on the client's authorization scopes, the ACS also determines the maximum cap on the validity of requested subscription. Depending on design decisions and use-case requirements, if the validity period of the requested subscription exceeds the client's scopes, the ACS may either reject the client's request altogether or accept the request but adjust the expiration date on the requested subscription accordingly.

4.1.3 Recipient's Consent

The ACS also identifies the recipient of the subscription request and follows the necessary steps for ensuring recipient's consent. The details for this step depend on the delivery protocols, for example webhooks or email (see Section 3.2 and 3.6).

4.1.4 Persisting the Subscription

After passing all the authorization checks, the ACS persists the Subscription resource in the FHIR persistence layer, usually a database. This record includes all the visible FHIR attributes as defined by the FHIR specifications, but must also include a record of the requesting client ID which is necessary for tracking the subscriptions created by a client, should the client's authorization scopes change or get revoked.

4.1.5 Activating Subscription

Persisting a subscription ensures that the details of the subscription are recorded but in order for a subscription to be actually enacted, the FHIR Server must activate it and invoke the Subscription Service to start delivering notifications. Activating subscriptions is usually hooked to the event of creating a new subscription.

4.1.6 Response to Client

Once the subscription has been created or updated, and then activated successfully, an OK response is sent back to the client. Note that depending on the design, the OK response may also be sent right after the successful persistence of the subscription without waiting for its activation.

4.2 Delivering Notifications

Subscriptions are ultimately actualized by delivering the appropriate notification to the recipient once a relevant event occurs in the FHIR server. Figure 3 shows the flow for notification delivery discussed in this section.

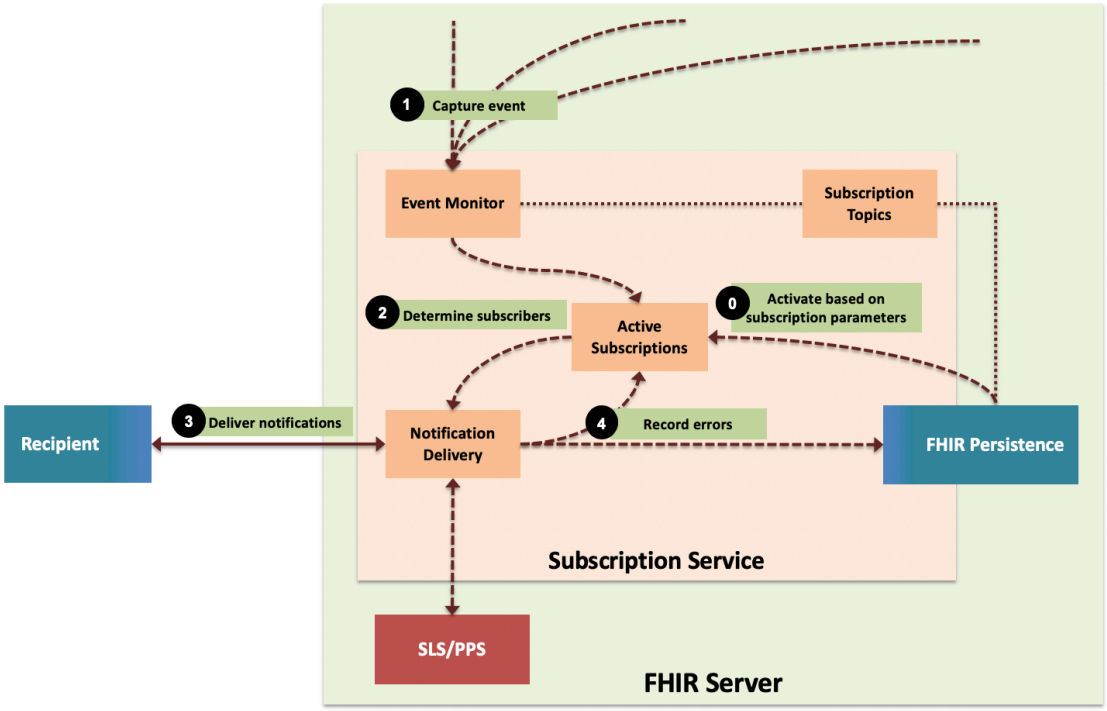

Figure 3: The Flow for Notification Delivery

4.2.1 Capturing the Event

Based on the subscription Topics supported by the FHIR server, the Event Monitor component of the Subscription Service must be aware of potentially many various events within the functions of the FHIR server in order to determine an event associated with a subscription Topic has occurred. As discussed earlier, since the details of the functions of a FHIR server are not standardized, this is deeply implementation-dependent and intertwined with the internal functions of the FHIR server. The outcome of the Event Monitor is a trigger to indicate an event associated with a subscription Topic has occurred.

4.2.2 Determine Subscribers

The Subscription Service maintains a list of active subscriptions indexed by the Topics they are associated with. After the Event Monitor determines that an event associated with a Topic has taken place, the Notification Delivery component examines this list to determine the set of recipients that have active subscriptions associated with the Topic and need to be notified.

4.2.3 SLS and PPS

After determining the list of subscribers for an activated Topic, the Notification Delivery component prepares the notification data objects based on the subscription configurations. If the configurations require a notification with a payload, the bundle of FHIR resources is sent to the SLS and PPS for labeling and to be subject to any required transformation, based on the context of the subscription (e.g., the identity of the recipient or the purpose of use for the subscription).

4.2.4 Notification Delivery

Once the notification is prepared, it is sent to the recipient via the mechanisms specified in the subscription configurations.

4.2.5 Statistics, Status, and Error Report

Relevant statistics, status, or error information about the delivery of notification (e.g., a failed delivery) is collected by the Notification Delivery component and is noted on the active subscription record for internal housekeeping purposes by the Subscription Service. The server's exception handling process, for example, may re-attempt notification delivery in order to be robust against momentary errors such as service or network unavailability.

Some status information will also be persisted on the Subscription resource to be available to authorized clients[3]. This information could be used by the client for generating reports, monitoring the state of the subscription, and troubleshooting.

4.3 Revocation or Change in Client's Authorization Scopes

As discussed earlier, the revocation of a client's authorization scopes or a change in them needs to be propagated and applied to any subscriptions previously created by the client to ensure that notifications are delivered only while authorized. Figure 4 shows the flow for processing revocation or change in authorization scopes as discussed in this section.

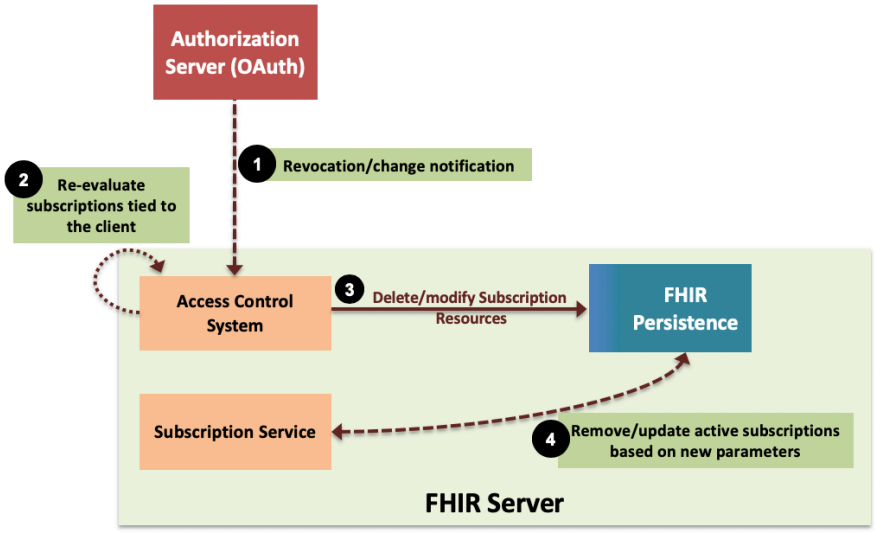

Figure 4: The Flow for Processing Revocation or Change to a Client's Authorization Scopes

4.3.1 Revocation/Change Webhook

The flow starts with a webhook from the OAuth server which informs the FHIR server that a change has happened in the client's authorization scopes, or the client's token has been revoked entirely. The webhook must include the details of the changes in the client's scopes.

In order to receive this webhook the FHIR server must have subscribed to the OAuth server's webhook services and provided an endpoint for receiving such notifications[4].

As mentioned before, these webhooks are not part of the OAuth 2.0 standard specifications, although they are essential in handling the case of subscription management.

[3] As of the time of preparing this report for publication, the community seems to be moving towards consolidating this information into a separate resource `SubscriptionStatus`.
[4] Note that this is itself a form of subscription provided by the OAuth server.

4.3.2 Re-Evaluating Client's Subscriptions

Based on the changes in the client's scopes, as notified by the OAuth server's webhook, the Access Control System of the FHIR server determines any changes that must take place accordingly to the subscriptions previously created by the client. This involves retrieving all active subscriptions created by the client based on the client's identifier. It is crucial that the FHIR server records the client identifier with the subscription record in order to be able to find a given client's subscriptions at this step.

4.3.3 Deleting or Updating Subscriptions

By re-evaluating the client's subscription, the Access Control System should proceed with adjusting the existing subscriptions; some of the client's subscriptions may be deactivated or deleted if the client has completely lost the authorization scopes required to enact them. Others could be simply adjusted via an update to accommodate the changes in authorization scopes.

5 LEVERAGING SUBSCRIPTIONS IN SECURITY AND PRIVACY SERVICES

By enabling applications outside the FHIR Server to *see* the events inside the FHIR server, subscriptions provide visibility into the inner functions of a FHIR server and an opportunity for external services to hook into those events and trigger additional processing. This provides an opportunity for other FHIR-related services, including security and privacy services such as SLS and Federated Provenance Service to be implemented outside of the FHIR server as microservices with their own independent codebase and deployment, while tightly interacting with the FHIR server through standard interfaces such as subscriptions and the REST API.

Subscriptions are a crucial piece of this puzzle, since without subscriptions, hooking into the internal FHIR server events would require implementers to hack into the FHIR server's code and customize it. That approach is far less desirable from a software engineering perspective as discussed below.

First and foremost, modifying and customizing the code of the FHIR server requires access to its source code, which is not always a possibility for commercial off-the-shelf solutions.

Even if access to the source code is available, modifying existing code is generally more costly because new modules have to be tightly woven into the existing FHIR server code and such modifications require intimate knowledge of how the FHIR server code is designed and functions.

The need to update the FHIR server product itself further complicates this since all the customizations need to be applied again every time a new version of the FHIR server is available. Note that while changes to the FHIR standard interfaces and API are less frequent, clearly versioned, and take place in a transparent manner, changes to the internal logic of a FHIR server product are not subject to any of such processes and restrictions and an abrupt or radical changes or refactoring could severely affect the utility and portability of the legacy custom code when new versions of the FHIR server code is available.

Aside from tackling the above issues, implementing security and privacy services outside of the FHIR server codebase also enables the opportunity for independent development, building, deploying, scaling, and iterative updates for these services, potentially by independent, smaller, and more flexible teams, since the scope of each module is limited to the functions and the requirements of one well-defined service.

In summary, FHIR subscriptions provide a crucial building block in developing services such as the SLS and PPS, as an external service, with only minimal and standard coupling to the FHIR server, and particularly untethered from the non-standard implementation details of a FHIR server.

6 ACRONYMS

ACS	Access Control System
API	Application Programming Interface
DSUB	Document Metadata Subscription
EHR	Electronic Health Records
FHIR	Fast Healthcare Interoperability Resources
HL7	Health Level Seven
HTTP	HyperText Transfer Protocol
IHE	Integrating the Healthcare Enterprise
JSON	JavaScript Object Notation
OASIS	Organization for the Advancement of Structured Information Standards
PPS	Privacy-Preserving Service
REST	Representational State Transfer
SLS	Security Labeling Service
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
TLS	Transport Layer Security
URL	Universal Resource Locator
XDS	Cross-Enterprise Document Sharing
XML	Extensible Markup Language

7 REFERENCES

[1] HL7 FHIR v4.0.1, Messaging using FHIR Resources. https://www.hl7.org/fhir/messaging.html

[2] Zapier, REST Hooks, August 27, 2013. https://resthooks.org/docs

[3] IHE IT Infrastructure Technical Framework Supplement, Document Metadata Subscription (DSUB), August 31, 2015. https://ihe.net/uploadedFiles/Documents/ITI/IHE_ITI_Suppl_DSUB.pdf

[4] OASIS, Web Services Base Notification 1.3 (WS-BaseNotification), OASIS Standard, 1 October 2006. http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf

[5] The OAuth 2.0 Authorization Framework, Section 1.5.: Refresh Token, October 2012. https://tools.ietf.org/html/rfc6749#section-1.5

[6] OAuth 2.0 Token Revocation, August 2013. https://tools.ietf.org/html/rfc7009

[7] FusionAuth, Revoking JWTs. https://fusionauth.io/learn/expert-advice/tokens/revoking-jwts