OAuth 2.0 Tutorial

Using OAuth 2.0

Applications must secure and protect the privacy of patients and their data. To help meet this objective, Epic supports using the OAuth 2.0 framework to authenticate and authorize applications.

Why OAuth 2.0?

OAuth 2.0 enables you to develop your application without having to build a credential management system. Instead of exposing login credentials to your application, your application and an EHR's authorization server exchange a series of authorization codes and access tokens. Your application can access protected patient data stored in an EHR's database after it obtains authorization from the EHR's authorization server.

Before You Get Started

To use OAuth 2.0 to authorize your application's access to patient information, two pieces of information need to be shared between the authorization server and your application:

1. `client_id`: The `client_id` identifies your application to authentication servers within the Epic community and allows you to connect to an organization.
2. `redirect_uri`: The `redirect_uri` confirms your identity and is used to validate and redirect authorization requests that originate from your application. Epic's implementation allows for multiple `redirect_uri`s.

You can register your application for access to the sandbox and Epic organizations here. You'll provide information to us, including one or more `redirect_uri`s, and Epic will generate a `client_id` for you.

Apps can be launched from within an existing EHR or patient portal session, which is called an EHR launch. Alternatively, apps can be launched standalone from outside of an existing EHR session.

EHR launch: The app is launched by the EHR calling a launch URL specified in the EHR's configuration. The EHR launches the launch URL and appends a launch token and the FHIR server base URL (ISS parameter) in the query string. The app exchanges the launch token, along with the client identification parameters, to get an authorization code and eventually the access token.

Standalone launch: The app launches directly to the authorize endpoint outside of an EHR session and requests consent from the EHR's authorization server.

Considerations for native apps: Additional security considerations must be taken into account when launching a native app to sufficiently protect the authorization code from malicious apps running on the same device. Android, iOS, and Windows have introduced a method by which a native app can claim an https request URI, and instruct the OS to invoke the app when that specific http URI is navigated to. Apple's iOS calls this feature "Universal Links"; Android, "App Links"; Windows, "App URI Handlers". This association of an http URI to native app is platform-specific and only available on the big mobile platforms. Additionally, although these features are relatively recent: ~80% of deployed Android devices support it worldwide (Android v6 - API level 23) and ~97% of iOS devices have this feature. Epic recommends the use of Universal Links/App Links in lieu of custom redirect uri protocol schemes.

The app you build will list both a production Client ID and a non-production Client ID. While testing in the Epic USCDI on FHIR sandbox, use the **non-production Client ID**.

Our Epic USCDI on FHIR Sandbox URL: https://uscdi.epic.com/interconnect-uscdi-oauth/

EHR Launch (SMART on FHIR)

How It Works

Step 1: Your Application is Launched from the Patient Portal or EHR

Your app is launched by the EHR calling the launch URL specified in the EHR's configuration. The EHR sends a launch token and the FHIR server base URL (ISS parameter).

- `launch`: This parameter is an EHR generated token that signifies that an active EHR session already exists. This token is one-time use and will be exchanged for the authorization code.
- `iss`: This parameter is the FHIR resource server's EHR base URL.

Step 2: Your application retrieves the Conformance statement

To determine which authorize and token endpoints to use in the EHR launch flow, you should make a GET request to the metadata endpoint which is constructed by taking the iss provided and appending /metadata.

If no Accept header is sent, the document returns XML by default. If an Accept header is sent with a value of application/json, application/fhir+json, or application/json+fhir, the document returns JSON.

Here's an example of what a full metadata request might look like.

```
GET https://uscdi.epic.com/interconnect-uscdi-oauth/api/FHIR/DSTU2/metadata
Accept: application/fhir+json
```

Here is an example of what the authorize and token endpoints would look like in the metadata response.

```
"extension": [
  {
    "url": "http://fhir-registry.smarthealthit.org/StructureDefinition/oauth-uris",
    "extension": [
      {
        "url": "authorize",
        "valueUri": "https://uscdi.epic.com/interconnect-uscdi-oauth/oauth2/authorize"
      },
      {
        "url": "token",
        "valueUri": "https://uscdi.epic.com/interconnect-uscdi-oauth/oauth2/token"
      }
    ]
  }
]
```

Step 3: Your application requests an authorization code

Your application now has a `launch` token obtained from the initial EHR launch as well as the authorize endpoint obtained from the metadata query. To exchange the launch token for an authorization code, your app needs to redirect (using HTTP GET) from the launch URI to the authorize endpoint and append the following querystring parameters.

- `response_type`: This parameter must contain the value "code".
- `client_id`: This parameter contains your web application's client ID issued by Epic.
- `redirect_uri`: This parameter contains your application's redirect URI. After the request completes on the Epic server, this URI will be called as a callback. The value of this parameter needs to be URL encoded. This URI must also be registered with the EHR's authorization server by adding it to your app listing.
- `scope`: This parameter describes the information for which the web application is requesting access. A scope of "launch" is required for EHR launch workflows.
- `launch`: This parameter is required for EHR launch workflows. The value to use will be passed from the EHR.
- `state`: This optional parameter is generated by your app and is opaque to the EHR. The EHR's authorization server will append it to each subsequent exchange in the workflow for you to validate session integrity. While not required, this parameter is recommended to be included and validated with each exchange in order to increase security. For more information see RFC 6819 Section 3.6.

Here's an example of an authorization request. You will replace the **[redirect_uri]**, **[client_id]**, **[launch_token]**, and **[state]** placeholders with your own values.

```
https://uscdi.epic.com/interconnect-uscdi-oauth/oauth2/authorize?scope=launch&response_type=code&redirect_uri=[redirect_uri]&client_id=[client_id]&launch=[launch_token]&state=[state]
```

This is an example from the SMART on FHIR launchpad:

```
https://uscdi.epic.com/interconnect-uscdi-oauth/oauth2/authorize?scope=launch&response_type=code&redirect_uri=https%3A%2F%2Fuscdi.epic.com%2Ftest%2Fsmart&client_id=d45049c3-3441-40ef-ab4d-b9cd86a17225&launch=GWCpmKCCTxJaq1JRDISdtSzDqNTxrdiqqTiZuWZhYvMA57QbfIbtKgiYAN9kfKc2kDYfaFriLBLQivMhD8BY-942hYgGQB_6DfevP1uHt_pe6tM_-fRfi2WB_-IuwG&state=abc123
```

Step 4: EHR's authorization server reviews the request

The EHR's authorization server reviews the request from your application. If approved, the authorization server redirects the browser to the redirect URL supplied in the initial request and appends the following querystring parameter.

- `code`: This parameter contains the authorization code generated by Epic, which will be exchanged for the access token in the next step.
- `state`: This parameter will have the same value as the earlier state parameter. For more information, refer to Step 3.

Here's an example of what the redirect will look like if Epic's authorization server accepts the request:

```
https://uscdi.epic.com/test/smart?code=yfNg-rSc1t5Oq6jVAZLyY0BuOOte5KM1y3YUxqsJQnBKfMNsYqOPTyVqcCH3YXaPkLztO9Rvf7bhLqQTwALHcHN6raxpTbR1eVgV2QyLA_4K8HrJO92et3qRXiXPkj7&redirect_uri=https://uscdi.epic.com/test/smart&client_id=d45049c3-3441-40ef-ab4d-b9cd86a17225
```

Step 5: Your Application Exchanges the Authorization Code for an Access Token

After receiving the authorization code, your application trades the code for a JSON object containing an access token and contextual information by sending an HTTP POST to the token endpoint using a Content-Type header with value of "application/x-www-form-urlencoded". For more information, see RFC 6749 section 4.1.3.

The following parameters are required in the POST body:

- `grant_type`: For the EHR launch flow, this should contain the value "authorization_code".
- `code`: This parameter contains the authorization code sent from Epic's authorization server to your application as a querystring parameter on the redirect URI as described above.
- `redirect_uri`: This parameter must contain the same redirect URI that you provided in the initial request.
- `client_id`: This parameter must contain the application's client ID issued by Epic that you provided in the initial request.

Here's an example of what an HTTP POST request for an access token might look like:

```
POST https://uscdi.epic.com/interconnect-uscdi-oauth/oauth2/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=yfNg-rSc1t5Oq6jVAZLyY0BuOOte5KM1y3YUxqsJQnBKfMNsYqOPTyVqcCH3YXaPkLztO9Rvf7bhLqQTwALHcHN6raxpTbR1eVgV2QyLA_4K8HrJO92et3qRXiXPkj7&redirect_uri=https://uscdi.epic.com/test/smart&client_id=d45049c3-3441-40ef-ab4d-b9cd86a17225
```

The authorization server responds to the HTTP POST request with a JSON object that includes an access token. The response contains the following parameters:

- `access_token`: This parameter contains the access token issued by Epic to your application and is used in future requests.
- `token_type`: In Epic's OAuth 2.0 implementation, this parameter always includes the value `bearer`.
- `expires_in`: This parameter contains the number of seconds for which the access token is valid.
- `scope`: This parameter describes the access your application is authorized for.
- `patient`: This parameter provides the DSTU2 FHIR ID for the patient in context at time of launch.
- `state`: This parameter will have the same value as the earlier state parameter. For more information, refer to Step 3.

Note that you can pass additional parameters if needed based on the integration configuration. Here's an example of what a JSON object including an access token might look like:

```
{
  "access_token": "Nxfve4qJH9TKs5fsvf6kRYAZqzK7j9LHvrg1Bw7fU_07_FdV9aRzLCI1GxOn20LuO2Ah15RkRnz-p8u1MeYWqA85T8s4Ce3LcgQqIwsTkI7wezBsMduPw_xkVtLziUZO",
  "token_type": "bearer",
  "expires_in": 3240,
  "scope": "Patient.read Patient.search ",
  "patient": "TIw15bk8n1YVgvWk9D05BmRVRP13ECImNSKBDKyKltsMB",
  "state": "abc123",
}
```

At this point, authorization is complete and the web application can access the protected patient data it requests using FHIR APIs.

Step 6: Your Application Uses FHIR APIs to Access Patient Data

With a valid access token, your application can now access protected patient data from the EHR database using FHIR APIs. Queries must contain an Authorization header that includes the access token presented as a Bearer token.

Here's an example of what a valid query looks like:

```
https://uscdi.epic.com/interconnect-uscdi-oauth/api/FHIR/DSTU2/Patient/TIw15bk8n1YVgvWk9D05BmRVRP13ECImNSKBDKyKltsMB

Authorization: Bearer Nxfve4qJH9TKs5fsvf6kRYAZqzK7j9LHvrg1Bw7fU_07_FdV9aRzLCI1GxOn20LuO2Ah15RkRnz-p8u1MeYWqA85T8s4Ce3LcgQqIwsTkI7wezBsMduPw_xkVtLziUZO
```

Standalone Launch

How It Works

Step 1: Your application requests an authorization code

Your application would like to authenticate the user using the OAuth 2.0 workflow. To initiate this process, your app needs to link (using HTTP GET) to the authorize endpoint and append the following querystring parameters.

- `response_type`: This parameter must contain the value "code".
- `client_id`: This parameter contains your web application's client ID issued by Epic.
- `redirect_uri`: This parameter contains your application's redirect URI. After the request completes on the Epic server, this URI will be called as a callback. The value of this parameter needs to be URL encoded. This URI must also be registered with the EHR's authorization server by adding it to your app listing.
- `state`: This optional parameter is generated by your app and is opaque to the EHR. The EHR's authorization server will append it to each subsequent exchange in the workflow for you to validate session integrity. While not required, this parameter is recommended to be included and validated with each exchange in order to increase security. For more information see RFC 6819 Section 3.6.

Here's an example of an authorization request. You will replace the **[redirect_uri]**, **[client_id]**, and **[state]** placeholders with your own values.

```
https://uscdi.epic.com/interconnect-uscdi-oauth/oauth2/authorize?response_type=code&redirect_uri=[redirect_uri]&client_id=[client_id]&state=[state]
```

This is an example link:

```
https://uscdi.epic.com/interconnect-uscdi-oauth/oauth2/authorize?response_type=code&redirect_uri=https%3A%2F%2Fuscdi.epic.com%2Ftest%2Fsmart&client_id=d45049c3-3441-40ef-ab4d-b9cd86a17225&state=abc123
```

Step 2: EHR's authorization server authenticates the user and authorizes access

The EHR's authorization server reviews the request from your application, authenticates the user, and authorizes access If approved, the authorization server redirects the browser to the redirect URL supplied in the initial request and appends the following querystring parameter.

- `code`: This parameter contains the authorization code generated by Epic, which will be exchanged for the access token in the next step.
- `state`: This parameter will have the same value as the earlier state parameter. For more information, refer to Step 1.

Here's an example of what the redirect will look like if Epic's authorization server accepts the request:

```
https://uscdi.epic.com/test/smart?code=yfNg-rSc1t5Oq6jVAZLyY0BuOOte5KM1y3YUxqsJQnBKfMNsYqOPTyVqcCH3YXaPkLztO9Rvf7bhLqQTwALHcHN6raxpTbR1eVgV2QyLA_4K8HrJO92et3qRXiXPkj7&state=abc123
```

Step 3: Your Application Exchanges the Authorization Code for an Access Token

After receiving the authorization code, your application trades the code for a JSON object containing an access token and contextual information by sending an HTTP POST to the token endpoint using a Content-Type header with value of "application/x-www-form-urlencoded". For more information, see RFC 6749 section 4.1.3.

The following parameters are required in the POST body:

- `grant_type`: For the EHR launch flow, this should contain the value "authorization_code".
- `code`: This parameter contains the authorization code sent from Epic's authorization server to your application as a querystring parameter on the redirect URI as described above.
- `redirect_uri`: This parameter must contain the same redirect URI that you provided in the initial access request.
- `client_id`: This parameter must contain the application's client ID issued by Epic that you provided in the initial request.

Here's an example of what an HTTP POST request for an access token might look like:

```
POST https://uscdi.epic.com/interconnect-uscdi-oauth/oauth2/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=yfNg-rSc1t5Oq6jVAZLyY0BuOOte5KM1y3YUxqsJQnBKfMNsYqOPTyVqcCH3YXaPkLztO9Rvf7bhLqQTwALHcHN6raxpTbR1eVgV2QyLA_4K8HrJO92et3qRXiXPkj7&redirect_uri=https://uscdi.epic.com/test/smart&client_id=d45049c3-3441-40ef-ab4d-b9cd86a17225
```

The authorization server responds to the HTTP POST request with a JSON object that includes an access token. The response contains the following parameters:

- `access_token`: This parameter contains the access token issued by Epic to your application and is used in future requests.
- `token_type`: In Epic's OAuth 2.0 implementation, this parameter always includes the value `bearer`.
- `expires_in`: This parameter contains the number of seconds for which the access token is valid.
- `scope`: This parameter describes the access your application is authorized for.
- `patient`: For patient-facing workflows, this parameter provides the DSTU2 FHIR ID for the patient in context at time of launch. The patient's FHIR ID is not returned for provider-facing standalone launch workflows.
- `state`: This parameter will have the same value as the earlier state parameter. For more information, refer to Step 3.

Note that you can pass additional parameters if needed based on the integration configuration. Here's an example of what a JSON object including an access token might look like:

```
{
  "access_token": "Nxfve4qJH9TKs5fsvf6kRYAZqzK7j9LHvrg1Bw7fU_07_FdV9aRzLCI1GxOn20LuO2Ah15RkRnz-p8u1MeYWqA85T8s4Ce3LcgQqIwsTkI7wezBsMduPw_xkVtLziUZO",
  "token_type": "bearer",
  "expires_in": 3240,
  "scope": "Patient.read Patient.search ",
  "patient": "TIw15bk8n1YVgvWk9D05BmRVRP13ECImNSKBDKyKltsMB",
  "state": "abc123",
}
```

At this point, authorization is complete and the web application can access the protected patient data it requested using FHIR APIs.

Step 4: Your Application Uses FHIR APIs to Access Patient Data

With a valid access token, your application can now access protected patient data from the EHR database using FHIR APIs. Queries must contain an Authorization header that includes the access token presented as a Bearer token.

Here's an example of what a valid query looks like:

```
https://uscdi.epic.com/interconnect-uscdi-oauth/api/FHIR/DSTU2/Patient/TIw15bk8n1YVgvWk9D05BmRVRP13ECImNSKBDKyKltsMB

Authorization: Bearer Nxfve4qJH9TKs5fsvf6kRYAZqzK7j9LHvrg1Bw7fU_07_FdV9aRzLCI1GxOn20LuO2Ah15RkRnz-p8u1MeYWqA85T8s4Ce3LcgQqIwsTkI7wezBsMduPw_xkVtLziUZO
```

Epic uses cookies to improve your experience on our website. By continuing to access uscdi.epic.com, you will receive all cookies from the Epic website. You may adjust your browser settings if you do not wish to receive cookies. Read our privacy policy here.

© 1979-2020 Epic Systems Corporation. All rights reserved. Protected by U.S. patents. For details visit www.epic.com/patents. HL7, FHIR® and the FLAME mark are the registered trademarks of HL7 and are used with the permission of HL7. The use of these trademarks does not constitute a product endorsement by HL7.