Provider Access Connectivity Hub

Companion Guide

A guide for payers implementing Availity Provider Access APIs to meet CMS-0057 regulatory requirements.

Disclosure Statement

Availity provides the information in this document for educational and informational purposes only. While Availity believes the information is accurate at the time of publication, this document does not provide legal, financial, or professional advice. For guidance specific to your organization, consult a qualified attorney or advisor.

References to third-party organizations, products, tools, or websites do not imply endorsement by Availity. Availity is not responsible for the content, availability, or accuracy of third-party resources.

Contents

Introduction

The Provider Access API, as defined in CMS-0057, requires CMS-regulated payers to make certain clinical data available to providers for members with whom they have an active treatment relationship. This mandate becomes effective on January 1, 2027.

Provider Access enables providers to retrieve relevant member data directly from payers by using standardized **FHIR-based APIs**. This approach reduces reliance on manual processes, such as faxed records or proprietary data feeds. Providers initiate requests, and payers are responsible for validating requests, enforcing member opt-out requirements, and returning the appropriate data.

This guide is intended for payer customers implementing Availity's Provider Access Connectivity Hub. It provides a high-level overview of the solution, payer responsibilities, and the steps required to establish connectivity and to begin testing.

This document follows the terminology used in the HL7 Payer Data Exchange (PDex) Implementation Guide:

- **Payer**: A healthcare plan queried for patient-provider relationships and related clinical data.
- **Provider**: A healthcare practitioner entitled under CMS-0057 Provider Access requirements to request clinical data for members with whom they have an active treatment relationship.
- **Member (or patient)**: An individual whose data a provider requests from a payer.

The problem: provider access data exchange

Without a connectivity-hub model, supporting the Provider Access API introduces operational challenges for payers.

1. **Connectivity scale**

In a direct point-to-point model, a payer must:

- Establish and maintain integrations with a large and growing number of provider systems and EHR vendors.
- Monitor and update endpoint configurations over time.
- Respond consistently to requests arriving from heterogeneous implementations.

This approach does not scale efficiently as provider participation increases.

2. **Provider identity validation**

In addition to managing multiple integrations, payers must:

- Validate the identity of each requesting provider.
- Establish trust relationships and credentialing mechanisms.
- Manage provider onboarding, credential lifecycle, and offboarding.

This identity and trust burden increases with provider participation and can become a significant operational and security challenge.

How Availity solves the problem

Availity's Provider Access Connectivity Hub addresses these challenges through a network-based, hub-and-spoke architecture.

Single Connectivity Layer

Payers establish a **single integration** with Availity instead of maintaining direct connections with each provider or EHR system. Provider Access requests route through the hub, significantly reducing the number of interfaces payers must support.

Established Provider Identity Vetting

Availity maintains a standardized process for:

- Vetting provider identities as part of network participation and issuing digital credentials to validated providers.

As a result, payers receive Provider Access requests from a trusted, authenticated network without independently managing provider credentialing.

How does Availity solve for PDex?

Establishing, testing, and maintaining interfaces with multiple payers in a many-to-many architecture is complex and costly. Availity eliminates this burden by serving as a centralized intermediary.

The Availity Provider Access Connectivity Hub

Availity leverages existing relationships with participating payers to facilitate patient-data exchange through a hub-and-spoke model. Instead of connecting directly with every payer, a provider submits a standardized set of FHIR API calls to Availity. Availity routes requests to the appropriate payer, retrieves the requested data, and securely returns it to the requester.

NOTE	Provider Access is an opt-out use case. Payers are responsible for maintaining member opt-out lists and enforcing opt-out requirements as part of the Provider Access workflow.

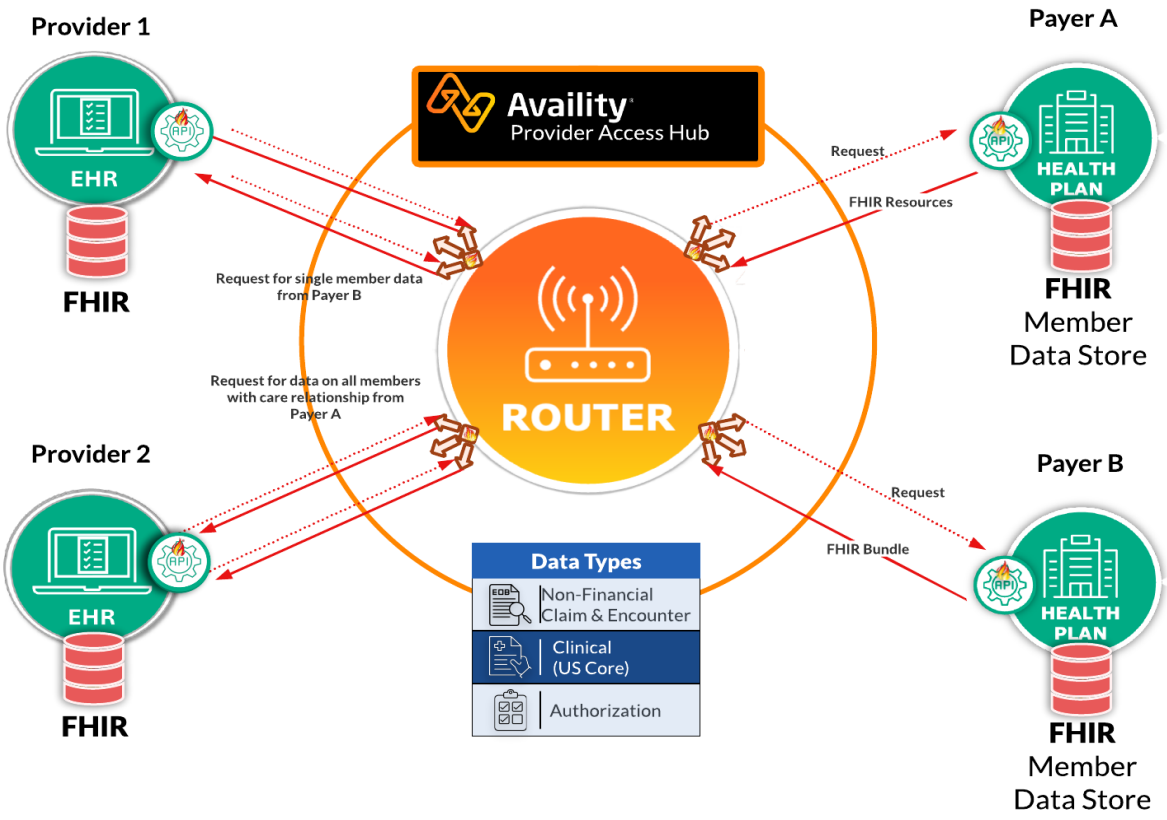


Figure 1 – High Level Overview of Data Exchange Process

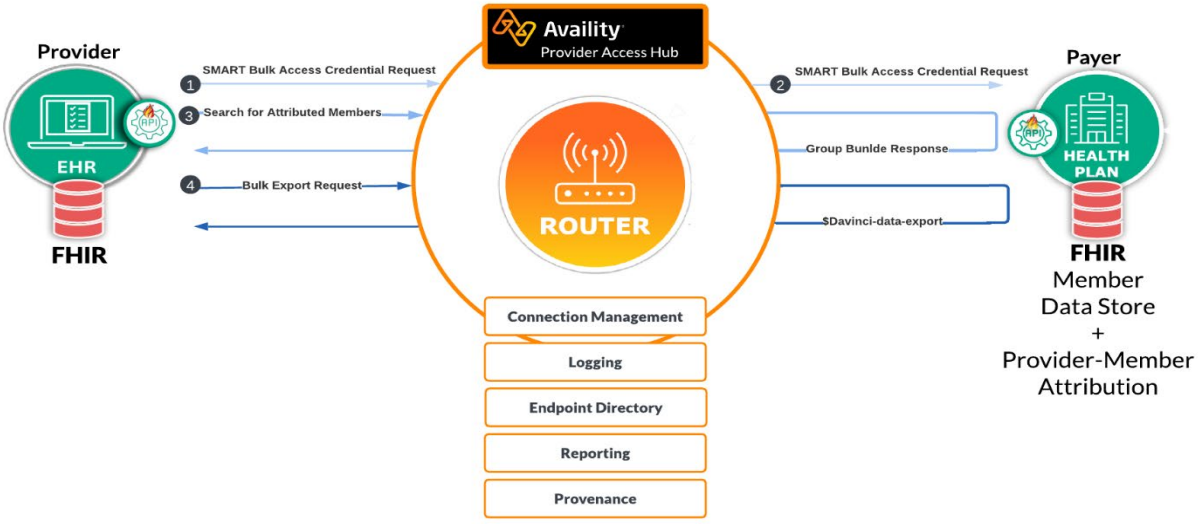


Figure 2 – High Level Request Order

Overview

Exchange modalities

Bulk (multiple patients)

At a high level, Provider Access requests follow this sequence. This guide presents the flow conceptually. Detailed API walkthroughs are available in separate developer documentation.

1. **Authorization**
   - A provider authenticates by using Availity-issued digital credentials.
   - Availity validates the credentials and proxies the authenticated request.
2. **Group search**
   - The provider initiates a group-based Provider Access request.
   - Availity routes the request to the appropriate payer based on request identifiers
3. **$davinci-data-export**
   - The provider requests a bulk export of data for the matched group.
   - Availity proxies the request to the payer.
4. **Status endpoint**
   - The provider checks the status of the bulk export through Availity.
   - Availity monitors and proxies status responses.
5. **Final download links**
   - After the export completes, the payer generates secure download URLs.
   - Availity provides proxied download links to the provider.

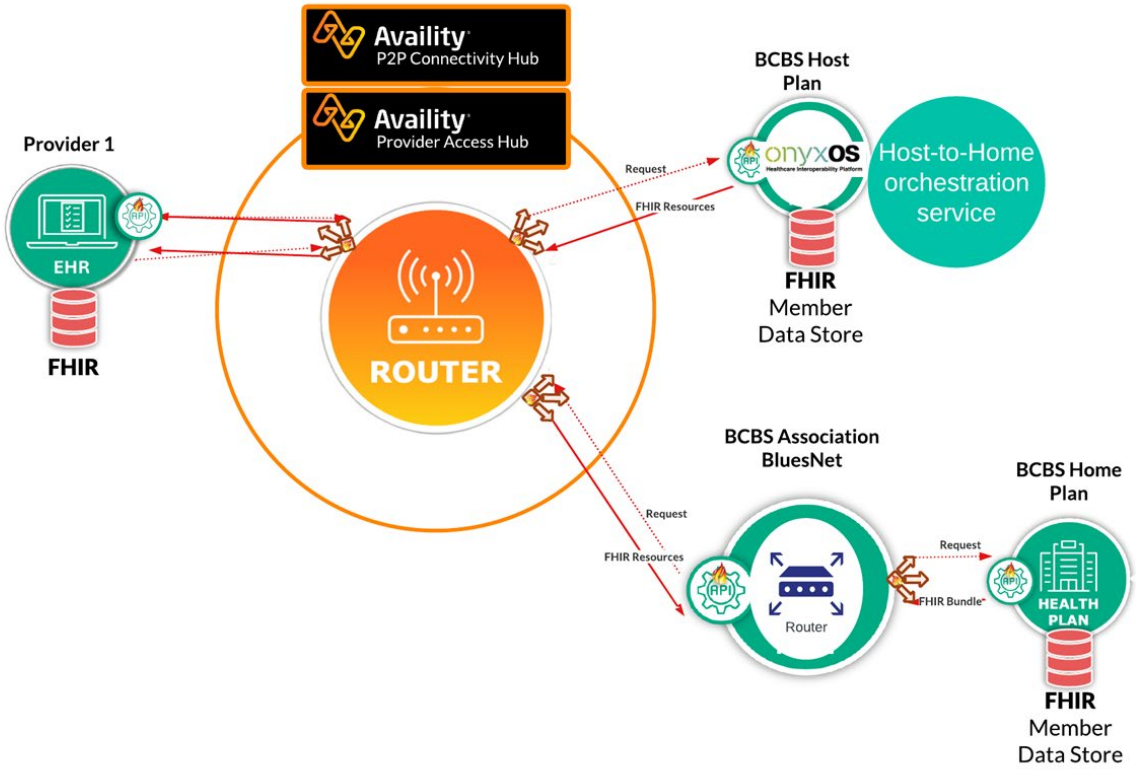
Throughout this process:

- Providers never connect directly to payer endpoints.
- Availity does not modify payer data.
- Payers remain responsible for opt-out enforcement, attribution validation, and data accuracy.

Blue Cross Blue Shield Association overview

The following sequence illustrates a Provider Access request routed through the Availity hub:

1. The provider sends an asynchronous PDex Provider Access request to Availity.
2. Availity routes the request to Onyx.
3. Onyx identifies hosted provider relationships and sends synchronous requests to the Association or Home Plan through Availity.
4. The Home Plan returns responses through the Association and Availity.
5. Onyx aggregates the data and returns the appropriate results to the provider through Availity.



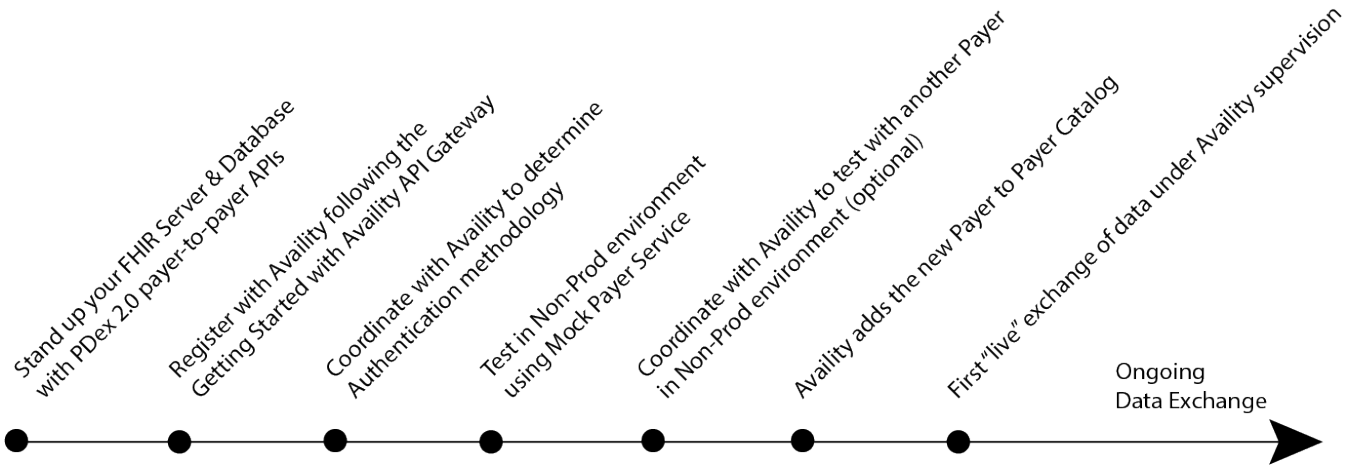
Implementation plan

Implementation steps vary depending on payer capabilities. In general, onboarding follows this sequence.

1. Onboarding objectives:
   - Determine whether the payer is onboarding only to the Provider Access Connectivity Hub or to both the hub and the Onyx solution.
   - Confirm whether the payer must connect to the Blue Cross Blue Shield Association Interoperability Hub (Blue plans only).

- Ensure the payer meets prerequisites, including FHIR and PDex alignment.
- Establish the authentication method (for example, OAuth 2.0 with a client ID and client secret).
- Implement a FHIR R4+ server aligned with PDex v2.0 or later.
- Register with the Availity API Gateway in the non-production environment, production environment, or both.
- Test in the non-production environment by using the Mock Payer Service (recommended).
- Complete supervised production readiness testing.
- Perform the first live data exchange under Availity supervision

2. Payer implements a FHIR 4.0+ server/database with PDex v2.0 or higher payer-to-payer API(s).
3. Payer follows the “Getting started with Availity API Gateway” steps to register with Availity (non-production environment, production environment, or both)
4. Authentication methodology is determined
5. Payer tests in the non-production environment with the Mock Payer Service (optional but recommended)
6. Payer works with Availity to test with another cooperating payer in a non-production environment (optional).
7. Availity’s Payer Catalog is updated with the onboarding payer’s information.
8. Payer makes first “live” exchange of data under Availity supervision.



Provider Access Connectivity Hub features

The table below highlights the key features of Availity’s Provider Access Connectivity Hub.

Feature	Description
Authentication, trust, and network access	Secure connectivity with trusted healthcare payers and providers across the Availity network
Endpoint catalog	An up-to-date, API-accessible list of participating payers, plans, and identifiers
Dynamic request routing	Routes requests to participating payers by using catalog identifiers
Asynchronous data retrieval	Supports bulk member requests with secure token and URL mapping
Mock Payer Service	Enables testing with synthetic data to avoid PHI and PII exposure
Operational monitoring	Tracks service availability and uptime
Payload validation	Identifies validation issues and trends
Provenance	Adds FHIR Provenance resources identifying source payer and Availity as intermediary
Documentation and enablement	Provides guides, portals, training, and consultations
Onboarding and support	Serves as the primary onboarding and support contact

Attribution and patient matching

Payers must maintain **member-to-provider attribution data**. In PDex v2.0–v2.1, webhooks enable providers to submit attestations for patient-provider relationships that may not already exist in payer systems. Future PDex versions may support provider-submitted bulk member-match requests.

Matching criteria

Each payer defines its own matching logic. At a minimum, the following values should be included in a member-match request:

- Member ID
- Last name
- First name
- Date of birth
- Phone number
- Address (full address or postal code)

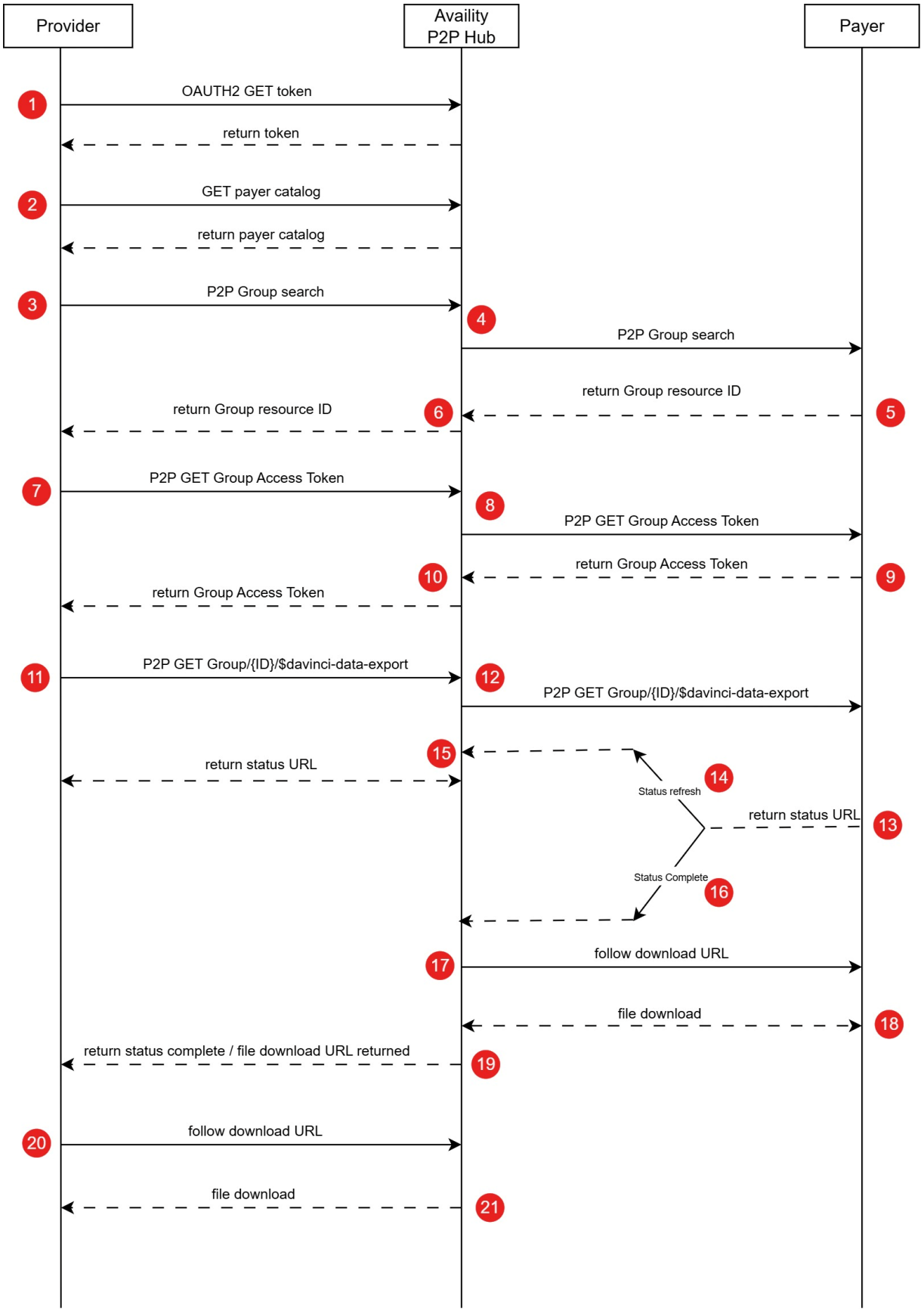
Technical overview – $multi-member-match request

Availity establishes connectivity with payers in advance and manages authentication exchanges on their behalf. The Provider Access Connectivity Hub acts as a proxy between parties, preventing direct system-to-system access and masking internal implementation details, such as FHIR server URLs.

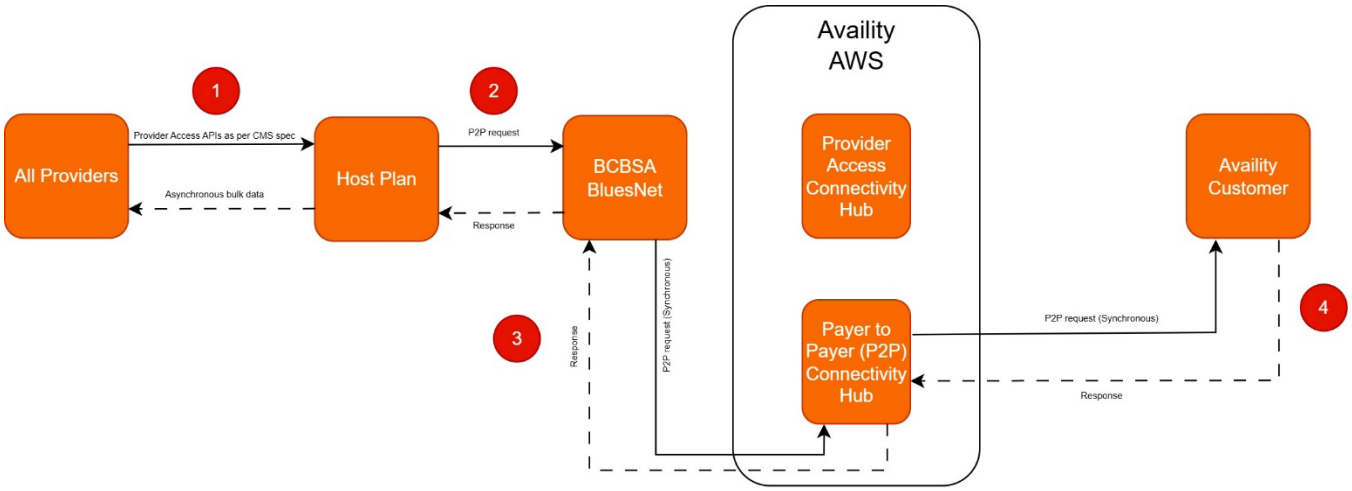
This section explains the request and response flow for the **$multi-member-match** workflow and describes each step in the exchange sequence.

End-to-end request flow

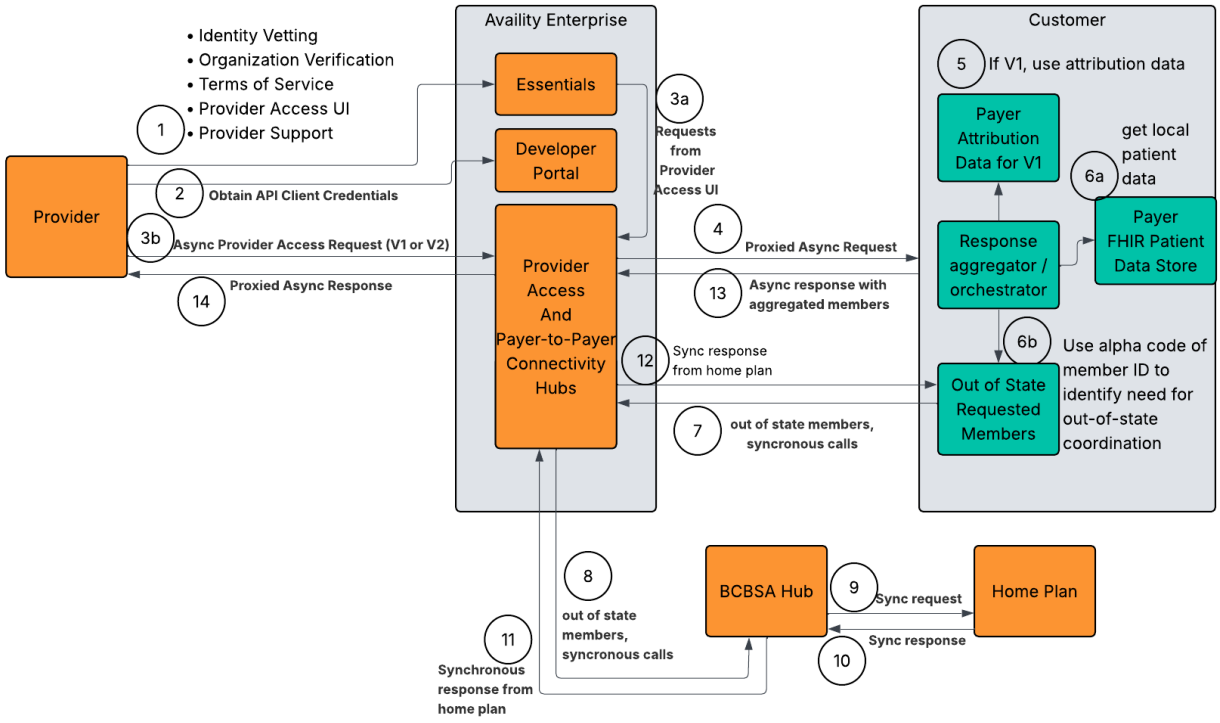
1. The provider uses the credentials provided during onboarding to authenticate with Availity's Provider Access Connectivity Hub and retrieve an access token.
2. The provider requests the most recent Payer Catalog to identify the correct payer and plan. Availity returns the catalog.
3. The provider submits a group search request to Availity, identifying a list of patients linked to a single payer.
4. Availity authenticates with the defined payer and forwards the request.
5. The payer matches members and returns a Group ID.
6. Availity proxies the URLs and Group ID and forwards the response to the provider.
7. The provider requests a scoped token to access data for the group.
8. Availity proxies the token request to the defined provider.
9. The payer returns a Group Access Token (also called a Group Scoped Token).
10. Availity proxies the token and URLs and forwards them to the provider.
11. The provider uses the scoped Group Access Token to submit a GET $davinci-data-export request for all data associated with the group's members.
12. Availity proxies the URLs and routes the request to the payer.
13. The payer returns a status URL that Availity uses to monitor the progress of bulk export file preparation.
14. Availity proxies the URLs and routes the $davinci-data-export response to the provider and continues polling the status endpoint at regular intervals.
15. Availity returns a status URL to the provider who checks that status endpoint until complete.
16. The payer polls the endpoint and detects the **Complete** status.
17. Availity receives and follows the download URL(s).
18. Availity downloads the file(s).
19. Availity updates the provider status to **Complete** with a download URL.
20. Provider follows the download URL.
21. Provider downloads the file(s).


Provider
Availity P2P Hub
Payer
1
OAUTH2 GET token
return token
2
GET payer catalog
return payer catalog
3
P2P Group search
4
P2P Group search
return Group resource ID
5
return Group resource ID
6
7
P2P GET Group Access Token
8
P2P GET Group Access Token
return Group Access Token
9
10
return Group Access Token
11
P2P GET Group/{ID}/$davinci-data-export
12
P2P GET Group/{ID}/$davinci-data-export
15
return status URL
Status refresh
14
return status URL
13
Status Complete
16
17
follow download URL
file download
18
return status complete / file download URL returned
19
20
follow download URL
file download
21

Provider sending a request to an Availity BCBS customer.



Provider sending a request to an external BCBS plan with the Availity BCBS customer identified as the home plan.



Get started with the Availity API Gateway

The Availity API Gateway provides a unified point of access to Availity's externally facing APIs. This section is intended for developers who are new to the API Gateway and need to complete initial registration, application setup, and subscription tasks.

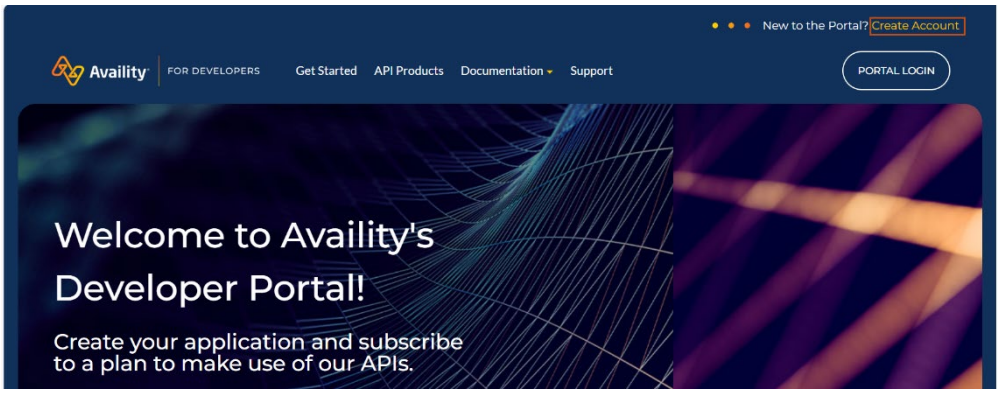
NOTE	Some screenshots in this section are intentionally shortened or truncated to improve readability and to avoid unnecessary repetition. Nothing that requires your attention or interaction is omitted. If an element is not shown in a screenshot, It may be safely disregarded.

Create an account

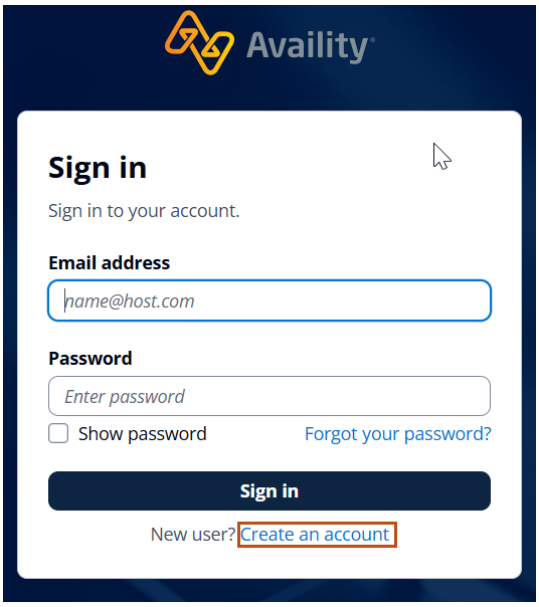
To access the API Gateway, you must register with the Availity Developer Portal.

1. Open a web browser and go to one of the following URLs:
   - **QA environment:** https://qua.developer.availity.com/
   - **Production environment:** https://developer.availity.com/

2. Select **Create Account**.



   - In the **Sign in** dialog box, select **Create an account**.



- Enter the required information:
    - **Email address**
    - **Given name**
    - **Family name**
    - **Password**
    - **Confirm password**
- Select **Sign up**.

Set up multi-factor authentication

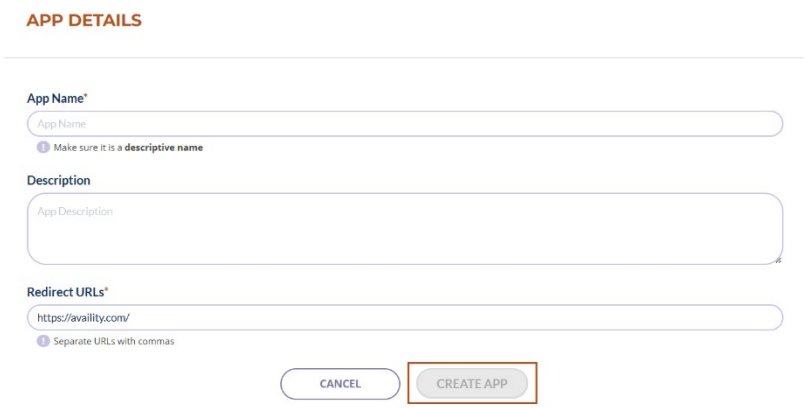
After you create your account, complete multi-factor authentication (MFA) setup.

1. Install an authenticator app on your device.
2. Register your email address in the authenticator app.
3. Scan the QR code displayed on the screen.
4. In the authenticator app, generate a verification code.
5. Enter the verification code on the Multi-factor authentication (MFA) screen.
6. Select **Sign in** to complete setup and access the portal.

Create an app

After signing in, create an application to associate with your API subscriptions.

1. Sign in to the Availity Developer Portal.
2. From the navigation menu, select **My Apps**.
3. Select **+CREATE A NEW APP.**
4. Enter the following information:
   - **App Name**
   - **Description**
   - **Redirect URLs**
5. Select **CREATE APP.**

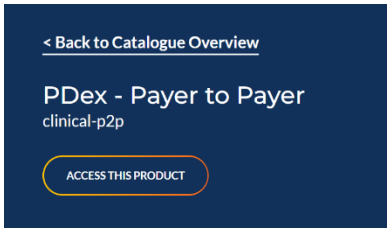


Subscribe to the PDex API

To use the Payer Data Exchange (PDex) APIs, you must request a subscription.

1. Sign in to the Availity Developer Portal.
2. From the navigation menu, select **API products**.
3. Search for PDex. The following products will display:
   - **PDex – Payer to Payer**
   - **PDex – Payer to Payer - Payer Catalog**
   - **PDex - Payer to Payer Bulk**
4. Select **More info** for the desired API product.
5. Select **ACCESS THIS PRODUCT** to view the available subscription plans.

Note: You must submit a subscription request and receive approval before you can use the API product.



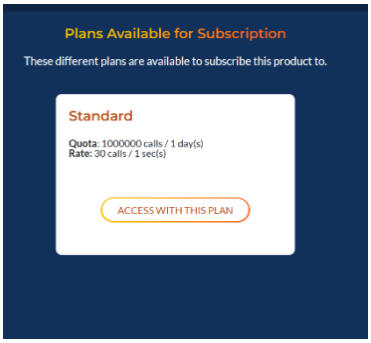
Subscription plan details:

- **Demo Plan**
  Provides a sandbox environment for testing communication with Availity APIs. Demo subscriptions are automatically approved and are recommended for learning the Availity REST API workflow.
- **Standard Plan**
  Provides full access to Availity REST APIs and production data.

Submit a subscription request

After selecting a subscription plan, submit your request.

1. From the subscription notification, select **ACCESS WITH THIS PLAN**.



2. Select **Go to cart** to complete the subscription request.

   Note: Alternatively, you can select the cart icon next to your username.

3. Verify that the correct API product displays in the **Selected Products** section.
4. Verify that the correct subscription plan displays in the **Selected Plan** section.
5. In the **Select an App** section, select either of the following:

- **Create a New App** and then enter the application details, or
- **Existing App** and then select an existing application.

6. Select **SUBMIT REQUEST**.
7. The Trading Partner Management team assists with contracting and activation.
8. After contract verification, Availity approves the subscription. You can then view access and credential information on the ***My Apps > Approved Access*** tab.

Test with Availity's Mock Payer Service

To facilitate initial testing of the Provider Access Connectivity Hub, Availity offers a Mock Payer Service. The Mock Payer Service is intended to validate that providers can connect to the Availity Provider Access Connectivity Hub and can make all the necessary API calls in an interoperable manner.

Providers should plan to test using the Mock Payer Service initially and then plan to conduct system-to-system testing with a cooperative peer in the non-production environment, before moving to a live Production environment. Availity can assist in arranging peer-to-peer testing with an appropriate and willing partner to the benefit of each party.

Scheduling time for testing with Availity in a non-production environment is important for all parties. Such cooperative testing ensures that our partners have an appropriate understanding of the exchange and that all your implementation questions are answered before attempting to go live in the production environment.

The Mock Payer Service is only available in a non-production environment. Mock Service Payer IDs can be discovered through the Payer Catalog; the synthetic payers used for testing all have names that end in "-mock." The Mock Payer Service does not proxy calls to a real payer. Instead, it returns responses with synthetic data. You can direct API calls to any *-mock payer ID, and the results will be identical.

An API collection is available to help with testing. It accelerates connectivity verification, discovery, and evaluation of platform functionality and performance. New customers can access the API collection through the Availity Clinical Solutions online support hub during onboarding.

Limitations of the Mock Payer Service

The Mock Payer Service is designed to help users become familiar with the Provider Access Connectivity Hub APIs and validate authentication and request flows. It is not intended to represent a full end-to-end PDex implementation.

Be aware of the following limitations when testing with the Mock Payer Service:

- Regardless of which -mock payer ID you specify in API calls, all responses originate from the mock server and are identical.
- The sample patient ID provided for testing can be used with any -mock payer ID.
- The Mock Payer Service does not evaluate demographic or biometric fields included in the member-match request. Only the provided patient ID is evaluated.
- The Mock Payer Service verifies that a Consent resource is present in the request and rejects requests that do not include a required Consent section, as defined in the applicable implementation guides.
- The Mock Payer Service does not evaluate the content, terms, or conditions of the Consent resource and does not alter its response based on consent details.

Important: Although the Mock Payer Service does not enforce consent rules, consent (opt out) **must** be fully respected and enforced by the payer in the production environment.

Use the Mock Payer Service only to validate connectivity, authentication, and API sequencing. Do not rely on mock responses to validate business logic, matching accuracy, or consent enforcement behavior.

The Payer Catalog

1. Availity's Provider Access Connectivity Hub enables providers to request clinical data according to the HL7 Payer Data Exchange (PDex) standards by using FHIR API calls.
2. In many cases, the individual requesting their data can provide information about their previous insurer and plan. However, this information may not always be available. In these situations, the **Payer Catalog** enables Providers to identify the appropriate payer and plan required to route the data request.

Purpose of the Payer Catalog

1. The Payer Catalog supports provider access workflows by providing:
   - A complete list of payers participating in the Provider Access Connectivity Hub
   - Plan-level details for payers that support multiple plans, and
   - Unique identifiers required for routing API requests
2. The Payer Catalog ensures that requests are routed accurately and consistently, even when member-provided information is incomplete or unavailable.

Payer Catalog availability

1. Availity makes the Payer Catalog available through a RESTful API. The Payer Catalog:
   - Is maintained and is updated regularly as network participation changes
   - Is available on demand, and
   - Is returned in a JSON format that is easy to consume programmatically
2. Because the Payer Catalog is accessible in near real time, it always reflects the most current set of participating payers and plans.

Common use cases

1. Providers can use the Payer Catalog:
   - To identify the correct **payer ID** and **plan ID** for routing Provider Access requests
   - To populate internal workflows that require payer and plan selection

Payer Catalog content

1. Each payer entry in the catalog includes:
   - **payerId**
   - **payerName**
   - **payerDescription**
2. If a payer supports multiple plans, the catalog also includes a **plans** array containing:
   - **planId**
   - **planName**
   - **planDescription**
3. These values are required inputs for subsequent API calls made through the Provider Access Connectivity Hub.

Authenticate to obtain an Access Token

Before you can call Provider Access APIs, you must authenticate with Availity's API Gateway and obtain an access token. The access token authorizes subsequent API requests, including requests to the Payer Catalog.

To obtain an access token, use your client ID and client secret with the following API call:

POST {root_URL}/v1/token

The response includes an **access_token** for use in the GET /payer-catalog call:

```
{
"token_type": "Bearer",
"access_token": "XXXXXXXX-EXAMPLE-XXXXXXXXX",
"scope": "hipaa",
"expires_in": 300,
"consented_on": 1728330235
}
```

Request the Payer Catalog

Place the **access_token** value into the **Bearer token** field of your GET /payer-catalog call, and send that request to the API gateway:

GET {root_URL}/fhir/r4/PayerCatalog

The response will look like the example below : a block of JSON-formatted data containing an array of payers, and if a payer offers multiple plans, a sub-array of those plans. In the example below, there are three sample payers represented: Centene, Blue Cross Blue Shield, and Humana.

NOTE	Observe that Centene does not offer multiple plans, so its **plans[]** array is empty. By contrast, both Blue Cross Blue Shield and Humana each offer three different plans, so their plans[] arrays are populated.

```
{
    "payers": [
        {
            "payerId": "centene-mock",
            "payerName": "Centene",
            "payerDescription": "Test Payer Centene",
            "plans": []
        },
        {
            "payerId": "bcbsfl",
            "payerName": "Blue Cross Blue Shield",
            "payerDescription": "Test Payer BCBS",
            "plans": [
                {
                    "planId": "peachstate12345",
                    "planName": "Blue Cross Blue Shield GA",
                    "planDescription": "Plan A Desc"
                },
                {
```

```
                    "planId": "testPlanIDB",
                    "planName": "BCBS Test B",
                    "planDescription": "Plan B Desc"
                },
                {
                    "planId": "testPlanIDC",
                    "planName": "BCBS Test C",
                    "planDescription": "Plan C Desc"
                }
            ]
        },
        {
            "payerId": "humana-mock",
            "payerName": "Humana",
            "payerDescription": "Test Payer Humana",
            "plans": [
                {
                    "planId": "humanaPlanIDA",
                    "planName": "Humana Test A",
                    "planDescription": "Humana Plan A Desc"
                },
                {
                    "planId": "humanaPlanIDB",
                    "planName": "Humana Test B",
                    "planDescription": "Humana Plan B Desc"
                },
                {
                    "planId": "humanaPlanIDC",
                    "planName": "Humana Test C",
                    "planDescription": "Humana Plan C Desc"
                }
            ]
        }
    ]
}
```


Key Information

From these Payer Catalog results, it is easy to locate the payer and if needed, the specific plan, as well as the associated **payerID** and **planId** values for use in subsequent API calls to the Provider Access Connectivity Hub for retrieving a patient's data from their payer.

Provider Access Connectivity Hub Version 1 – Bulk API Call Sequence

The HL7 PDex specification describes two workflows for providers requesting data from payers. In **version 1**, the provider relies on the payer to maintain an attribution list that defines provider–patient relationships. The provider can access data only for patients with whom the payer has identified an existing relationship. In **version 2**, the provider initiates the request by submitting a member-match query for specific patients and includes an attestation confirming a treatment relationship.

Availity supports both Provider Access v1 and Provider Access v2. Provider Access v2 is currently in the HL7 balloting process. After HL7 formally publishes the specification, this companion guide will be updated to include v2 workflows.

Overview

Retrieving bulk patient data from the payer requires a sequence of API requests:

1. POST /token to obtain an access token.
2. GET /PayerCatalog to identify the payer ID for routing the patient data request to the correct party.
3. POST /fhir/r4/Group/ to search for attributed members.
4. GET /fhir/r4/Group/{id}/token to obtain a token scoped to the group of members.
5. GET /fhir/r4/Group/{id}/$davinci-data-export to initiate export of member data in NDJSON format.
6. GET /fhir/r4/Group/$davinci-data-export/{id}/status to check the status of the bulk export request.

Each step is described in more detail below with examples shown in the Insomnia API tool. If you are using Postman rather than Insomnia, there will be some minor variations, such as screen layout, configuring environments, and the naming convention for variables, but the general process and sequence of API calls will remain the same.

NOTE 1	The examples provided here assume that you are using the Insomnia API collection that Availity provides. This collection includes pre-configured API calls for various mock patients, pre-populated with the URL for our API gateway (https://qua.api.availity.com/). Nonetheless, please pay attention to the URL path in the screenshots, as the first API call includes "v1/" after the URL, and the remaining three calls include "fhir/r4/" after the URL. We will define a variable for **baseUrl** to make testing a little easier.

NOTE 2	For security, Availity's API gateway limits the duration of authentication tokens to 300 seconds (5 minutes). You will need to complete the entire sequence of calls within this timeframe before your access token expires.

1. POST /token

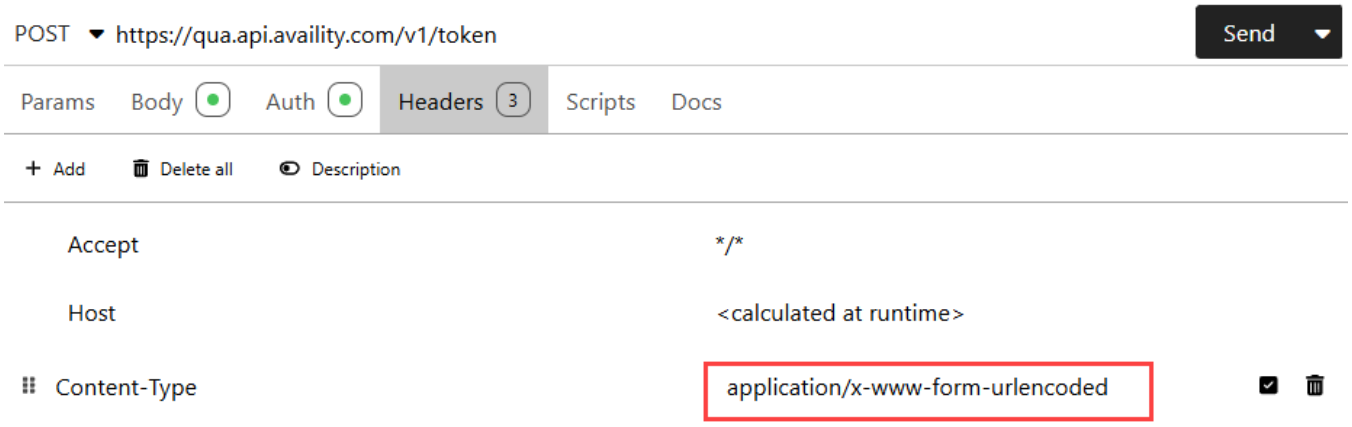
This step authenticates the Provider Access Connectivity Hub and retrieves the authorization token required for subsequent requests. Before you send this request:

1. Note that the URL for this request is `https://qua.api.availity.com/v1`. All subsequent requests use the same URL defined in the **baseUrl** variable.
2. Set the body type to **Form**.
3. Enter the form data values for **ClientID** and **client_secret.**

Set the Body Type to **Form**

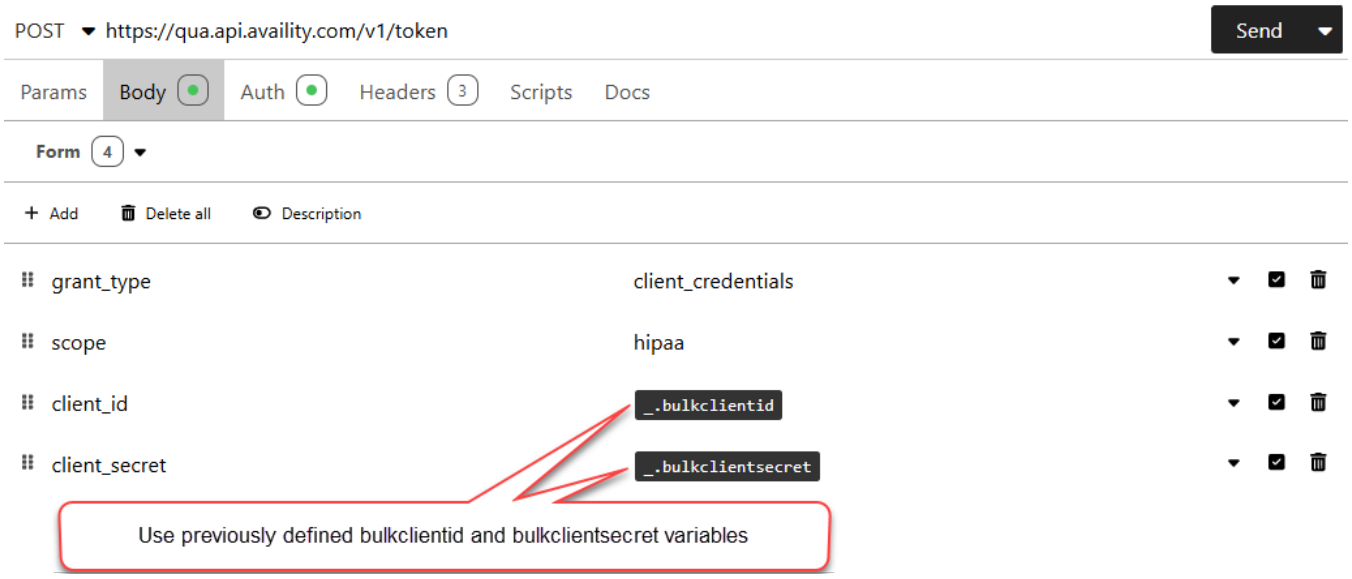
NOTE	This is already done for you in the P2P API collection that Availity makes available, but it is worth verifying if only to familiarize yourself with the different tabs and components of an API call as represented in Insomnia. Postman's UI is similar.

The POST /token call uses a form data Body, while the rest of the API calls have JSON payloads in their Body. In an API tool such as Postman or Insomnia, when you change the body type pulldown from **JSON** to **Form**, the Headers tab will be populated with a **Content-Type** header having a value of **application/x-www-form-urlencoded**, as shown below:



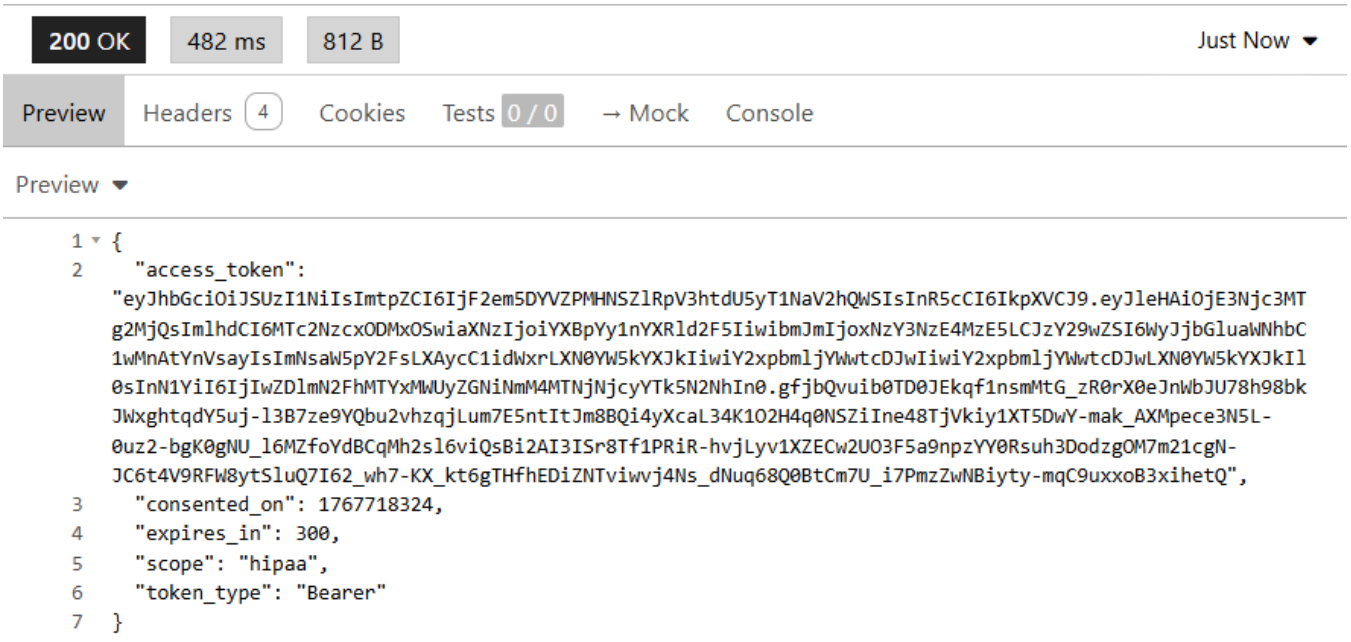
Populate the client ID and client secret

In the **Body > Form** tab, populate **client_id** and **client_secret** with the unique values Availity provided to you. If you are using variables as suggested, populate the form with the variable names as shown below, and then select **Send**.



Response to the POST /token call

The response to this call yields an **access_token** for use with the remaining API calls (line 3 in the screenshot below). The text within the quotation marks is the token value.



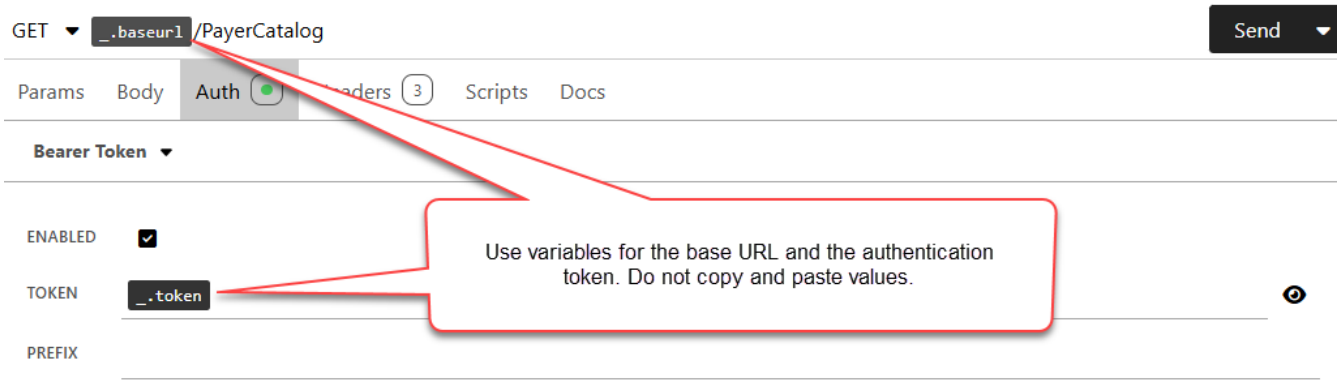
In the remaining API calls you will place the **access_token** into the header value named Authorization in the format `Bearer <access_token>`. But, if you have set up a variable for access_token, you can avoid copying/pasting this value repeatedly by populating the value for Bearer Token as "**_.token**", to take its value from the variable previously defined.

2. GET /PayerCatalog

Availity provides a Payer Catalog to help identify details for the plan. The Payer Catalog includes values for **payerId**, **payerName**, and **payerDescription**. If a payer offers multiple plans, the Payer Catalog also lists a **planId**, **planName**, and **planDescription** for each plan.

This information is required for subsequent requests of the Provider Access Connectivity Hub. To retrieve a patient's data from their plan, you must know both the payer to populate the **CoverageToMatch** section of the $member-match request, and the **PatientID** so the **Provider Access Connectivity Hub** can request the correct patient's data.

After you obtain the **access_token**, send a **GET** /PayerCatalog request, as shown below, and include the **Token** parameter.



The response is a JSON-formatted array that lists all payers participating in the Provider Access Connectivity Hub and the plans each offers, as shown in the following example:

200 OK | 423 ms | 1768 B | Just Now

Preview | Headers 13 | Cookies | Tests 0 / 0 | → Mock | Console

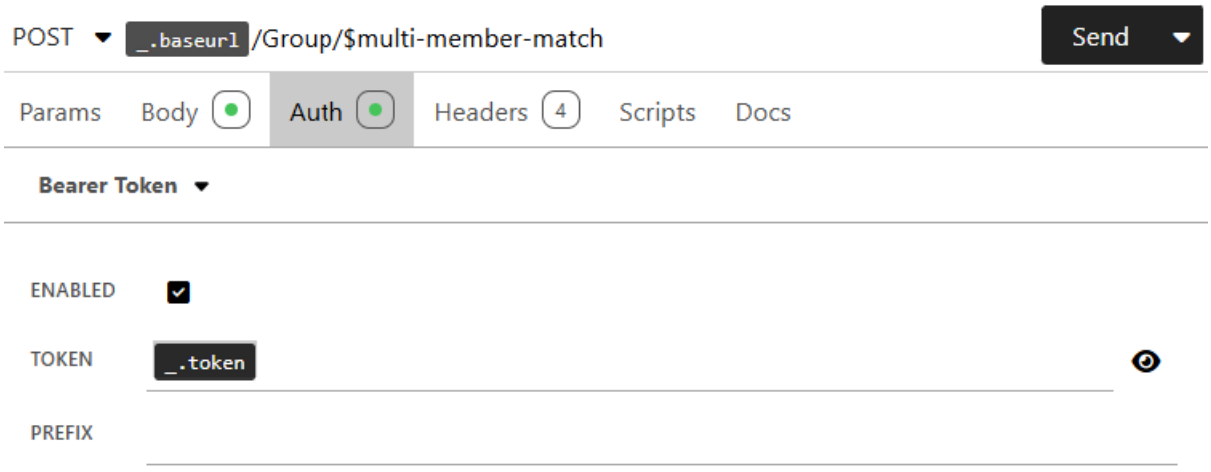
Preview


```
{
  "payers": [
    {
      "payerId": "hcsc",
      "payerName": "HCSC",
      "payerDescription": "HCSC Description",
      "isPayerCustomer": null,
      "isPayerConnected": null,
      "plans": []
    },
    {
      "payerId": "floridablue",
      "payerName": "Florida Blue",
      "payerDescription": "Florida Blue Description",
      "isPayerCustomer": null,
      "isPayerConnected": null,
      "plans": []
    },
    {
      "payerId": "centene-bulk-test",
      "payerName": "Test Payer",
      "payerDescription": "Test Description",
      "isPayerCustomer": true,
      "isPayerConnected": null,
      "plans": []
    },
    {
      "payerId": "centene-bulk",
      "payerName": "Test Payer",
      "payerDescription": "Test Description",
      "isPayerCustomer": true,
      "isPayerConnected": null,
      "plans": []
    },
    {
      "payerId": "humana",
      "payerName": "Humana",
      "payerDescription": "Humana Description",
      "isPayerCustomer": true,
      "isPayerConnected": true,
      "plans": []
    },
    {
      "payerId": "centene",
      "payerName": "Test Payer",
      "payerDescription": "Test Description",
      "isPayerCustomer": true,
      "isPayerConnected": null,
      "plans": []
    },
    {
```

3. POST /fhir/r4/Group/

Send a **POST** request to the /fhir/r4/Group/. This request retrieves the FHIR Patient ID values for attributed members.

In the **Bearer Token** tab, ensure that the **TOKEN** field is populated with the variable name used in previous calls.



Select the **Body** tab. The request body includes a JSON-formatted payload with required parameters, such as:

1. **Member information** – Includes the patient's identifier from the payer's system.
   - **Payer details** – Identifies from whom to request the patient's data (**CoverageToMatch**).
   - **Patient consent** – This opt out process specifies which data the payer can release to the provider (not shown here).

The **Body** tab contains a JSON-formatted array of parameters. Sample arrays are available on the Availity Clinical Solutions Support Hub.

NOTE	You must be a registered user to access the Availity Clinical Solutions Support Hub. Contact your Availity representative for assistance.

Response to the POST /fhir/r4/Group/$multi-member-match

The response to this call includes three sections:

- **ResourceIdentifier** – This section includes a group of matched patients.
- **NoMatch** – This section includes a group of unmatched patients.
- **Consent Constraint** – This section has a group of patients who has consent constraints.

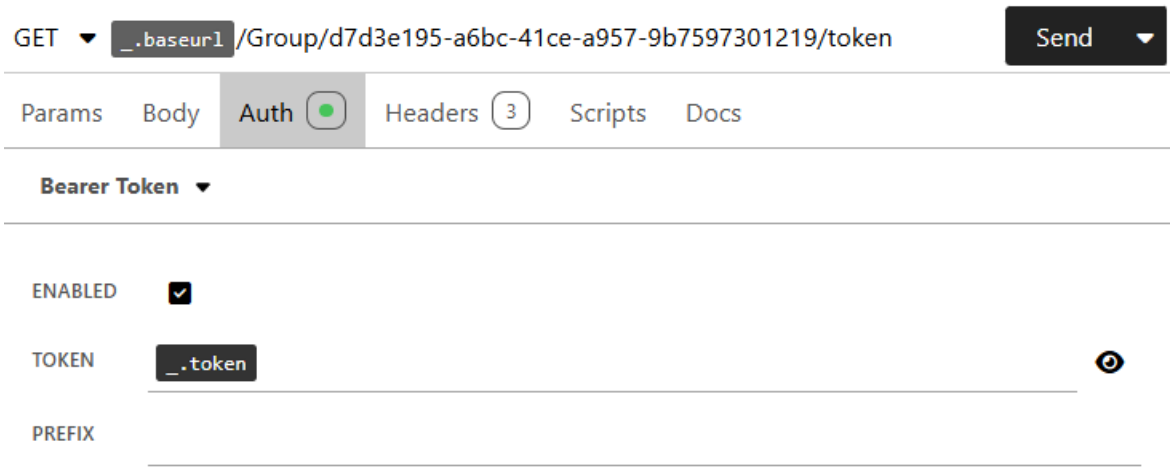
For an example, See this detailed sample response.

NOTE	The Provider Access Connectivity Hub functions as a proxy that masks each party's system details by mapping identifiers between them. As a result, these values are not the payer's literal FHIR Patient ID. Instead, they are Availity-provided, temporary IDs that the Connectivity Hub maps to the payer's FHIR Patient ID to enable the secure exchange of data.

4. GET /fhir/r4/Group/{id}/token

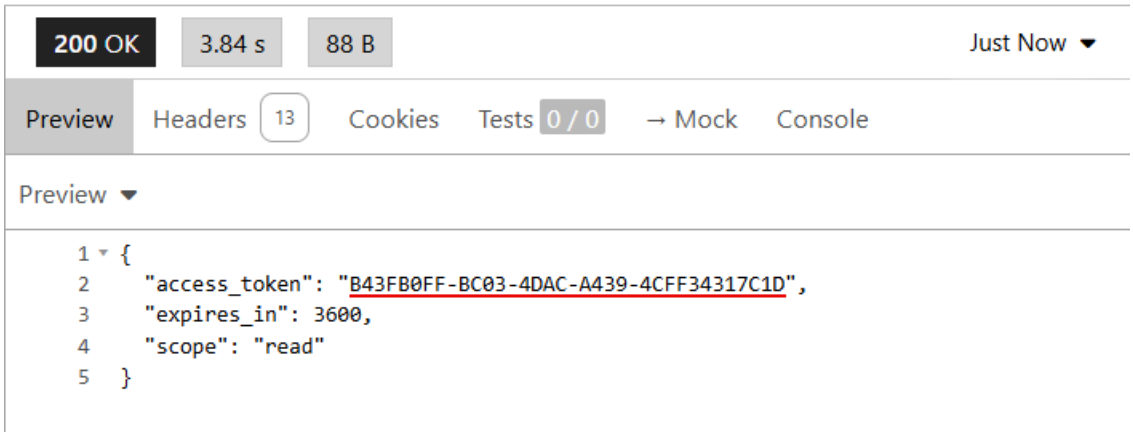
Use your general authorization token to access the Provider Access Connectivity Hub. To access data for multiple patients in a group, you need an additional layer of authentication. This section explains how to obtain the **Group Access Token**.

In the fifth API request, use the Group Access Token along with your general authorization token to request resources for all patients in the group. The Group ID from Step 3 should be included directly in the API call as a **path variable**, placed between the /Group segment and the /token segment.



Response to the GET /fhir/r4/Group/{id}/token call

The response to this request includes the Group Access Token, shown on line 2 in the example below. This **Group Access Token** grants permission to request the FHIR Patient resource for multiple patients on the payer's server. Use the Group Access Token value in the next API call to retrieve all patient resources that belong to the group.



5. GET /fhir/r4/Group/{id}/$davinci-data-export

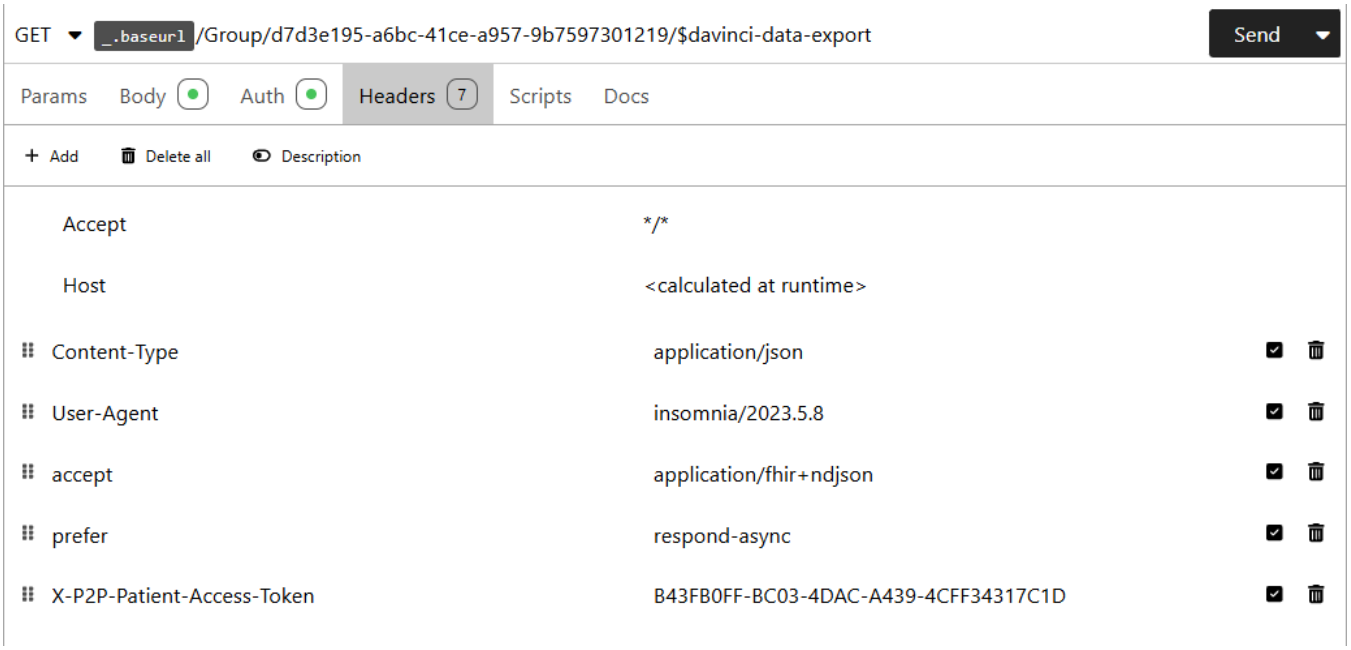
Use the GET /fhir/r4/Group/{id}/$davinci-data-export call to retrieve the data for multiple patients from the payer.

Insert the Group ID as a path variable, just as in Step 4.

In the Headers tab, add a custom header called **X-P2P-Patient-Access-Token** and set its value to the **Group access_token** from Step 4.

The receiving system will use this field to validate that the incoming request has the necessary authentication to obtain the multiple Patient data.

> **NOTE** The usual Bearer Token in the Auth tab is also still required, as in Steps 2, 3, and 4.



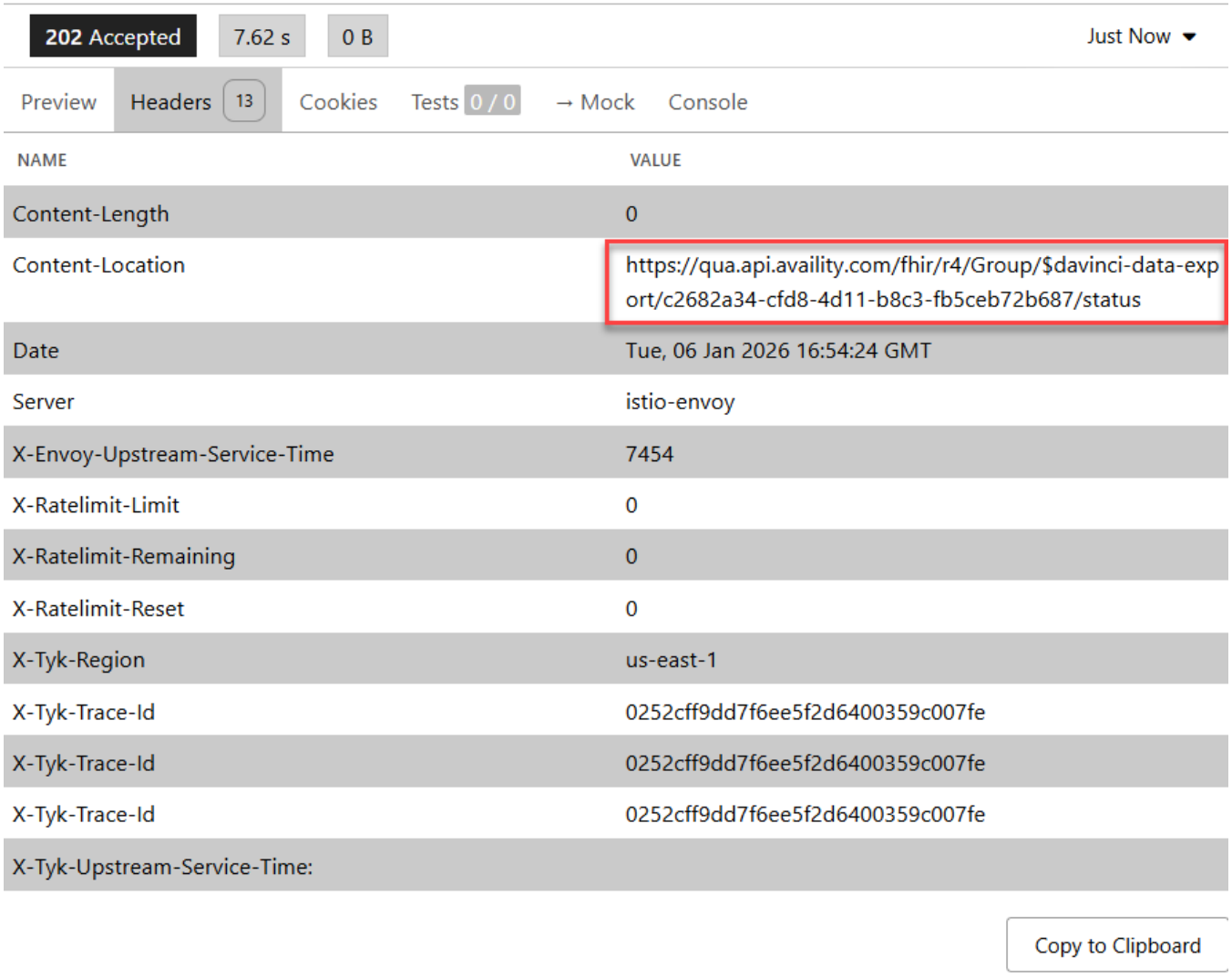
After populating these three values **Auth Token**, **Group ID** in the path, and **X-P2P-Patient-Access-Token** select **Send**.

Response to the GET /fhir/r4/Group/{id}/$davinci-data-export call

The response to the **GET** /fhir/r4/Group/{id}/$davinci-data-export call is a FHIR Bundle containing all the data shared by the payer in accordance with the terms of the Consent opt out check.

> **NOTE** The Mock Payer Service does not evaluate the Consent opt out check.

Use the Content location URL in the Headers tab to download the data,



6. GET /fhir/r4/Group/$davinci-data-export/{id}/status

Use the GET /fhir/r4/Group/$davinci-data-export/{id}/status call to retrieve the URL for obtaining the data of multiple patients from the payer.

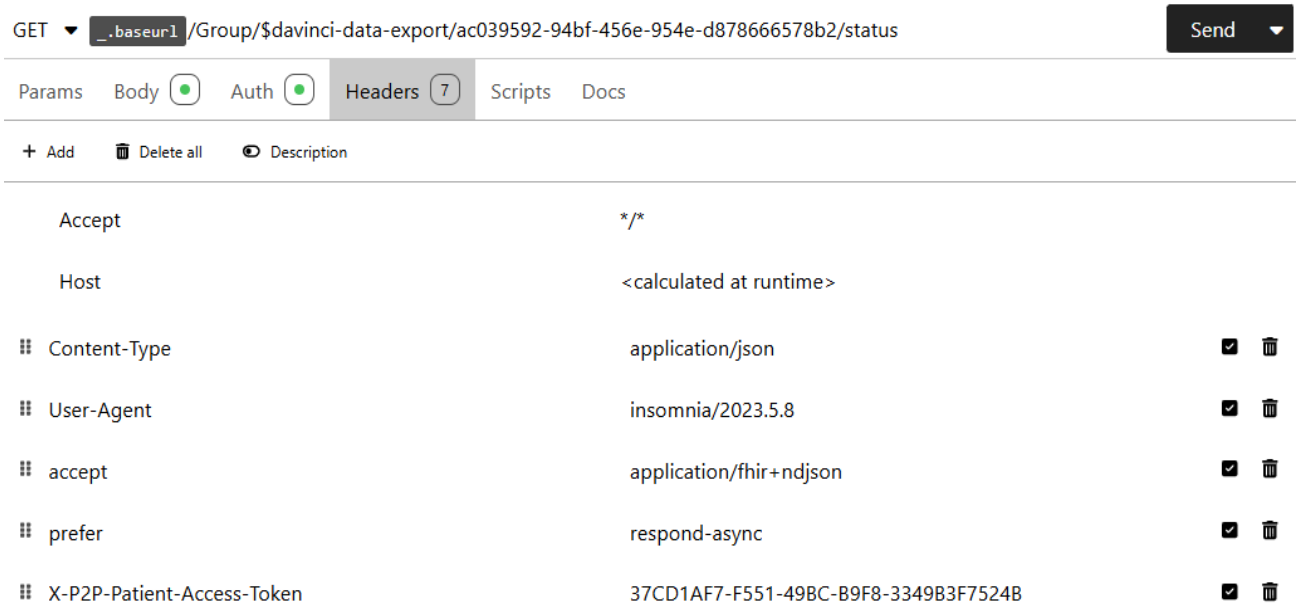
Insert the Group ID as a path variable, just as in Step 4.

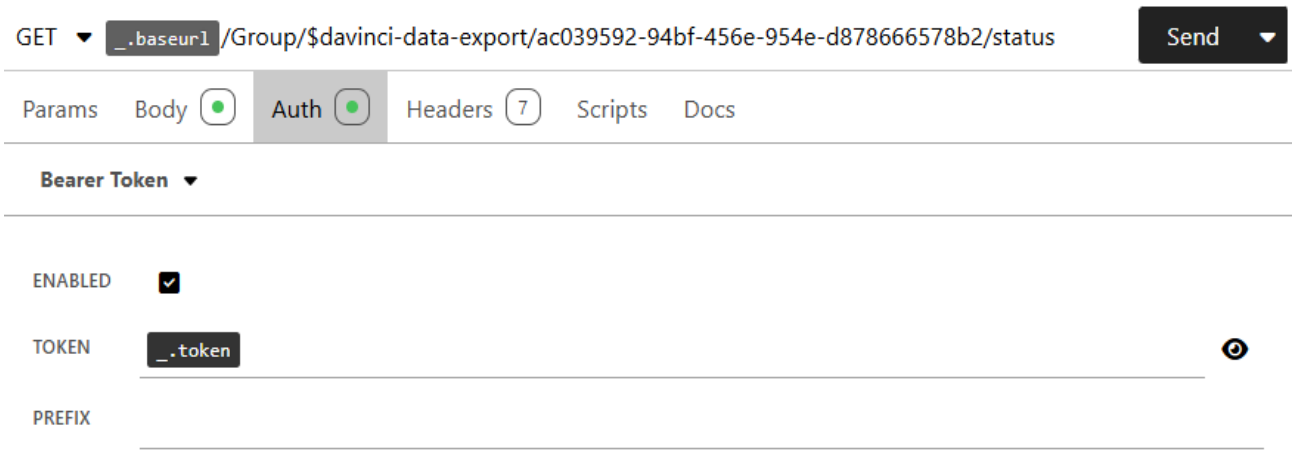
In the Headers tab, add a custom header named **X-P2P-Patient-Access-Token**, as shown below. Give it the value of the **Group access_token** obtained in Step 4.

The receiving system will use this field to validate that the incoming request has the necessary authentication to obtain the data of multiple patients. The request returns a status endpoint URL for tracking the file generation progress.

NOTE	Status URLs will be requested many times using the HL7 FHIR Asynchronous Request Pattern (https://hl7.org/fhir/R4/async.html)

- **Polling with Exponential Backoff**
    - Instead of constant polling, increase the delay between status checks exponentially (e.g., 1s → 2s → 4s → 8s).
    - This prevents overwhelming the server and complies with HL7 guidance.
- **Honor Retry-After Header**
    - If the server includes Retry-After in its response, use that value as the next polling interval.
    - This is critical for respecting server-side throttling.
- **Completion Status**
    - When the process finishes, the status will change to **Complete**.
    - The response will include an **output array** containing URLs for bulk data files.
- **Availity Workflow**
    - Availity will download the files proactively.
    - Provide **proxy URLs** for each file to the provider, ensuring secure and controlled access.





References and Resources

Environment URLs

The URLs for Availity's Provider Access Connectivity Hub environments are provided below. The non-production environment should be used for all testing purposes, as it includes both the synthetic payer and PHI/PII-free patients. Once testing is complete and you are ready to go live, coordinate the switch to the production environment with your Availity support representative.

Non-Production Environment: https://qua.api.availity.com

Production Environment: https://api.availity.com

Constraints within HL7 guides

Payer identifiers

In the PDex implementation guide member-match request, the "**CoverageToMatch**" identifies the payer. The identifier of the payer listed in **CoverageToMatch** is the value Availity expects for that target payer, used for routing API calls to the correct payer. See Member-Match explanation for further details.

Minimum member information

- **First Name**
- **Last Name**
- **Date of Birth**
- **Phone**
- **Address** (postal code alone may be used as address)

Differentiation of providers

A payer will receive requests from multiple providers through a single connection (Availity's Connectivity Hub). To differentiate requests from various providers, the payer must reference the provider identifier included in the call.

Important differences between actors (Payers and Providers)

Enterprises often use an API gateway to manage traffic with the outside world, so session authentication may look different for different actors. In addition, Availity's supported authentication methods may change over time. Ensure that an agreed-upon authentication method is established during the implementation period.

The **X-P2P-Patient-Access-Token** header for the patient-scoped token in a **$davinci-data-export** call is unique to Availity's implementation. See the **$davinci-data-export** call explanation for details.

External References

The HL7 resources below may provide valuable background information as your PDex solution is integrated with Availity's Provider Access Connectivity Hub.

HL7 Da Vinci PDex: https://build.fhir.org/ig/HL7/davinci-epdx/

HL7 Da Vinci HRex: https://hl7.org/fhir/us/davinci-hrex/STU1.1/

US Core 3.1.1: https://www.hl7.org/fhir/us/core/STU3.1.1/

US Core 6.1.0: https://hl7.org/fhir/us/core/STU6.1/

HL7 FHIR v4: http://hl7.org/fhir/R4/