athenahealth

End user authentication and authorization

The below documentation is for end user authentication using 3-legged OAuth credentials only. For service-to-service authentication using 2-legged OAuth credentials, please refer to the authentication documentation on the athenahealth Developer Portal.

Last updated August 10, 2021.

Contents

athenahealth

Introducing the athenahealth login

Just as you can register and log in with an athenahealth account to access the Developer Portal, patients can use their athenahealth account to access the athenahealth Patient Portal and other third-party services supporting these credentials. Incorporating the athenahealth login widget in your app provides a few key benefits:

- Reduces friction for end users, such as physicians and patients, who do not have to register a new set of credentials to use your app. These users only need to maintain a single email and password across applications that support login using an athenahealth account.
- Relieves you of the difficult work of registering, securing, and managing new user identities and permissions around sensitive health information.
- Incorporates lightweight, open, and API-based standards for easy accessibility by developers new to healthcare app development and athenahealth's platform.
- Supports the SMART on FHIR specification for interoperability and to minimize the effort required to connect your app with different technology platforms in healthcare.
- Ensures regulatory compliance and satisfaction of technical requirements established by federal and state governments regarding patient electronic access and healthcare interoperability (for example, see ONC's Cures Act Final Rule in relation to athena's list of APIs in adherence to 2015 Edition CEHRT Criteria and the below documentation for end user authentication).

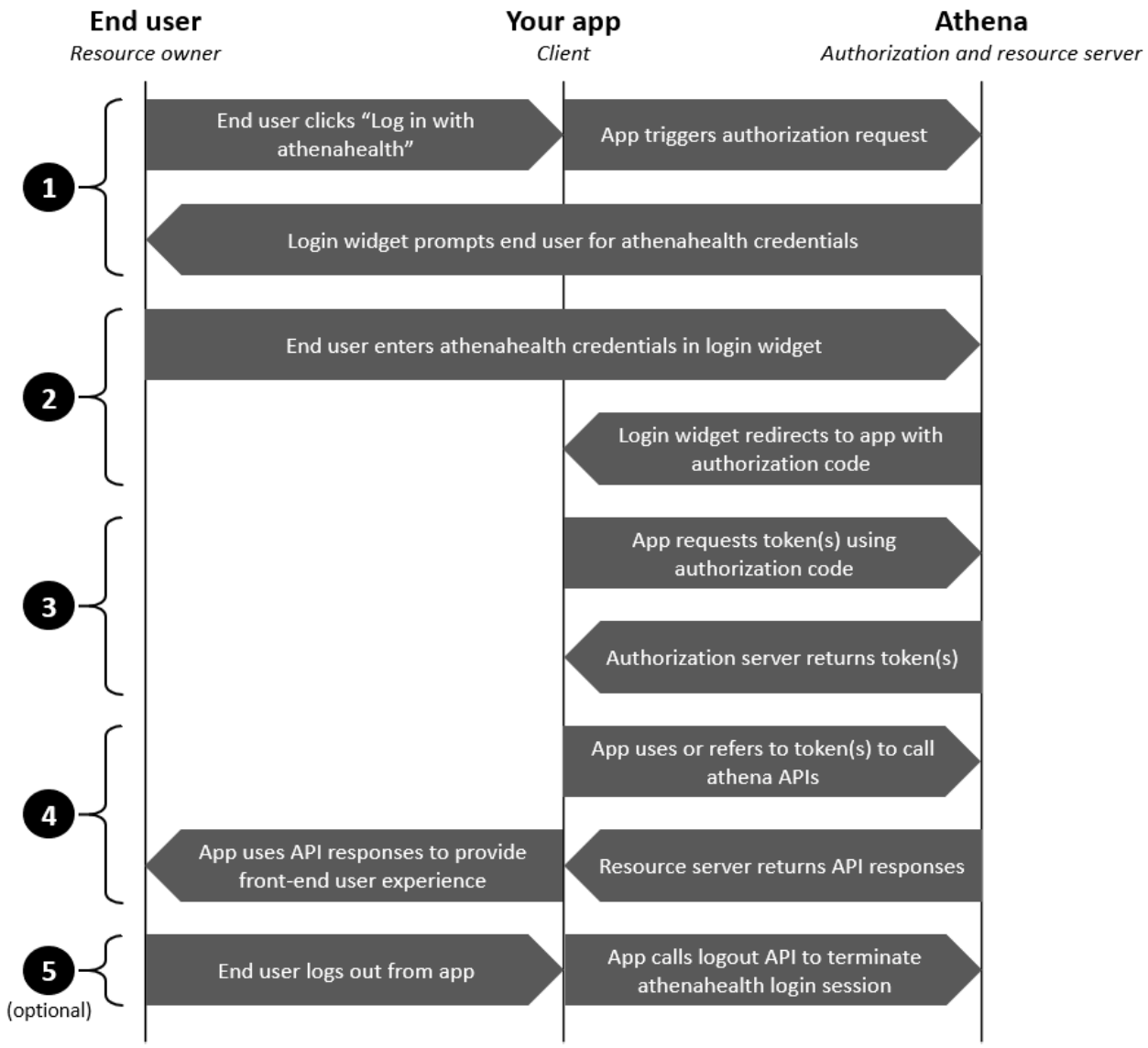
Before you begin

The athenahealth login widget uses OpenID Connect (OIDC) for authentication (verifying the user's identity) on top of the Authorization Code Grant workflow of OAuth 2.0 (for verifying the user's access permissions). This guide for connecting the athenahealth login widget to your application assumes you are familiar with these industry standards for requesting and using JSON web tokens and that you have already registered your Certified API app with athenahealth to obtain a client_id (and client_secret, if applicable). If this is your first time using these standards, we recommend reading through the links to the respective topics above.

Incorporating the athenahealth login requires configuring your app to request and receive tokens from athenahealth's OAuth 2.0 authorization server, hosted by Okta. The overall pattern for this exchange is:

1. End user clicks the "Log in with athenahealth" button in your app, which your app has configured to make a GET request to athena's authorization endpoint per the guidance and sample code below (see **The "Log in with athenahealth" button**). This request redirects the end user to the athenahealth login widget (see **Forming an authorization request**).
2. End user authenticates themselves using their athenahealth account email and password, returning them to your app's redirect URI with an appended authorization code that your app has been configured to accept and form into a request to athena's token endpoint (see **Redirect and token exchange**).
3. Your app makes a token request (see **Forming a token request**) and is returned a JSON response containing the requested tokens.

4. Your app parses the token JSON response to make or inform calls to those athenahealth APIs necessary to power your app. For example, your app might extract the access token to use as a bearer token for API calls authorized by the end user (3-legged OAuth), reference patient IDs and permissions from the ID token as parameters for service-to-service API calls (2-legged OAuth), or use the refresh token to request a new access token when the previous one expires. For more information, see **Using tokens**.
5. (Optional) If your app manages its own login session separate from the athenahealth login widget, your app may call athena's logout endpoint upon logout from your app to ensure the end user must re-authenticate with their athenahealth account the next time they access your app (see **Logging out**).



The "Log in with athenahealth" button

To give end users a consistent experience and to ensure they recognize the credentials they will use to authenticate, apps must use the standard "Log in with athenahealth" button to initiate login. See the example code for this button:

"Log in with athenahealth" button – blue (for light backgrounds)

"Log in with athenahealth" button – white (for dark backgrounds)

Note: The button HTML supplies the request parameters and button type (light or dark) to the button JavaScript file, which compiles the authorization request URL that launches athenahealth's login widget. The button CSS contains the required button formatting for light and dark backgrounds.

Forming an authorization request

The sample "Log in with athenahealth" button code above calls athenahealth's production authorization endpoint to launch the athenahealth login widget, which you may need to modify for your app and intended athenaNet environment. The authorization endpoint accepts the parameters below, which you will need to configure in your app's "Log in with athenahealth" button code.

Authorization Endpoint (GET)	
Production URL	`https://myidentity.platform.athenahealth.com/oauth2/v1/authorize`
Preview (sandbox) URL	`https://myidentity-support.platform.athenahealth.com/oauth2/v1/authorize`
Parameters	
Parameter	**Description**
`client_id`	Registered client ID of your app (3-legged OAuth credentials).
`response_type`	Must be set to `code` to obtain an authorization code for requesting tokens (see **Redirect and token exchange** below).
`redirect_uri`	Post-authentication redirect URI for your app, which must be configured to accept a `code` parameter to form a token request (see **Redirect and token exchange** below). **Note**: athenahealth must whitelist your redirect URI for your client_id or the authorization request will fail. You can provide athena with one or more redirect URIs to whitelist for your app when requesting your production client_id.
`scope`	Space-delimited string specifying the scopes your app is requesting (see **Scopes** below). At minimum you must request the OIDC workflow scope

	(`openid`) for successful authentication and to later obtain an ID token from your token request.
`nonce`	A one-time use arbitrary value (string) provided by your app and required if you need an ID token – but optional if you only require an access token. The value you provide here will be returned in the ID token so your app can compare the two values to prevent replay attacks.
`state`	An optional parameter that can be assigned an arbitrary value (string) that is then returned in the post-authentication redirect to your app's `redirect_uri` (i.e., a passthrough parameter). Per OAuth specification, `state` may be used to prevent cross-site request forgery attacks.
`aud`	Required only for patient apps, which must use either of two methods to specify in the `aud` parameter the athenaNet practice and Patient Portal brand against which the end user is authenticated: • The first method is to pass the launch URL (FHIR base URL), which is recommended for apps requesting the `launch/patient` scope (SMART on FHIR) and additionally specifies the chart group from which the patient record is retrieved. For example, in production, the `aud` parameter or launch URL for calling FHIR DSTU2 APIs where practice ID = 98765, brand ID = 2, and chart group ID = 24 would be: `https://apitest.athenahealth.com/v1/98765/2/24/fhir/dstu2` In athena's preview environment, the URL in the `aud` parameter would be: `https://api.athenahealth.com/preview1/98765/2/24/fhir/dstu2` • The second method is to specify practice and brand using the following format that is agnostic of athenaNet environment: `{"PRACTICEID":"98765","COMMUNICATORBRANDID":"2"}`

`code_challenge`
 	
and
 	
`code_challenge_method`
	Apps are recommended by athena per OAuth best practice and SMART App Launch v2.0.0 proposals to use a proof key for code exchange (PKCE, pronounced "pixie") in place of a `client_secret` in the token request (see **Requesting a token** below). To leverage PKCE for your app, please refer to the IETF's official overview and to Okta's implementation documentation for details and restrictions of these parameters.

	In the authorization request, `code_challenge` is a hashed string and `code_challenge_method` indicates how the `code_challenge` has been transformed from a `code_verifier` generated by your app (e.g., `plain` if no transformation, or `S256` if encoded using SHA-256) and later provided in the token request.

Scopes: Each scope represents a permission your app can request. Different scopes can determine the type of app launch, token contents, or APIs your app can access, but only if those scopes are granted. Scopes can either be requested in an authorization request or refresh token request (see **Forming a token request**). There are three checks to determine which scopes are granted from a request by your app:

1. Is the scope requested? Only scopes that are included in the `scope` parameter of the request can be granted.
2. Is your app permitted to request the scope? Only scopes your app has been approved to access by athena can be granted. By default, any app can request the `openid`, `offline_access`, and `launch/patient` scopes, as well as the following FHIR resource-level scopes associated with athena's Certified APIs. Apps may only request these resource-level scopes and not the wildcard scope (`patient/*.read`):

    - `patient/Patient.read` (**Note:** This scope gates 3-legged access to both the 2015 CEHRT APIs for Patient and Patient Demographics.)
    - `Patient/AllergyIntolerance.read`
    - `patient/Assessment.read` (**Note:** The API to access this resource uses a different FHIR base URL based on department (see 2015 CEHRT APIs).)
    - `patient/CarePlan.read` (**Note:** This scope gates 3-legged access to both the 2015 CEHRT APIs for CarePlan and Assessment where APIs for both resources use a different FHIR base URL based on department.)
    - `patient/CareTeam.read` (**Note:** The API to access this resource uses a different FHIR base URL based on department (see 2015 CEHRT APIs).)
    - `patient/Condition.read` (**Note:** This scope gates 3-legged access to the 2015 CEHRT APIs for Problems.)
    - `patient/Device.read` (**Note:** The API to access this resource uses a different FHIR base URL based on department (see 2015 CEHRT APIs).)
    - `patient/Immunization.read`
    - `patient/MedicationStatement.read`
    - `patient/Observation.read` (**Note:** This scope gates 3-legged access to the 2015 CEHRT APIs for Results, Smoking Status, and Vitals.)
    - `patient/Procedure.read`

3. If the scope determines API access and requires end user consent (e.g., offline_access or any of the FHIR resource-level scopes), does the user grant their consent? For an authorization request, the user is prompted to consent or deny access to any scopes requiring this consent in the athenahealth login widget prior to redirect. The end user's consent decisions are recorded by athena's authorization server and enforced upon subsequent refresh token request.

All requested scopes must pass both of the first two checks or the request will return an unauthorized error. Scopes that pass the first two checks but are denied consent by an end user will simply be excluded from the access token, thereby restricting calls to the API(s) gated by those scopes, while the rest of the requested scopes will be granted and included in the access token.

Other useful scopes include:

- `open_id`: OIDC workflow scope required for successful authentication and to later obtain an ID token from your token request.
- `offline_access`: If granted, returns a refresh token in the token response used for persistent access without requiring the end user to re-authenticate.
- `launch/patient`: Required for SMART on FHIR launch to access patient resources. You will need to include the patient launch scope (`launch/patient`) in addition to the scopes corresponding to those patient resources your app will be accessing.
- `athena/user/Identity.PatientMappings.read`: Adds a patient identity mappings (`pim`) claim in the ID token response listing the patient IDs, athenaNet practices and brands, and access levels associated with the authenticated user (see **Using tokens** for more details).
- `email`: Returns the user's email address in the ID token response.

Managing multiple athenaNet practices and brands: Through the `aud` parameter, athena's authorization endpoint currently requires patient users to be authenticated against a specific athenaNet practice and Patient Portal brand. Therefore, apps that are intended to provide end users access to multiple athenaNet tablespaces (across multiple unaffiliated doctor's offices) will need to have their own system for creating and managing unique entry points or listings within the app for different athenaNet gateways, for which the `aud` parameter in each gateway's authorization request (within the app's "Log in with athenahealth" button code) varies accordingly.

Redirect and token exchange

In addition to configuring your authorization request within the "Log in with athenahealth" button, you will also need to set up the redirect URI in your app to accept `GET` requests with a `code` parameter. This request is returned by athena's authorization endpoint following end user authentication. Your app's redirect URI must then use the `code` parameter to form a `POST` request to athena's token endpoint (see **Requesting a token** below).

The standard signature of the `GET` method for your app's redirect endpoint should include:

- `code` - Obscure authorization code generated by the authorization server and passed to the redirect URI as a parameter.

- `state` – The passthrough value provided in the `state` parameter of your authorization request, if applicable.

Forming a token request

To obtain a token (or set of tokens), your app will need to make a `POST` request to athena's token endpoint with the below parameters. The token endpoint returns a JSON response with the requested tokens.

Token Endpoint (POST)	
Production URL	https://myidentity.platform.athenahealth.com/oauth2/v1/token
Preview URL	https://myidentity-support.platform.athenahealth.com/oauth2/v1/token
Parameters	
Parameter	**Description**
`client_id`	Registered client ID of your app (3-legged OAuth credentials).
`redirect_uri`	Post-authentication redirect URI for your app. The `redirect_uri` provided in the token request must match the `redirect_uri` in the original authorization request.
`grant_type`	Required. For initial token requests: must be set to `authorization_code` to specify the OAuth 2.0 Authorization Code Grant workflow used for initial end user authentication. For subsequent token requests using a refresh token (does not require end user authentication): must be set to `refresh_token`.
`code`	Required only for initial token requests (where `grant_type = authorization_code`). The value must match that of the `code` parameter returned by athena's authorization endpoint to your app's redirect URI (see **Redirect and token exchange**).
`client_secret`	Required for apps issued a client secret upon app registration, even if your app chooses to use the recommended PKCE authentication flow (see `code_challenge` and PKCE details in **Forming an authorization request**). Apps not issued a client secret should omit this parameter in their token request. **Note:** Only apps capable of securely storing a client secret (i.e., apps with a server backend, or "private" apps) should be issued one, while all other apps (i.e., "public" apps) must use PKCE. Please contact athena if you believe your app has been incorrectly registered as a private app.
`code_verifier`	A one-time use, unique value generated by your app, for which the hashed string was supplied in the `code_challenge` parameter of your app's authorization request as part of PKCE (see `code_challenge` and PKCE details in **Forming an authorization request**).

	This parameter is only required for apps using PKCE and must conform to Okta's code verifier requirements.
`refresh_token`	Required only for refresh token requests (where `grant_type = refresh_token`). Value must be set to the `refresh_token` value returned by the token endpoint in response to your app's initial token request.
`scope`	Only used for refresh token requests (where `grant_type = refresh_token`). Your app may request all or a subset of those scopes requested and granted in your initial authorization request, which similarly determine the new token contents and APIs your app can call using the new access token.

Using tokens

Three tokens can be returned by the token endpoint: ID tokens, access tokens, and refresh tokens. ID tokens include information on the user and data they are able to access, which may inform additional 2-legged (i.e., service-to-service) API calls made by your app. Access tokens can be used as bearer tokens to make 3-legged (end user-authenticated) API calls – e.g., to call athena's FHIR APIs per the SMART on FHIR specification. Refresh tokens are used to obtain a new access token when the previous token expires and does not require the end user to re-authenticate.

Token durations vary based on type and configuration for your app. Default durations for each token type are:

- ID token – 60 minutes
- Access token (only used for 3-legged OAuth) – 5 minutes
- Refresh token (only used for 3-legged OAuth) – 100 days, where clock is reset every time it's used to get a new access token

How and when your app will use the tokens returned by the token endpoint depends on your app's use case. Most apps will use the athenahealth login widget to make 3-legged API calls to retrieve athenaNet data accessible by the user. To do so, they must use the access token as a bearer token to call those APIs. In scenarios where the app developer already has broad service-to-service API access to athenaNet data (e.g., a healthcare provider building an app for their own practice), the app may use the athenahealth login widget for authentication only, either looking only for a token endpoint success or referencing the end user information in the ID token to inform 2-legged API calls to athenaNet.

To use an access token for patient SMART on FHIR: Your app must pass the access token (64-bit encoded, exactly as returned by the token endpoint) as a parameter in the `GET` request to the FHIR API. The `GET` request must also include any other arguments the API requires (e.g., the athenaNet practice ID, Patient Portal brand ID, chart sharing group ID, and patient ID of the target patient record for FHIR chart APIs). Only athena's 2015 CEHRT Edition FHIR APIs with `{brandid}` in the base URL support 3-legged OAuth and will accept the access token as a bearer token, with the only exceptions being the non-FHIR APIs for Assessment, CarePlan, CareTeam, and Device with `{departmentid}` in the base URL. For the list of scopes associated with these 2015 CEHRT Edition APIs, see **Scopes**.

For FHIR APIs with {brandid} in the base URL, the practice ID, brand ID, and chart sharing group ID are all specified using the base URL. Patient apps using SMART on FHIR will need to modify their FHIR base URLs for each combination of these IDs (gateway). The gateways for an athenahealth customer are provided to you by athena upon that customer's expressed interest in activating that app for their patients (e.g., using the "Get Started Now" button for an app on the athenahealth Marketplace). Department-level mappings are also provided for these gateways to call non-FHIR Certified APIs using {departmentid}. Both categories of endpoint also require a patient ID, which is returned in the launch context of the token response alongside the ID and access tokens (see HL7's SMART App Launch documentation).

Your app will be constrained to GET requests only for (a) the patient ID included as a parameter alongside the token endpoint return, and (b) the APIs associated with those scopes granted by the end user (which may only be a subset of those in your app's authorization request), as specified in the access token.

To use an access token for provider SMART on FHIR: This use case is not yet formally supported – please refer to the athena Developer Portal authentication documentation for updates.

To use an ID token to inform 2-legged (service-to-service) API calls: If your app requested the athena/user/Identity.PatientMappings.read scope, your app can parse the id_token returned from the token endpoint for a pim claim, which contains the athenaNet practice, Patient Portal brand, patient IDs, and permissions for those records to which the end user has access. Your app can use these IDs to populate required arguments for any other athenaNet service-to-service APIs detailed in the athena Developer Portal that your app is permitted to access (e.g., to book appointments, check in a patient, or retrieve billing information) – either using the same 3-legged OAuth credentials (client ID) or separate OAuth credentials (client ID) requested and issued through the Developer Portal console for 2-legged access only.

The pim claim is structured as follows, where ctxt is the athenaNet practice ID, brnd is the Patient Portal brand ID, ptnt is the patient ID, and access is the type of Patient Portal permissions the user has to that health record: full access as the patient (SELF), full access as a family member or caregiver (FULL), or billing-only access (BILLING). For example:

```
"pim": [
{  "ctxt": 4321, "brnd": 1,  "ptnt":  1234",  "access": "SELF" },
{  "ctxt": 4321, "brnd": 1,  "ptnt":  2000",  "access": "BILLING" },
{  "ctxt": 4321, "brnd": 2,  "ptnt":  1235 , "access": "FULL" }
]
```

- **Note:** Because the pim claim provides patient information without end user authentication, only apps with broad service-to-service access to athenaNet APIs will be whitelisted by athena for the patient identity mappings scope (athena/user/Identity.PatientMappings.read) to obtain this claim. Without this scope, the pim claim will not be included in the ID token.

To refresh a token (request a new access token when the previous one expires): Use the refresh_token parameter returned from your app's initial token request (where grant_type = authorization_code) to call the token endpoint again, this time using grant_type =

`refresh_token`. Your app must provide the refresh token value in the token endpoint's `refresh_token` parameter and specify which of the previously granted scopes your app would like included in the access token (refer to token endpoint details in **Requesting a token**).

- **Note:** You will not receive a refresh token from your app's initial token request unless you include and are granted the `offline_access` scope with your app's authorization request.

Logging out

Login sessions remain active until they time out after 10 minutes of idle time (which can present security risk, particularly on devices shared by multiple end users) or until logout is triggered by your app. To trigger logout, your app must make a `GET` request to athena's logout endpoint. Note that login sessions are distinct from token duration and terminating a login session will not invalidate any active access or refresh tokens. Terminating a login session ensures that an end user will be forced to reauthenticate upon the next authorization request from your app, whereas they would otherwise be automatically redirected and not prompted for their email and password if a previous login session were still active.

Logout Endpoint (GET)	
Production URL	https://myidentity.platform.athenahealth.com/oauth2/v1/logout
Preview URL	https://myidentity-support.platform.athenahealth.com/oauth2/v1/logout
Parameters	
Parameter	**Description**
`id_token_hint`	Current ID token for the open session. Value must be set to the `id_token` value returned by the token endpoint following your app's latest token request (regardless if it was an initial request where `grant_type = authorization_code` or a subsequent `refresh_token` request).
`post_logout_redirect_uri`	The redirect URI where the end user should land after logout, such as the landing page for your app where you have included the "Log in with athenahealth" button. **Note:** This redirect URI must be among those whitelisted as part of your app's registration with athenahealth. Please contact athena if you need additional redirect URIs whitelisted.
`state`	Optional passthrough parameter appended to your redirect URI post-logout.

Additional resources

- OpenID Connect & OAuth 2.0 API (Okta)
- Authorization Code Grant Type (Okta)
- FHIR DSTU2 (HL7)
- athenahealth Certified APIs
- athenahealth Complete List of APIs
- athenahealth 2-legged OAuth documentation

Troubleshooting FAQs

Errors calling the authorization endpoint

Q: Authorization API returns "Error: Invalid client_id. Verify the client_id query parameter."

A: This error indicates your value for the `client_id` parameter is incorrect. Please check your client ID for typos and verify that both your client ID and authorization URL are for the correct athenaNet environment, as both values will vary between Production and Preview (sandbox).

Q: Authorization API returns 403 response with {"error":"access_denied","error_description": "Policy evaluation failed for this request, please check the policy configurations."}

A: This error indicates your app is requesting a scope outside of those your app is permitted to request. See **Scopes** for more information on which scopes are available to apps by default.

Q: Authorization API returns {"error":"invalid_scope","error_description":"One or more scopes are not configured for the authorization server resource."}

A: This error indicates your app is requesting a scope that does not exist in athena's authorization server. Please check the `scope` parameter in your authorization request to ensure there are no typos and that it is not URL-encoded. For example, `openid athena/user/Identity.PatientMappings.read` is acceptable in the `scope` parameter (note the space between the scope names) whereas the URL-encoded version `openid%3Bathena%2Fuser%2FIdentity.PatientMappings.read` is not (both because of the "`%3B`" in place of the space delimiter and because of the URL-encoding of the scope name).

Q: Authorization API returns 500 response with "error in workflowHandler"

A: This error indicates the athenahealth login widget is having difficulty interpreting contents of your app request, most commonly due to incorrect formatting of the athenaNet practice ID and brand ID passed in the `aud` parameter (see `aud` parameter requirements in **Forming a token request**). Please check the aud parameter in your authorization request to ensure there are no typos and that it is not URL-encoded. For example, `{"PRACTICEID":"12345","COMMUNICATORBRANDID":"1"}` is acceptable in the `aud` parameter whereas the URL-encoded version `%7B%22PRACTICEID%22:%22195900%22,%22COMMUNICATORBRANDID%22:%221%22&7D` is not.

Q: Authorization API returns "503: Bad Gateway"

A: This error indicates an outage with the athenahealth login service. Please try again later.

Q: Authorization API returns "504: Gateway Timeout" or spins upon login without redirecting

A: This error may indicate there is a trailing slash on the launch URL provided in the `aud` parameter (in which case removing the trailing slash will resolve the error) or that your app is misconfigured. If you believe your app is misconfigured and you have ruled out other errors described in this document, please reply to the email in which you were provided your client ID with a description of your error and steps to recreate so athena can verify your app's configuration.

Q: Authorization API returns "PKCE code challenge is required when the token endpoint authentication method is 'NONE'

A: This error indicates athena has registered your app's client ID as "public" (i.e., is not issued and cannot securely store a client secret), and you will therefore need to use a proof key for code exchange (PKCE) instead of the `client_secret` parameter. Please refer to the `code_challenge` parameter description under **Forming an authorization request** on how to use PKCE. Including the necessary parameters for PKCE in your authorization request should resolve this error.

Errors authenticating a user

Q: Login widget appears to accept credentials but returns error "You are not configured to access this Patient Portal."

A: This error only appears if the authenticated user has an athenahealth account but does not have access to any patient records in the athenaNet practice and brand specified in the `aud` parameter of your app's authorization request. You may encounter this error if you have an active session in the athenahealth Developer Portal and attempt patient login through the same browser, which would automatically log in your Developer Portal account and return this error if your email address used with the Developer Portal has not been granted access to any athenaNet patient records. In this scenario, you should first terminate the active login session by logging out from the Developer Portal (or any other app with which you're using your athenahealth account in that browser) and/or clear the browser cache. Then, retry logging in via your app's authorization request.

Errors calling the token endpoint

Q: Token response is successful (200) but the ID token claims are empty

A: ID token contents are filtered based on those athenaNet tablespaces for which your app is whitelisted by athena. If you believe your app should have access to a tablespace (e.g., if you have been notified of an athenahealth customer's interest in activating your app but continue to get this error), please have the customer notify their Customer Success Manager to submit and resolve your case.

Q: Token endpoint returns {"error":"invalid grant","error_description":"The grant was issued to another client. Please make sure the 'client_id' matches the one used."

A: This error indicates your app's client ID used in the token request does not match the client ID. This is an error often experienced by apps testing across different environments (which requires multiple client IDs) where the client ID has been updated in the app's authorization request code and not the token request.

Q: Token endpoint returns a 400 response with {"error":"invalid_grant","error_description":"PKCE verification failed."}

A: This error indicates that your PKCE code (originally provided in the `code_challenge` parameter of your app's authorization request) is no longer valid, most commonly for one of two reasons:

a) The code has expired after 60 seconds following the successful authorization request.
b) The PKCE `code_challenge` your app provided in the authorization request does not match the authorization server's transformed version of your `code_verifier` by the method indicated in the `code_challenge_method` of your authorization request. Note athena's authorization server uses our host's (Okta's) following transformation steps for a `code_verifier`, where "S256" is the `code_challenge_method`:
    1. Hashes the provided `code_verifier` string using SHA-256 to return a byte array.
    2. Converts the byte array into a string and Base64-encodes the string.
    3. Modifies the Base64-encoded string to make it URL-safe, specifically by replacing each "+" with "-", replacing each "\" with "_" and removing all "=".

    Note: A PKCE `code_verifier` must be 43 to 128 characters in length. For more information and resources for implementing PKCE, see **Forming an authorization request**).

Errors calling FHIR APIs using 3-legged OAuth

Q: FHIR API returns a 403 response with {"error":"Incorrect permissions"}

A: This error may indicate one or both of the following conditions:

a) Your app is not whitelisted for access to the athenaNet tablespace specified in the base URL and/or parameters of the FHIR API call. If you believe your app should have access to a tablespace (e.g., if you have been notified of an athenahealth customer's interest in activating your app but continue to get this error), please have the customer notify their Customer Success Manager to submit and resolve your case.
b) Your app is not whitelisted for access to the API you are attempting to call. For apps registered to use athenahealth's Certified APIs, this would include non-GET calls to FHIR APIs or any calls made to endpoints outside of those APIs athena has certified to 2015 Edition CEHRT (g)(7), (g)(8), or (g)(9) criteria.

Q: FHIR API returns 500 response with {"error":"Invalid permissions, incorrect scope passed."}

A: This error indicates your app has not been granted the scope needed to call this FHIR API, either because your app is not whitelisted for that scope or because the scope was denied by the end user in the authentication workflow and therefore not included in the resulting access token.

Q: FHIR API returns 406 response with {"error":"An unsupported Accept header was given."}

A: Per DSTU2 specification, athena's FHIR APIs only permit an Accept header beginning with "application/json" or "application/json+fhir" where trailing characters (e.g., for a charset) are permitted. Some apps may get the above error if using "application/fhir+json," used by later FHIR versions. Apps can also avoid the above error by ensuring their Accept headers are passed for JSON, not XML.

Miscellaneous

Q: Can I call other authorization server endpoints beyond /authorize, /token, and /logout?

A: While your client ID will not be restricted from calling all Okta authorization server APIs (using the same athenaNet environment-specific base URLs documented in earlier sections for /authorize, /token, and /logout), only the following endpoints are formally supported today:

- /authorize (see **Forming an authorization request**)
- /token (see **Forming a token request**)
- /logout (see **Logging out**)
- /introspect (used to check the status and contents of token; see Okta documentation)
- /revoke (used to invalidate an active access or refresh token; see Okta documentation)
- /keys (used to verify token signatures; see Okta documentation)

The following endpoints are not guaranteed to contain accurate or complete information and should not be referenced:

- /userinfo
- /.well-known/oauth-authorization-server
- /.well-known/openid-configuration

Q: Are there any tools you recommend for validating the Login with athenahealth workflow?

A: Yes – athena recommends using Postman, consistent with our documentation for testing service-to-service API calls using 2-legged OAuth. Note that most parameters (base URLs, client IDs, scopes, etc.) will vary from the 2-legged OAuth scenario and should refer to the appropriate values provided in this documentation or by athena when registering your app. Once you complete validation, you can then use Postman to generate code snippets for your app itself.

Q: What is the process for connecting my app with real athenahealth customer sites and patients?

A: Processes vary based on whether the developer is an athenahealth customer, which APIs the app will be accessing, and whether the app will be using the athenahealth login for patients or providers. The overall process for a third-party developer (i.e., neither an athenahealth customer nor their business associate) building a service-to-service app calling any athena API is outlined here. Provider SMART on FHIR apps using the athenahealth login are not yet formally supported. The process for a patient SMART on FHIR app using the athenahealth login to access Certified APIs only is as follows:

1. Developer requests registration of a Production client ID per the instructions on the Certified API Access page of the athenahealth Developer Portal, indicating in their request their

interest in connecting their app with athenahealth customers through the athenahealth Marketplace.

2. Pending action from athena, developer receives and validates their end user workflow using their Production client ID and has athena publish the tile for their app on the athenahealth Marketplace.
3. Customers of athenahealth discover and learn about the app through the Marketplace and request activation for their athenaNet tablespaces using the "Get Started Now" link on the app's tile. The "Get Started Now" form submission is delivered via email to the app developer as well as the athenahealth operations team, who whitelists the app's access to the customer's athenaNet tablespaces and emails the developer with a CSV file (spreadsheet) containing the associated gateway information (see **Managing multiple athenaNet practices and brands**).
4. Developer references the gateway CSV file to configure the appropriate Login with athenahealth entry points (e.g., enabling authorization calls specifying the athenaNet practice and brand for that customer) within their app. Should the developer like to coordinate any additional testing or seek input from a customer before publishing their app for that customer's patients, the developer can reach the customer at the contact information provided via their "Get Started Now" form submission.