A White Paper on

Developer Portals for CMS patient access FHIR APIs

Ryan M. Harrison
Mike Hiner

Amida Technology Solutions, Inc.
December 2021

Summary

The Centers for Medicare and Medicaid Services (CMS) Interoperability and Patient Access Final Rule (CMS-9115-F) requires commercial payers, Medicaid Advantage plans, and state Medicaid agencies to provide a patient access API that will let patients delegate access to their claims and clinical records to third-party applications. This white paper describes the key features of developer portals and emphasizes how they enable third parties to use CMS-mandated APIs "without special effort."

Conceptual Architecture of a CMS patient access API solution

Application developers would use a Medicaid FHIR developer portal to build custom applications for Medicaid members or to integrate Medicaid member data into existing applications. A single portal can serve both first-party and third-party developers; it is not necessary to maintain two distinct portals.[1] In Figure 1, we show the developer portal within the context of the major components in a patient access solution.

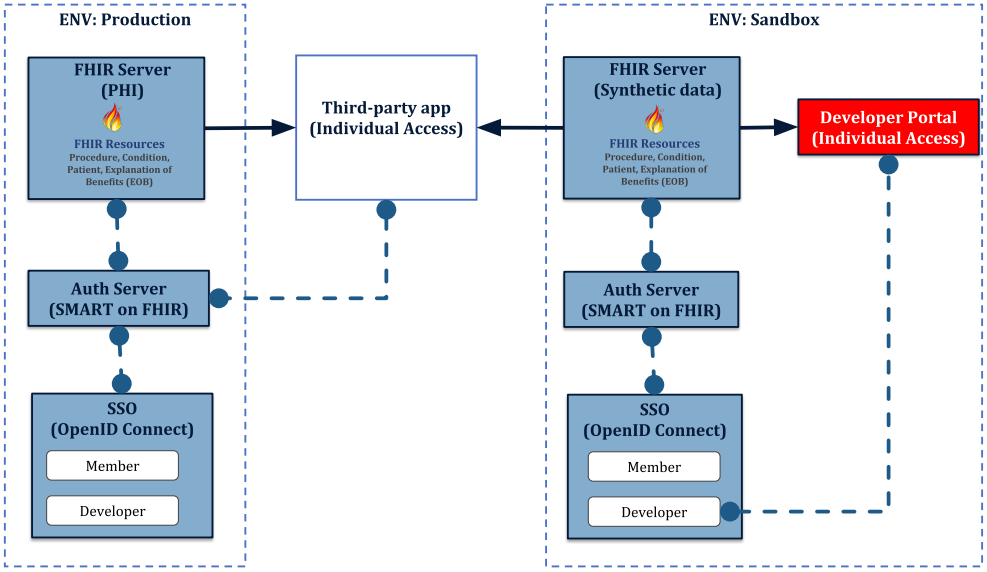

***Figure 1** – Architectural context for the developer portal (highlighted in red).*

[1] Indeed, the author considers the act of "dogfooding" one's own APIs to be a best practice.

Landing Page (API Summary)

Developers start here. The landing page contains a marketing description of the API. Where the developer portal serves multiple APIs, the landing page provides a high-level graphical depiction that summarizes the APIs, with links to more detailed marketing pages about each. Some organizations provide separate audience-specific marketing pages, e.g. "Medicaid Administrators," "Fiscal Agents," and "Community Developers."

Quickstart Guide

A guiding principle for modern developer-portal design is "show, don't tell." The Quickstart guide shows, in code, a minimum viable example of how to use the API's core functionality. Links to sample applications that show more complex functionality are often available on public source-control repositories such as GitHub or GNU Savannah. A short sixty- to ninety-second introduction video can serve as a marketing tool, and it can reduce the number of HelpDesk inquiries for the most frequently asked developer questions.[2]

API Documentation

The API documentation serves as a complete reference for all available endpoints. Common API specification languages, used in the definition of API contracts, include Swagger/OpenAPISpec, RAML, and API Blueprint. API documentation can be generated automatically from these contracts. "Static" documentation only describes API usage, whereas "Live" documentation describes usage *and* allows the developer to make API requests from within the API documentation. "Live" documentation can be handled by either a mock server (which returns data based on the API specification) or a sandbox environment (which returns synthetic data). API release notes would also appear in this section. Technical marketing and update guidance often accompany major releases — for example, an update from FHIR R4 to R5.

Sandbox Access

Whereas some APIs contain public information and do not require developer registration, FHIR APIs provide access to patient health information;[3] therefore, the developer portal must give instructions for application registration. Once registered, the application will be provided credentials that allow the developer to work in a production-like "sandbox" environment populated with sample data. Unlike production access, sandbox access does not require manual review and can be granted automatically.

The benefits of providing sandbox access include: 1) developers build their applications in a SMART on FHIR implementation identical to the production environment, which reduces the risk of access control misconfiguration; 2) the sandbox contains sample data, which lets developers design the application UI using responses similar to those used in the production environment; 3) reduced HelpDesk inquiries, because developers can "self-service" specific usages simply by making requests.

[2] Likewise, a user-facing video that covers how to use third-party apps, what information will be shared, and how to revoke access can reduce HelpDesk volume for common user inquiries.
[3] The individual access APIs mandated for 2021 are read-only. Neither the sandbox nor production environments will support third-party applications writing data to the state Medicaid agency's FHIR implementation.

Production Access

Production access is required for state Medicaid members to delegate access to their information. Unregistered applications cannot retrieve member data. Before allowing production access, patient access APIs should apply a vetting process to ensure that applications meet a minimum standard. Data breaches and other negative PR events will reflect poorly on the API owner, even if a third-party application developer is technically at fault.[4]

In their API Maintenance of Certification criteria, the Office of the National Coordinator for Health Information Technology (ONC) is prescriptive about permitted timelines for application vetting.[5,6] Briefly, a certified API Developer has ten days to complete Authenticity Verification, followed by five days for Application Registration. Vetting requirements should include a privacy policy and a data use policy that explain to the Medicaid member – in plain English – how the member's information will be used, combined, and shared. Naturally, the automation effort applied to vetting should be proportional to the volume of application registrations. At low volume, a wholly manual process is sufficient (e.g., emailing a checklist); at higher volume, semi-automation with a dashboard for application registrants and vetting staff is advisable.

Support

Developer portals often provide links to resources such as source repositories[7] and community forums (e.g., Mastodon communities). Support also includes system status pages to notify the community of outages and production issues, as well as a security contact for reporting security vulnerabilities in the API. Support is the most important fixture of developer community engagement. Responsiveness – including alerts to the developer community about major updates (especially those that break API compatibility) and announcements of security vulnerabilities – builds trust and loyalty over time.[8]

Budget Gotchas

Build vs Buy

Organizations must budget for both the upfront and the ongoing costs of a developer portal. The most important decision is whether to buy and configure a vendor product (e.g., 3Scale, Apigee, Kong, MuleSoft, Tibco, etc.)[9] or build a custom portal.[10]

---

[4] Facebook received widespread criticism for data misuse as a result of the Cambridge Analytica scandal, despite the fact that the bulk of the misuse was committed by Cambridge Analytica (enabled by Facebook's API design).
[5] Summarized in "21st Century Cures Act: Interoperability, Information Blocking, and the ONC Health IT Certification Program Final Rule" [https://www.healthit.gov/sites/default/files/facas/2020-03-18_ONC_21st_Century_Cures_Act_Final_Rule_Presentation_1.pdf]
[6] Although distinct, the ONC (84 FR 7424) and CMS (84 FR 7610) rules are designed to work in concert. The CMS rules "pass through" API standards to ONC; satisfying the ONC API technical criteria for certification should meet the CMS rule.
[7] One advantage of publicly available software backlogs is that developers can comment directly on the issues that impact them. In other words, the link between support and development is made transparent.
[8] The role of Developer Advocate has emerged to serve this need.
[9] At the Enterprise-tier, the developer portal is offered as a component of a complete API Management solution, encompassing API design, API Gateway (e.g., rate-limiting), and non-technical administration (e.g., a non-technical user using the vendor GUI to recompose a set of individual API end-points into an API Product with a monetization strategy). These features, often branded as "API life-cycle management," are most valuable when administering collections of APIs, as opposed to a single API.
[10] Both vendor products and custom builds often utilize a content management system, like Drupal, as the technical backbone of the developer portal. For smaller projects, developer portals can be built on GitHub Pages.

Technical Writers

Technical writing is frequently overlooked as an upfront and ongoing cost. An API contract – for example, a Swagger specification – written by engineers and suitable for communication between first-party development teams, will *not* be sufficient as API documentation for third parties. As the API evolves over time, code changes must be accounted for in the documentation. The need for technical writing waxes and wanes, with peaks around major release dates, but it must be considered until deprecation or end-of-life.

Technical writing is not just a cost center; it can contribute to top-line revenue for monetized APIs. Technical writers can increase API adoption by creating inbound marketing collateral — for example, blog posts that introduce important API features. These materials can highlight the differences between community tiers and paid tiers, and they are invaluable technical marketing collateral.

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 in 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.