

A decorative grid of small blue dots in the top-left corner, arranged in a pattern that tapers to the right.

# Smile Guide: SMART on FHIR

Development and configuration of SMART on FHIR with Smile CDR

A decorative grid of small blue dots in the bottom-right corner, arranged in a rectangular pattern.

## Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>List of Figures</b>	<b>2</b>
<b>What to Expect</b>	<b>3</b>
<b>Background</b>	<b>3</b>
<b>Objective</b>	<b>3</b>
<b>Prerequisites</b>	<b>4</b>
<b>Configuring Smile CDR</b>	<b>5</b>
OIC Client Configuration / Creation of an OIC Client	5
Enable SMART on FHIR requests	7
<b>Creating SMART on FHIR WebApp.</b>	<b>10</b>
Creating the App	10
launch.html	11
index.html	11
package.json	12
Create Patient Resource	13
Create Test User With Launch Context.	14
Run and Test the App	16

### What to Expect

 *Reading time = 1 hour*

By the end of this guide you'll be able to:

- Configure Smile CDR to support a Smart on FHIR app on an EHR or resource server
- Create a simple SMART on FHIR app and
- Connect it to Smile CDR

### Background

SMART on FHIR is an open source standards-based API which enables innovators to develop an app once and have it run anywhere in the healthcare system. SMART on FHIR has defined a secure authorization method that allows health apps to connect and access protected information from EHR systems.

### Prerequisites

The following items/knowledge are required:

1. It's assumed that there's an understanding of the documentation below:
  - a. [SMILE CDR](#)
  - b. [SMART on FHIR](#) or [Smarthealthit](#)
  - c. [FHIR HL7](#)
2. Smile CDR is already installed. If it's not installed, please consult the link [here](#):
3. The following SMART app development prerequisites are available:
  - a. A local server to host the app. (For the purposes of this document, we'll use node-based [http-server](#). There's no need to install it separately.)
  - b. A basic understanding of JavaScript is beneficial to understand this app.
  - c. Nodejs installed. If it's not installed, please consult the link [here](#)

## Tools

The following tools/software will be needed to complete this guide.

- API Testing Platform (Insomnia, Postman or similar)
- Visual Studio Code or Notepad++ to edit HTML and JSON files
- Local server to host the app

## Configuring Smile CDR

To use Smile CDR as a launch platform for Smart on FHIR apps, we need to configure the following modules:

1. OIC Client Configuration
2. SMART Outbound Security Module
3. FHIR Endpoint Module

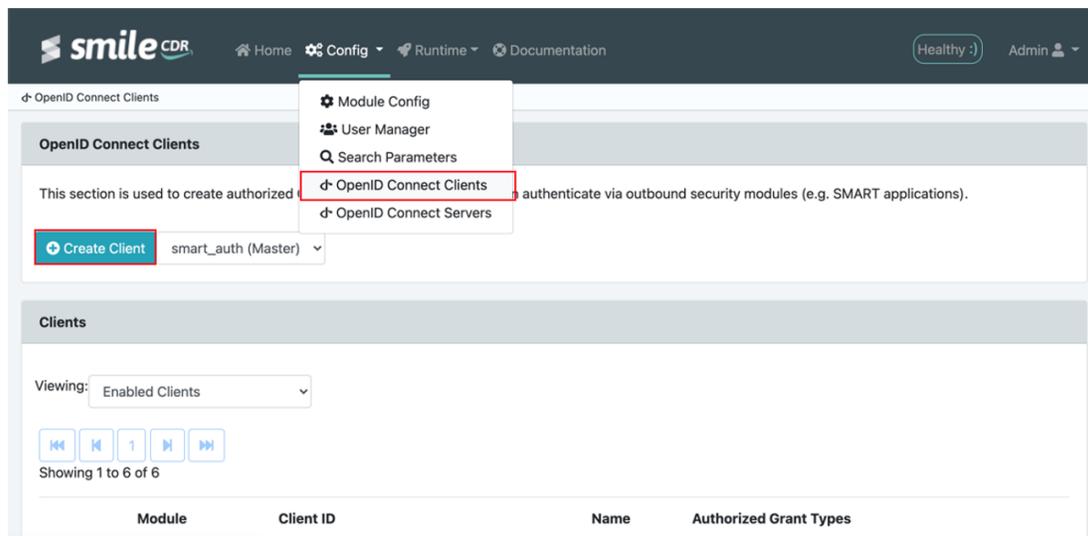
### OIC Client Configuration/Creation of an OIC Client

To support launching the SMART on FHIR applications using Smile CDR, we need to configure the *OpenId Connect Client* (OIC Client) module. [OpenId Client or OIC](#) is a simple identity layer built on top of the OAuth 2.0 protocol. It allows clients to verify the identity of the End-User based on the authentication performed by an Authorization Server as well as obtain basic profile information about the End-User in an interoperable and REST-like manner. To configure the OIC:

1. **Sign in** to the Web Admin Console of Smile CDR by typing in this link:

<http://localhost:9100>

2. On the top menu bar, **select** "Config," then **select** "OpenID Connect Client." You should see a page like this:



- To create a new client, **click** on “Create Client.”
- On the “Create Client” page **set the configurations** to the specifications below. (There are more configurations than given below; however, for this guide, only the configurations indicated below are required.)

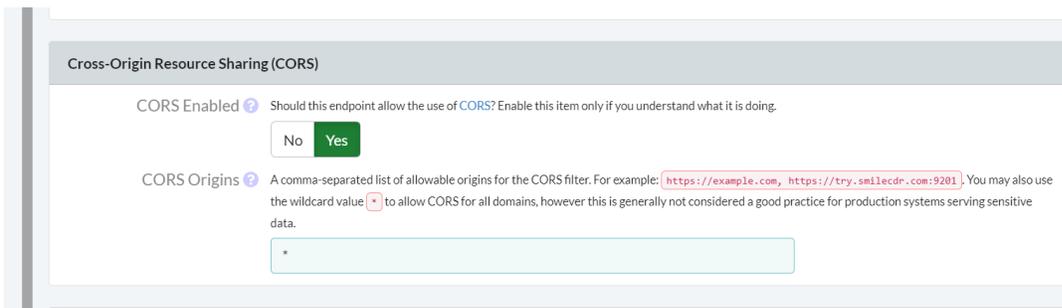
Configuration	Sample Value	Description
<b>Client ID</b>	patient_app_demo	This value needs to match the value of client_id being passed from the App while sending an authorization request
<b>Client Name</b>	SMART Patient	Name of the SMART App. This will be shown to the user while authorizing.
<b>Authorized Grant Types</b>	Authorization Code	Enable authorization types that the SoF app will support. For more information <a href="#">see here</a>
<b>Access Token Validity</b>	3600	Tokens requested by this client will be valid for the given time period.
<b>Refresh Token Validity</b>	86400	If <a href="#">refresh tokens</a> are enabled, any refresh tokens requested by this client will be valid for the given time period
<b>Authorized Redirect URLs</b>	http://127.0.0.1:9201/ http://127.0.0.1:9201/index.html	These are the URLs that the SoF app is allowed to use. Upon successful authorization, the user will be redirected to this URL.
<b>Scopes</b>	openid launch patient/*.read offline_access	A list of SMART scope (space separated) that client is permitted to request. <a href="#">Click here</a> to read more about scopes

- Click** “Create” and it should create an OIC client for you.
- The client list should display newly created OIC clients. For future changes to the clients configuration, **click** “Modify” and change any configurations as desired.

### Enable SMART on FHIR requests

We'll need to make a few more module configurations to allow the SMART on FHIR App to submit a FHIR request to Smile CDR.

1. From the top menu bar, **select** "Config," then select "Module Config."
2. We need to enable CORS for the SMART Authentication Module. To do this:
  - a. On the left pane, **go** to the "smart\_auth module."
  - b. **Scroll down** to the "Cross-Origin Resource Sharing (CORS)" section.
  - c. **Toggle** "CORS Enabled" to "YES."



- d. Scroll to the top of the page and then, **click** "Save" and then **click** "Restart."
3. Next, we need to enable the SMART Authentication for the FHIR Endpoint Module.
    - a. **Select** the "fhir\_endpoint" module from the left pane.
    - b. **Scroll down** to the "Dependencies" section.
    - c. For the "OpenID Connect Authentication" config, **select** "smart\_auth" from the drop-down menu.

**Dependencies**

FHIR R4 Storage	The FHIR R4 Storage engine to use as a manager for this module.
	<div style="border: 1px solid #ccc; padding: 2px; display: flex; justify-content: space-between; align-items: center;"> <span>persistence (FHIR Storage (R4 Relational))</span> <span>▼</span> </div>
Username/Password Authentication	The inbound security module to use for authenticating and authorizing users to this module where authentication requires a username and password.
	<div style="border: 1px solid #ccc; padding: 2px; display: flex; justify-content: space-between; align-items: center;"> <span>local_security (Local Inbound Security)</span> <span>▼</span> </div>
OpenID Connect Authentication	The inbound security module (or outbound security module if using internal access tokens) to use for authenticating and authorizing users to this module using OpenID Connect Authentication.
	<div style="border: 1px solid #ccc; padding: 2px; display: flex; justify-content: space-between; align-items: center; background-color: #ffff00;"> <span>smart_auth (SMART Outbound Security)</span> <span>▼</span> </div>
Websocket Subscription Endpoint	Add this to include the websocket URL in your server's FHIR CapabilityStatement
	<div style="border: 1px solid #ccc; padding: 2px; display: flex; justify-content: space-between; align-items: center;"> <span></span> <span>▼</span> </div>
Validation Support	This dependency supplies validation artifacts (StructureDefinitions, ValueSets, etc.) and provides terminology services used by the validator.

- d. On the left pane, **scroll down** to *"Auth: OpenID Connect."*
- e. **Toggle** *"OpenID Connect Security"* to **"YES"**.

**Auth: OpenID Connect**

OpenID Connect Security Should this endpoint support the use of OpenID Connect Authentication (e.g. SMART)?

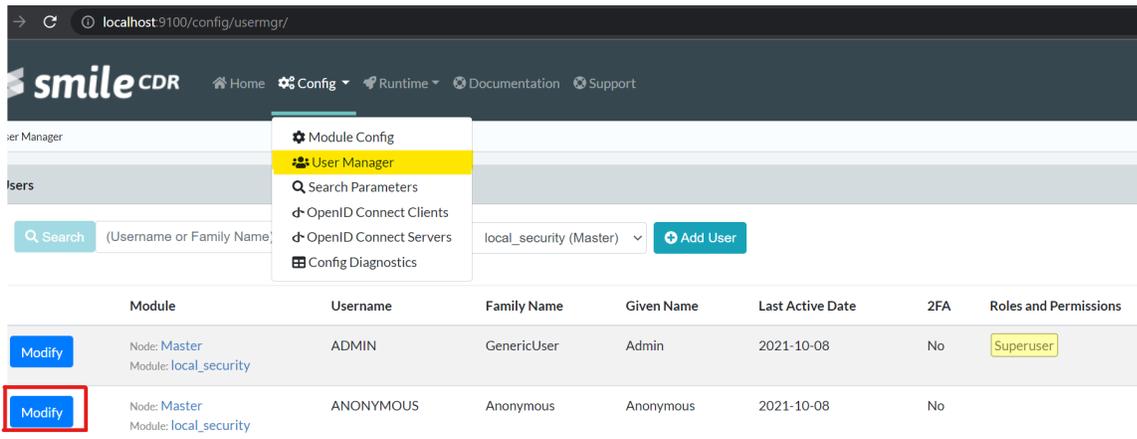
?

No

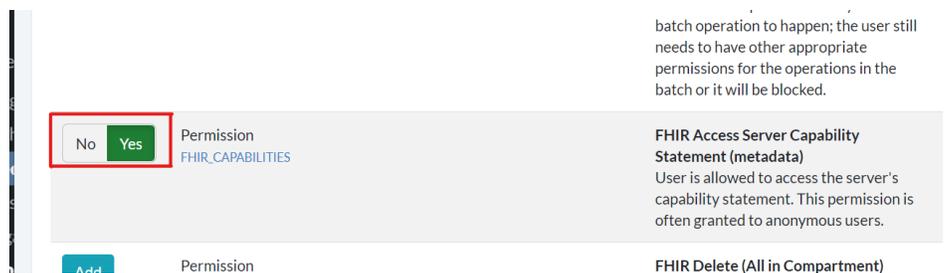
Yes

- f. **Click** *"Save,"* then *"Restart."*

4. Now we need to enable the Anonymous Access to Capability Statement. When doing this, any requests that don't supply credentials will be granted the authorities of the designated Anonymous user. By default, this is a user with the username "ANONYMOUS," but can be modified using the [Anonymous Account Username](#) setting:
  - a. From the main menu on top, **select** *"Config,"* then *"User Manager."*
  - b. Look for the ANONYMOUS user from the list and **select** *"Modify."*



- c. From the “Roles and Permissions” section, **scroll down** to “FHIR\_CAPABILITIES” and **click** on “YES” to enable the permission.



- d. **Click “Save”** the user from the top of the page.

At this stage, we have Smile CDR configured to handle SMART on FHIR app requests. In the next section, we’ll create a SMART on FHIR app.

### Creating SMART on FHIR WebApp.

In this section we'll create a web app for our SMART on FHIR demo app.

To complete this section, you'll need:

- An API platform like Postman or Insomnia to send and receive data,
- A text editor to create html and JSONfiles

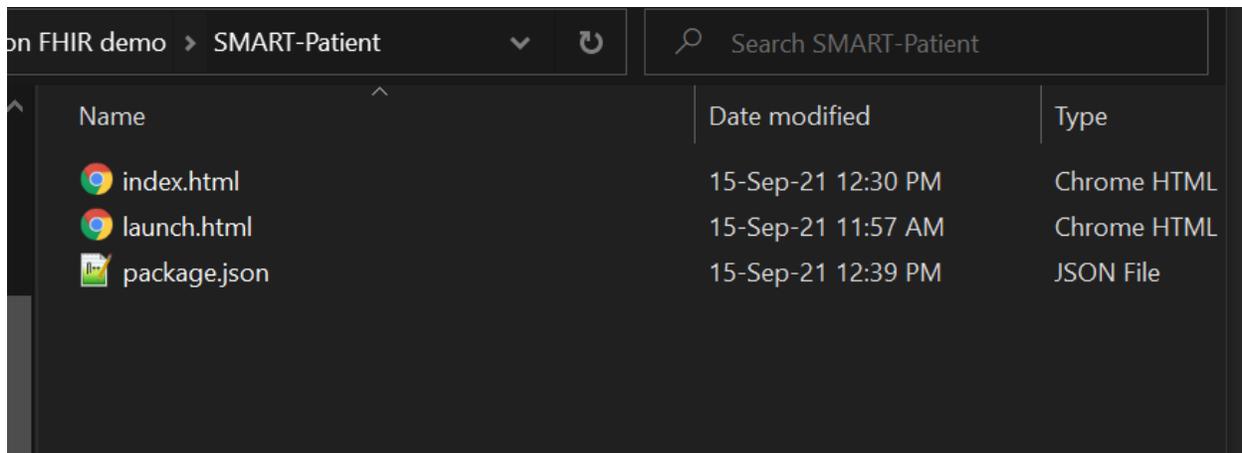
### Creating the App

We'll use the `fhir-client.js` library to connect our SMART app to the FHIR server. Additional documentation on `fhir-client.js` can be found [here](#).

We'll need to create a new directory and three files; two `.html` files and a `.json` file.

To begin:

1. **Create** a project directory called "SMART-Patient" for this SMART app project
2. **Open** the project directory
3. **Create** two `.html` files named `index.html` and `launch.html`
4. **Create** a `.json` file with the name `package.json` for the Node.js project configuration



## launch.html

launch.html is the SMART app's initial entry point and in a real production environment would be invoked by the application (i.e. App Gallery/App Sphere) launching your SMART app. This page typically initiates authorization flow which means it will take parameters from the URL and send the user to the authentication screen.

To begin:

1. **Insert** the code below in your "launch.html" file, then save and close the file.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8" />
  <title>SMART: Patient App</title>
  <script src="https://cdn.jsdelivr.net/npm/fhirclient/build/fhir-client.js"></script>
</head>
<body>
<script>
  FHIR.oauth2.authorize({
    // this value should match with the value in the OIC Client config
    clientId: "patient_app_demo",
    // this list should not include any scope that is
    // not mentioned in the Client configuration
    scope: "openid launch/patient patient/*.read offline_access",
    // (Optional) many servers are capable of using index.html by default
    redirectUri: 'index.html',
  });
</script>
</body>
</html>
```

## index.html

After the SMART authorization procedure, the authorization server will redirect users to this page. When this page is invoked, the SMART app will have the authorization to access the FHIR server.

To begin:

---

1. **Insert** the code below to your “index.html” file, then save and close the file. Note: **replace** the patient id (patient-a) in JavaScript code (line no. 19) with one of the existing patients’ ID, or create a new patient with this ID. The next section of this guide explains how to create a patient resource.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8"/>
  <title>Example SMART App</title>
  <!-- import fhir-client library-->
  <script src="https://cdn.jsdelivr.net/npm/fhirclient/build/fhir-client.js"></script>
</head>
<body>
<h2 id="patient_header"> Patient Details</h2>
<h4>Patient Information:</h4>
<pre id="info">Loading...</pre>
<script type="text/javascript">
  // this function generated name from HumanName object from Patient resource
  function displayName (name) {
    return (name.prefix || '') + name.given.join(' ') + ' ' + name.family
  }
  // Request current logged in Patient Data using fhir-client library and updates DOM on
successful response
  FHIR.oauth2.ready().then(async (client) => {
    return client.patient.read()
  }).then(
    function (pt) {
      document.getElementById('patient_header').innerText = displayName(pt.name[0])
      document.getElementById('info').innerText = JSON.stringify(pt, null, '\t')
    },
    function (error) {
      document.getElementById('patient_header').innerText = 'Error Occurred'
      document.getElementById('info').innerText = error.stack
    },
  ).catch(console.error)
</script>
</body>
</html>
```

### package.json

The objective of the package.json script is to record the metadata of the project. This file includes any dependencies needed to run the Node.js project. Here it just has one example of a dependency; an [http-server](#)—a package that creates a static HTTP server on your computer.

To begin:

1. **Insert** the code below to the “package.json” file, then save and close the file.

```
{
  "name": "smart_patient_app",
  "version": "1.0.0",
  "scripts": {
    "serve": "http-server -p 9201 -c-1",
    "start": "npm run serve"
  },
  "dependencies": {
  },
  "devDependencies": {
    "http-server": "^0.12.3"
  }
}
```

## Create Patient Resource

To create a Patient resource with id “patient-a” perform the following.

1. Complete the HTTP operation using either Postman or Insomnia:

URL: **http://localhost:8000/Patient/patient-a**

(this URL is for default settings. If you have a custom setting, change this URL to reflect those settings)

Method: **PUT**

Header: **Authorization: Basic YWRtaW46cGFzc3dvcmQ=**

(this creates the credentials admin/password)

Header: **Content-Type: application/fhir+json**

Body: (feel free to change name or birthdate, or to add additional patient details as you wish)

```
{
  "resourceType": "Patient",
  "id": "patient-a",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2020-11-10T20:24:48.194+00:00"
  }
}
```

```

    },
    "name": [ {
      "family": "Smith",
      "given": [ "John" ]
    } ],
    "gender": "male",
    "birthDate": "2020-01-01"
  }
}

```

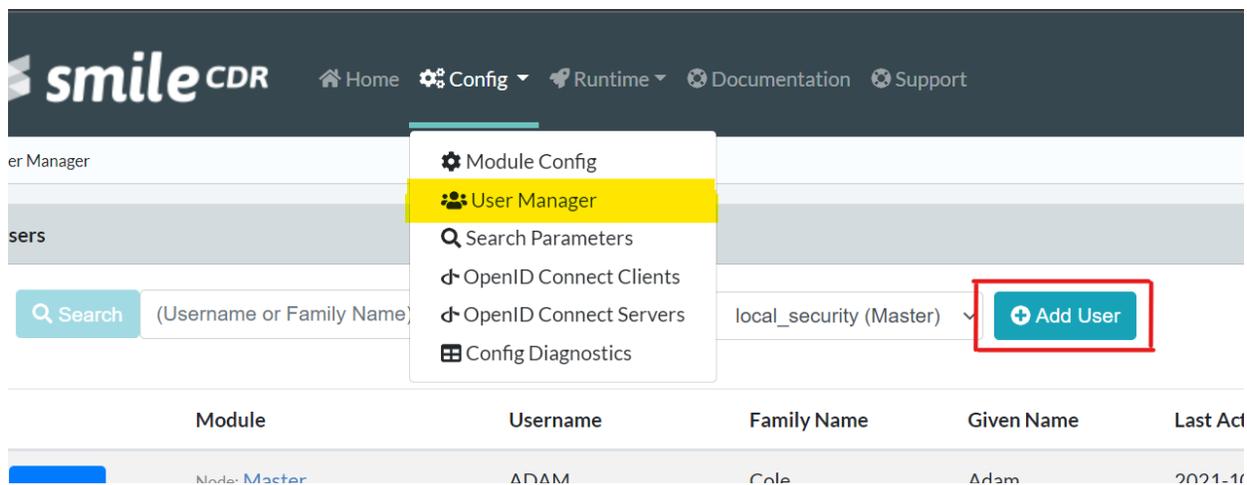
2. **Click** on *“Send”* (from Insomnia or Postman to send the request). The response should contain the resource data that we just created. If you want to learn more about different FHIR REST operations, please checkout [this guide](#).

### Create Test User with Launch Context.

In order to test the SMART on FHIR WebApp, we'll need to create a user with appropriate permissions and launch contexts.

To create a user:

1. **Go** to the *“Web Admin Console”* and **log in**.
2. **Click** on *“Config”* on the top menu bar, then *“User Manager.”*
3. **Click** on *“Add User.”*



4. **Fill** the necessary fields as indicated below:

- a. *Demographics*: set up the username (required) , name and email of the user in this section.

### Demographics

* Username	<input type="text" value="John"/>
Family Name	<input type="text" value="Doe"/>
Given Name	<input type="text" value="Jhon"/>
Email Address	<input type="text" value="jhon@smilecdr.com"/>

- b. *Security*: set the password for the user.

### Security

* Password	<input type="password" value="....."/>
Locked	A locked account is not capable of being used (functionally it is idk <input type="radio"/> No <input type="radio"/> Yes

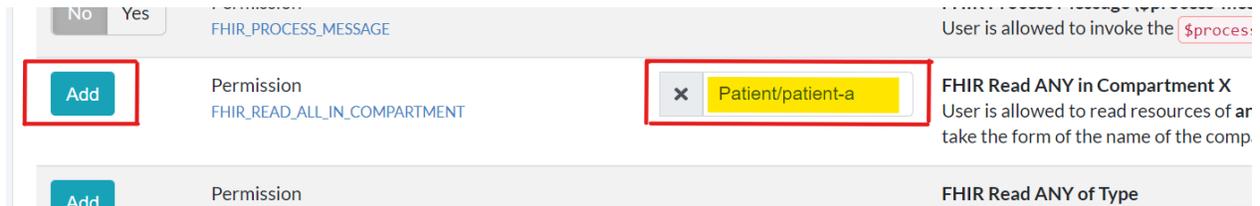
- c. *Default Launch Context*: set the contexts associated with a user. These context(s) will be added to SMART auth session, depending on the launch scopes requested(*i.e.* launch, launch/patient, launch/location etc..) In other words, these context(s) associate the user account with the default resources IDs.

### Default Launch Contexts

Default launch contexts are contexts associated with a user by default for the purpose of supplying a *launch context* claim

Patient	<input type="text" value="patient-a"/>
Encounter	<input type="text"/>
Location	<input type="text"/>

- d. *Roles and Permissions*: scroll down to *FHIR\_READ\_ALL\_IN\_COMPARTMENT* and **select "Add."** This will allow access to the resources of its own compartment and the user to access any resources related to a given patient. Fill the text box with resource ID: "*Patient/patient-a*" as below.



No	Yes	Permission	User is allowed to invoke the
		FHIR_PROCESS_MESSAGE	User is allowed to invoke the <code>\$proces:</code>
		Add Permission FHIR_READ_ALL_IN_COMPARTMENT	Patient/patient-a FHIR Read ANY in Compartment X User is allowed to read resources of ar take the form of the name of the comp
		Add Permission	FHIR Read ANY of Type

5. Scroll to the top of the page and **select "Save"** to create the user.

### Run and Test the App

Now that the Smart Web APP is created and Smile CDR is configured, we need to make sure that everything is functioning correctly.

1. **Open** the "*terminal/command*" prompt.
2. **Navigate** to the "*project directory*," then **run** the following command:

```
cd <path-to-project-dir>
```

Note: replace the `<path-to-project-dir>` with the actual path to project directory. i.e.  
D://SMART-Patient

3. **Run** the following command:

```
npm install
```

Note: this will install required dependencies for a node project listed in the package.json file. (You need to run this command only once unless you make any changes in the package.json file.)

4. **Run** the command to start the server:

```
npm start
```

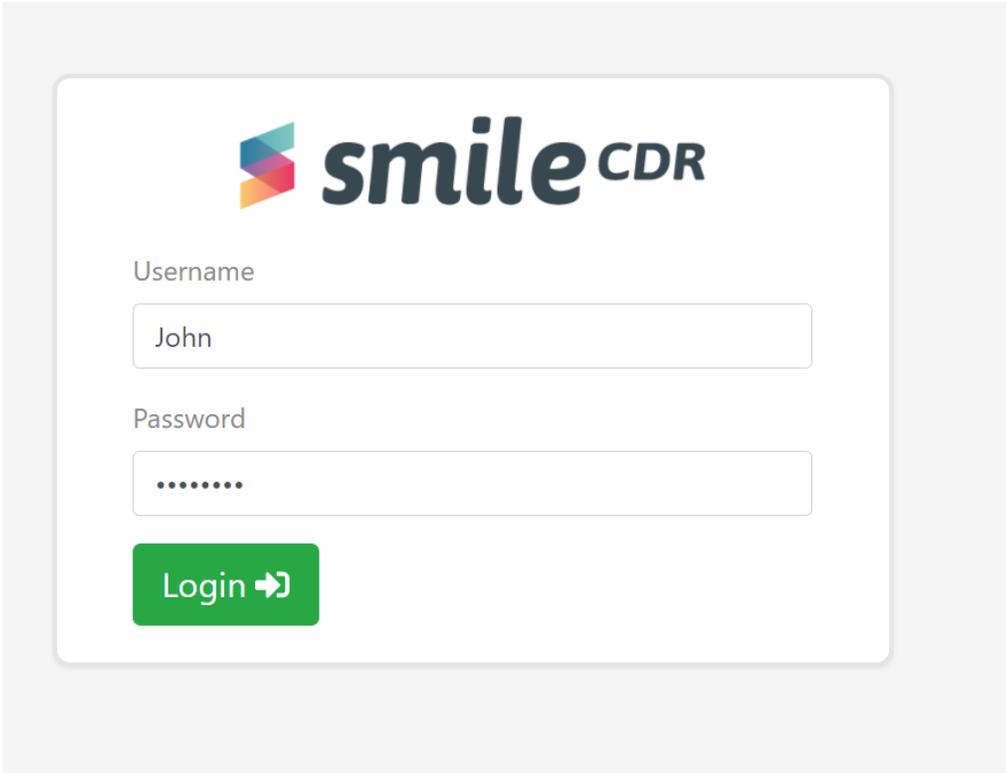
5. **Copy** and **paste** the following URL into the address bar:

<http://127.0.0.1:9201/launch.html?iss=http://localhost:8000&launch=A000>

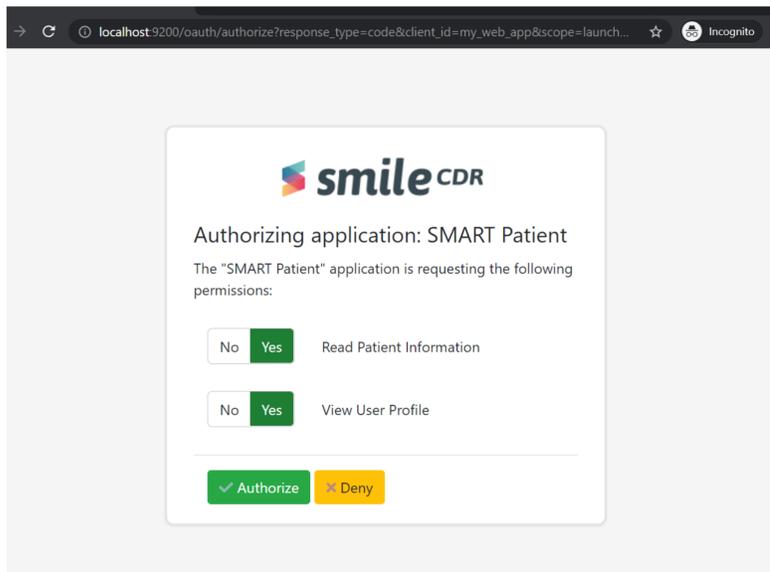
*iss*: The base URL for the FHIR endpoint. The app will load the server capability statement from this endpoint which allows it to figure out where to authorize.

*launch*: This is intended to be a one-time nonce. In a real scenario, this would be randomly generated.

6. After launching, an authentication page similar to the image below will appear:

The image shows a login form for Smile CDR. At the top is the Smile CDR logo. Below it are two input fields: 'Username' with the text 'John' and 'Password' with a masked password of seven dots. A green 'Login' button with a right-pointing arrow is positioned below the password field.

7. **Enter** the credentials then **select** "Login,"(Use the credential of the user created in the previous section.) The authorization screen should appear like the image below:
  - a. **Select** "Authorize"



- The authorization server will redirect the user to index.html or root "/" whichever was requested from launch.html

## John Smith

### Patient Information:

```
{
  "resourceType": "Patient",
  "id": "patient-a",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2021-12-07T18:11:37.533+00:00",
    "source": "#6yA3dLe4hnRBtQpe"
  },
  "name": [
    {
      "family": "Smith",
      "given": [
        "John"
      ]
    }
  ],
  "gender": "male",
  "birthDate": "2020-01-01"
}
```

Congrats, you've now successfully created and tested your first SMART on FHIR app with Smile CDR!

### Registering an App with the appSphere.

For the purpose of this guide, we'll use a locally installed App Gallery from Smile CDR. To use a local instance of Smile CDR, you need the App Gallery/App Management Tool module configured in Smile CDR. Check out [this document](#) to learn how to configure it locally.

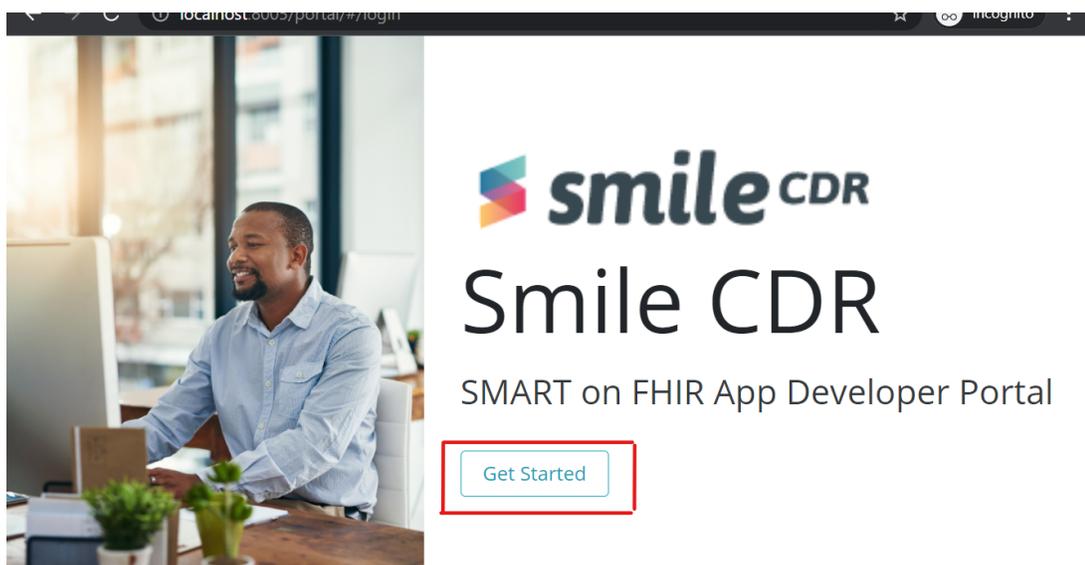
To register the app:

1. **Go to** the App Gallery portals's homepage, then **click** on the *"Developer Portal"*

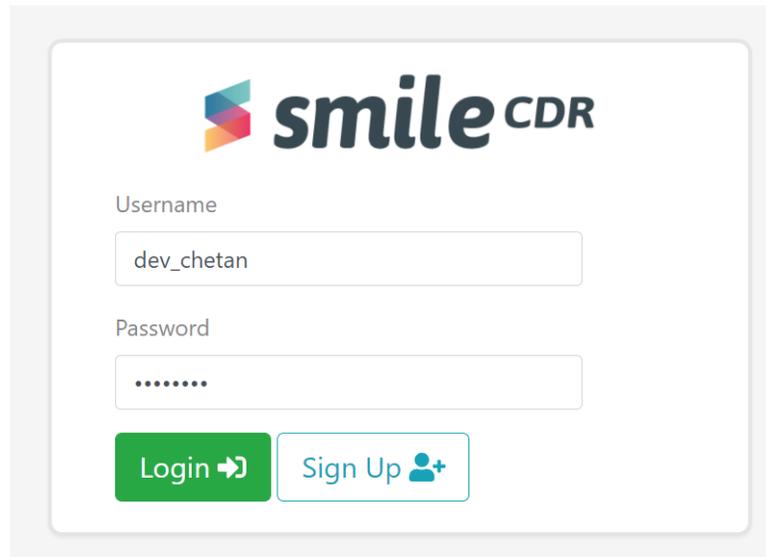


**Note:** Admin Console and Developer Portal require different user roles and running them si

2. The developer homepage will appear. **Select** *"Get Started"* button



3. This will take you to the authentication screen below:



4. **Enter** your developer credentials and **click** “Login” (if you do not have an account, **create one** by clicking “Sign Up”). Upon successful login, you should see the developer dashboard (below). **Select** “Register App.”

smile CDR SMART on FHIR App Developer Portal 🔍

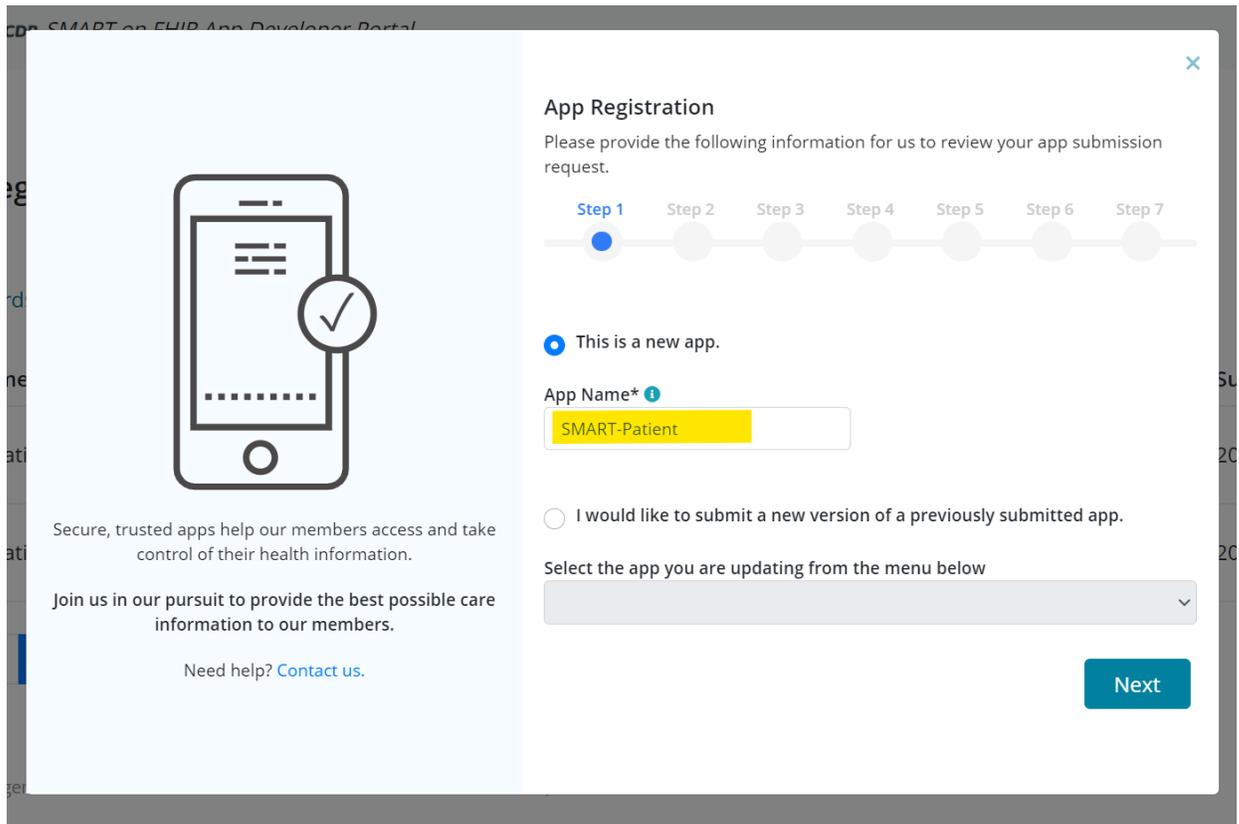
**My Registrations** + Register App Click to register new apps or re-register new versions of apps.

2 records  🔍

App Name	Status	Change	Message	Modified	Version
 Patient Lookup	Live	In Review → Live	Promote to Live 	2021.10.15	2

5. This opens a popup for the app registration process. **Proceed** with the steps below:

**Step 1:** Provide the app name as I—it will be displayed on the App Gallery.



**App Registration**

Please provide the following information for us to review your app submission request.

Step 1 Step 2 Step 3 Step 4 Step 5 Step 6 Step 7

This is a new app.

App Name\* ?

SMART-Patient

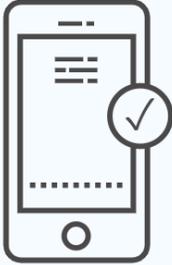
I would like to submit a new version of a previously submitted app.

Select the app you are updating from the menu below

Next

**Step 2:** Select the operating system (i.e., web, iOS or Android) for which the app is available and to be published in the public-facing site. **Fill** in the other information as shown below:

- App Homepage URL:* the URL where the app's download sites can be found. (Note: provide a homepage URL if no specific app page exists.)
- URL to the App's Privacy Policy:* URL of a webpage providing the app's Privacy Policy.
- URL to the App's Terms of Service:* URL of a webpage describing the app's Terms of Service.
- OAuth Redirect URL:* URL to which developers are redirected upon successful authentication.
- Web App Launch URL:* URL used to start the authentication process for web apps only.



Secure, trusted apps help our members access and take control of their health information.

Join us in our pursuit to provide the best possible care information to our members.

Need help? [Contact us.](#)

✕

### App Registration

Please provide the following information for us to review your app submission request.

Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7

**Supported Operating Systems**

OS Supported \*

Web

iOS

Android

**App Homepage URL\* <sup>?</sup>**

**URL to the App's Privacy Policy\* <sup>?</sup>**      **URL to the App's Terms of Service\* <sup>?</sup>**

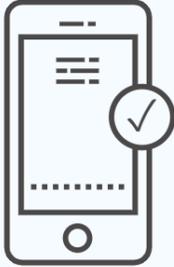
**Default OAuth Redirect URL\* <sup>?</sup>**

**Additional OAuth Redirect URLs (One URL per line) <sup>?</sup>**

**Web App Launch URL\* <sup>?</sup>**

**Step 3:** Provide an app description for the public-facing site.

- a. **Upload** an app icon: use the guidelines from the Google Play store (link provided) to upload an app icon of the acceptable specifications. An option to preview the uploaded image is provided.
- b. **Add** a short app description: this should be between 20-150 characters for the public-facing site.
- c. **Add** a long app description: this should be between 200-1000 characters for the public-facing site.



Secure, trusted apps help our members access and take control of their health information.

Join us in our pursuit to provide the best possible care information to our members.

Need help? [Contact us.](#)

×

### App Registration

Please provide the following information for us to review your app submission request.

Step 1 ● Step 2 ● Step 3 ● Step 4 ● Step 5 ● Step 6 ● Step 7 ●

App Descriptions ⓘ

Upload App Icon\* ⓘ

1635517110112.png (image/png)

Preview  


**Short App Description (20-150 characters) \***

SMART on FHIR app to allow user to view their full Patient profile

**Long App Description (200-1000 characters) \***

SMART on FHIR app to allow user to view their full Patient profile. This is Test Description. This is Test Description.

Back

Next

**Step 4:** Select all applicable categories from the given options.

- a. *Audience Category:* options include payer, provider, pharma, patient and developer.
- b. *App Use Category:* options include Health & Therapy Management, Provider Care, Coordination, Clinical Applications, Research and Data Monitoring Analysis.
- c. *FHIR Version Supported:* options include DSTU1, DSTU2, STU3 and R4.
- d. *Privacy & Security Compliance:* options include HIPPA, GDPR, CARIN Code of Conduct and ONC Model Privacy Notice (note: users may be asked to provide supporting documents).
- e. *Confidentiality:* if the app runs in an execution environment that enables the app to protect confidential information, leave it as “confidential;” if not, toggle to “public.”



Secure, trusted apps help our members access and take control of their health information.

Join us in our pursuit to provide the best possible care information to our members.

Need help? [Contact us.](#)

✕

### App Registration

Please provide the following information for us to review your app submission request.

Step 1   Step 2   Step 3   **Step 4**   Step 5   Step 6   Step 7



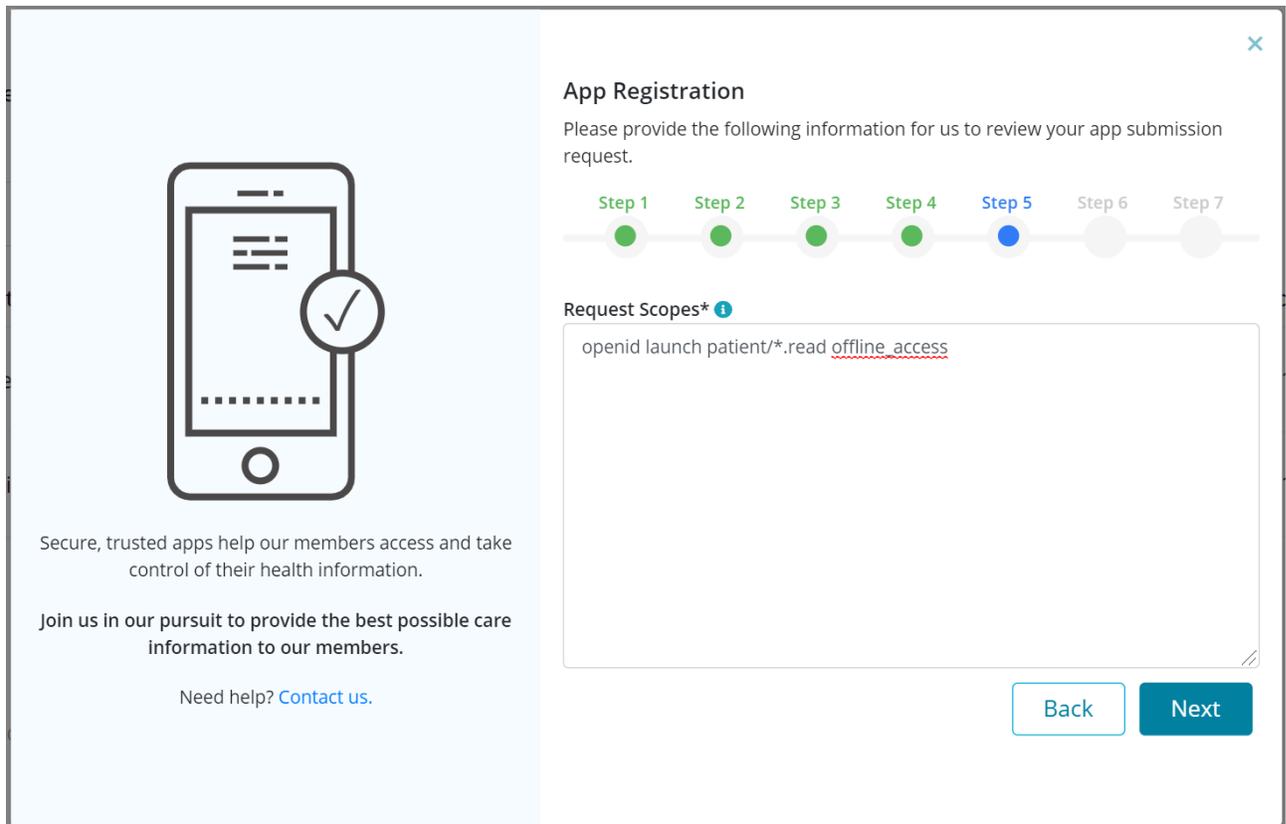
For the following categories, select all that are applicable: ⓘ

<p><b>Audience Category *</b></p> <p><input type="checkbox"/> Payer</p> <p><input type="checkbox"/> Provider</p> <p><input type="checkbox"/> Pharma</p> <p><input checked="" type="checkbox"/> Patient</p> <p><input type="checkbox"/> Developer</p>	<p><b>App Use Category *</b></p> <p><input type="checkbox"/> Health and Therapy Management</p> <p><input type="checkbox"/> Provider Care Coordination</p> <p><input checked="" type="checkbox"/> Clinical Applications</p> <p><input type="checkbox"/> Research</p> <p><input type="checkbox"/> Data Monitoring Analysis</p>
<p><b>FHIR Version Supported *</b></p> <p><input type="checkbox"/> DSTU 1</p> <p><input type="checkbox"/> DSTU 2</p> <p><input type="checkbox"/> STU 3</p> <p><input checked="" type="checkbox"/> R4</p>	<p><b>Privacy and Security Compliance</b></p> <p><input type="checkbox"/> HIPPA ⓘ</p> <p><input type="checkbox"/> GDPR ⓘ</p> <p><input type="checkbox"/> CARIN Code of Conduct ⓘ</p> <p><input type="checkbox"/> ONC Model Privacy Notice ⓘ</p>

**Confidentiality\* ⓘ**  
 Public  Confidential

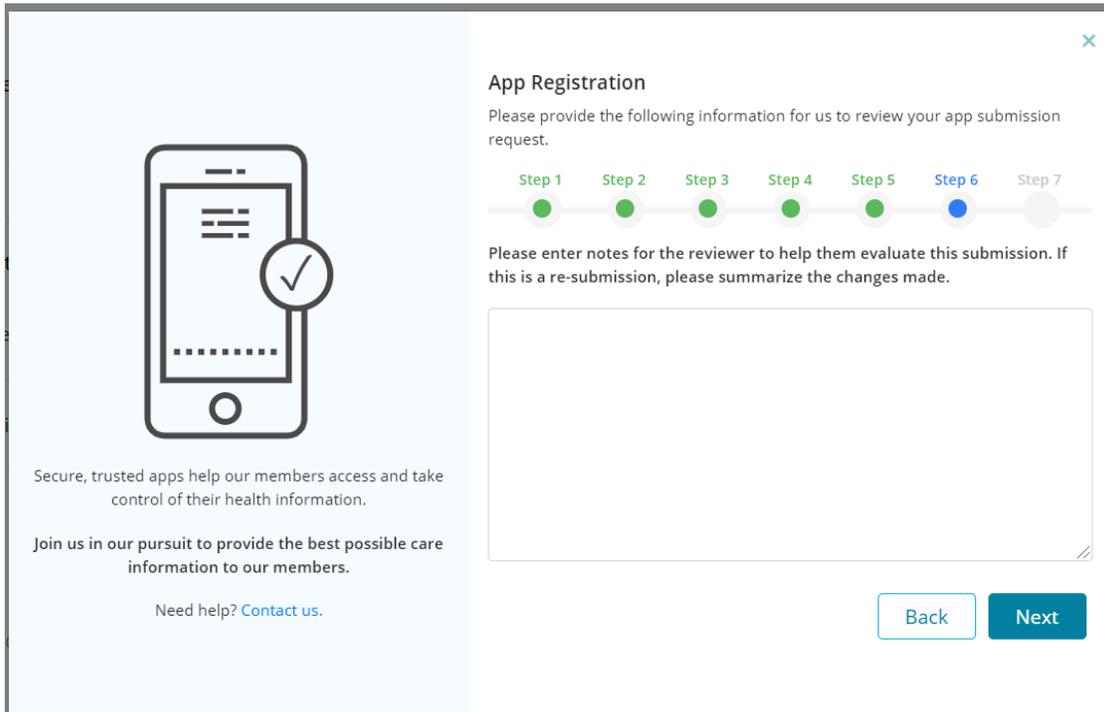
Back Next

**Step 5:** Enter any scopes that will be requested from the app from the authorization process. This field is similar to the scopes configured in the [OIDC client configuration](#).



The screenshot shows the 'App Registration' interface. On the left, there is a graphic of a smartphone with a checkmark, accompanied by the text: 'Secure, trusted apps help our members access and take control of their health information. Join us in our pursuit to provide the best possible care information to our members. Need help? [Contact us](#).' On the right, the 'App Registration' section has a sub-header 'Please provide the following information for us to review your app submission request.' Below this is a progress bar with seven steps; Step 5 is the active step, highlighted in blue. Underneath the progress bar is the 'Request Scopes\*' field, which contains the text 'openid launch patient/\*.read offline\_access'. At the bottom right of the form are 'Back' and 'Next' buttons.

**Step 6 (Optional):** Enter details about the app for reviewer evaluation purpose. These will not be shown to users and are only for the reviewer to verify. If you're just updating the app and not submitting a new version, summarize the changes made.



**App Registration**

Please provide the following information for us to review your app submission request.

Step 1 Step 2 Step 3 Step 4 Step 5 Step 6 Step 7

Please enter notes for the reviewer to help them evaluate this submission. If this is a re-submission, please summarize the changes made.

Secure, trusted apps help our members access and take control of their health information.

Join us in our pursuit to provide the best possible care information to our members.

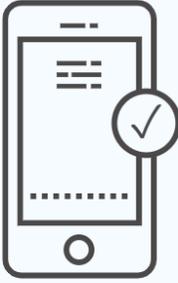
Need help? [Contact us.](#)

Back Next

**Step 7:** Carefully review the legal attestation and either accept or decline the terms that describe the minimum privacy and security criteria to sufficiently protect patients' protected health information in accordance with the CMS and ONC.

If you decline, the app will still be allowed to submit and can be approved, too. However, users will be warned that the app does not attest to the CMS ruling.

Submit the app for review.



Secure, trusted apps help our members access and take control of their health information.

Join us in our pursuit to provide the best possible care information to our members.

Need help? [Contact us.](#)

×

### App Registration

Please provide the following information for us to review your app submission request.

Step 1 Step 2 Step 3 Step 4 Step 5 Step 6 Step 7

**Legal Attestation\*** ⓘ

#### Legal Attestation

To assume that individuals are properly informed of their rights and risks in connection with the disclosure of their personal health information, this Attestation meets the standards described by the United States Centers for Medicare & Medicaid Services (CMS) Interoperability and Patient Access final rule (CMS-9115-F) in the Federal Register. To provide appropriate information to our members, *Smile CDR* is requesting that all Application Developers and Vendors (hereinafter referred to as

Accept     Decline

"I agree to all terms of the attestation." This attestation is legally binding.

Back
Submit

The app is now submitted for review and should be listed on the developer dashboard as shown below. An "In Review" status means it has been submitted to admin for review. The status will change once the admin either approves or rejects it. Results will be displayed as either "Live" or "Rejected."

## My Registrations

+ Register App
Click to register new apps or re-register new versions of apps.

3 records  Q

App Name	Status	Change	Message	Modified	Version	Submitted
SMART-Patient	In Review	--	--	2021.10.29	1	2021.10.29
Patient Lookup	Live	In Review → Live	Promote to Live	2021.10.15	2	2021.10.15
Patient Lookup	Retired	In Review → Live	Promote to Live	2021.10.15	1	2021.10.15

««
«
1
»
»»

## My Registrations

[+ Register App](#)

Click to register new apps or re-register new versions of apps.

3 records



App Name	Status	Change	Message	Modified	Version	Submitted
 SMART-Patient	Live	In Review → Live	Promote to Live 	2021.10.29	1	2021.10.29
 Patient Lookup	Live	In Review → Live	Promote to Live 	2021.10.15	2	2021.10.15
 Patient Lookup	Retired	In Review → Live	Promote to Live 	2021.10.15	1	2021.10.15

«« < 1 > »»

Should you run into trouble, contact [your Smile CDR Account Rep.](#)



## Smile CDR Inc.

622 College Street, Suite 401  
Toronto, Ontario M6G 1B4, Canada  
info@smilecdr.com  
1 (800) 683-1318  
www.smilecdr.com

Copyright ©2021 Smile CDR Inc.

*All product names, logos, and brands are the property of their respective owners. All company, product and service names used are for identification purposes only. The use of these names, logos, and brands does not imply endorsement.*

Version: 1.0

Last Updated: November 12, 2021

Principle Author: Chetan Shankar

