

記事

[Shintaro Kaminaka](#) · 2020年9月4日 13m read

## IRIS for Health 上でFHIR リポジトリ + OAuth2 認可サーバ/リソースサーバ構成を構築する パート1

開発者の皆さん、こんにちは。

今回の記事ではFHIRと組み合わせて使用されるケースが増えてきている、権限の認可 (Authorization) を行うためのOAuth2について取り上げます。

まずこのパート1では、IRIS for HealthおよびApacheのDockerコンテナの起動と、IRIS for Health上で、OAuth2 認可サーバ機能を構成し、REST開発ツールPostmanからアクセスし、アクセストークンを取得する方法について解説します。

さらにパート2以降では、IRIS for HealthにFHIRリポジトリ機能を追加し、OAuth2リソースサーバ構成を追加して、Postmanからアクセストークンを使用したFHIRリクエストの実行方法まで解説します。

InterSystems製品のOAuth2機能の解説については、すでいくつかの記事が開発者コミュニティ上に公開されていますが、今回は改めて最新バージョンでの構成方法を解説したいと思います。

[InterSystems IRIS Open Authorization Framework \( OAuth 2.0 \) の実装 - パート1](#)

今回の記事では、最新のInterSystems IRIS for Health 2020.3 Preview Editionを使用します。実際にこの記事元にして、環境を構築してみようと思われた方は必ずこのバージョン以降のキットを使用してください。一部機能がこのバージョン以前の製品には含まれていません。

### 事前準備

まずは事前準備です。セキュアな環境を構築するために、色々と準備することが多いです。

IRIS for Health 2020.3 Preview EditionはDockerコンテナバージョンだけが提供されています。([InterSystems Docker Hub/IRIS for Health](#))

また、OAuth2構成を行うためには、WebサーバおよびSSL構成を行う必要があります。今回はApacheを使用します。

Apache上でSSL構成を行う場合、SSL構成の証明書は正しくサーバのホスト名と一致している必要があります。この点を注意してください。

### intersystems-jp GitHub リポジトリからのサンプルファイルの取得

インターシステムズ開発者コミュニティ専用のGitHubリポジトリに今回の構成で使用するサンプルの docker-compose.yml/Dockerfile等が公開されています。

まずは以下のコマンドでご利用の環境にこのファイルを展開してください。(この記事の添付ファイルにも追加されているのでそちらからでもOKです。)

このdocker-

compose.yml/Dockerfile等のファイル

は、OpenExchangeに公開されている[iris-webgateway-exampleアプリケーション](#)を参考に作成しています。

```
git clone https://github.com/InterSystems-jp/IRIS4H-OAuth2-handson.git
```

## 使用するキットに合わせた構成の変更

このdocker-compose.ymlファイルでは、IRIS for Healthのコンテナと、docker buildコマンドでbuildされるApache(httpd)のコンテナの二つが起動するように構成されています。GitHubで配布しているdocker-compose.ymlファイルでは、IRIS for Health Community Edition Preview Edition(2020.3.200.0)を指定しています。Community Editionはインターシステムズ製品の評価のために利用していただくことができます。

```
iris:
  image: store/intersystems/irishealth-community:2020.3.0.200.0
```

異なるバージョン（正式リリースバージョンやより新しいバージョン）を使用する場合は、この部分の指定を変更してください。

### Apache

の方のコンテナ

はDockerfileの内容でbuildされ

ますが、その際にApacheからIRISに接続するための[WebGateway](#)のキットが必要になります。

キットの入手方法についてはインターシステムズパートナーの方々はWRCのキットダウンロードサイトするか、またはWRCサポートセンターへお問い合わせください。

それ以外の方々はお手数ですが、[こちらのアドレス](#)までお問い合わせください。

入手した製品に合わせてDockerfileの以下の箇所を変更します。ホストマシンのOSがWindows/Ubuntu/CentOSいずれであっても、ベースとなるhttpdのコンテナOSがDebianであるため、platformはInxubuntux64の指定となります。

```
ARG version=2020.3.0.200.0
ARG platform=inxubuntux64
ADD WebGateway-${version}-${platform}.tar.gz /tmp/
```

## SSL証明書の用意

次にSSL証明書を用意します。OAuth2 認可アクセス時には、そのWebサーバで設定されているSSL証明書がアクセスしているURLと一致しているか確認を行います。

公式な証明書でなくても、openssl等を使用した証明書作成で問題はありません。証明書作成時にCommon Nameにホスト名を入力してください。

また、作成した証明書は起動時に自動で読み込むため、パスフレーズが不要なファイルに変更しておく必要があります。以下のコマンドを参考にしてください。

```
$ openssl rsa -in cert.key.org -out cert.key
```

作成されたCRTファイルおよび、KEYファイルをそれぞれ server.crt / server.key というファイル名で、Dockerfileと同じディレクトリにおいてください。

また、Apache Webサーバで利用する以外にも、OAuth2の設定でSSL証明書が必要になります。これらはホスト名の入力等は必要ありませんが、3セット作成しておいてください。（以降の設定ではauth.cer/auth.key , client.cer/client.key , resserver.cer/resserver.key として登場します。 ）

## dockerビルド と dockerコンテナの起動

これでようやく準備完了です！今ディレクトリには、DLした4つのファイル以外に、Webゲートウェイのインストール

ールキット、そして二つのSSL証明書がある状態です。各ファイルのアクセス・実行権限等にもご注意ください。  
(例えば私は、webgateway-entrypoint.shに実行許可を追加しました。)

```
docker-compose build
docker-compose up -d
```

起動したら、`docker ps` コマンドで二つのコンテナが起動していることを確認します。  
ApacheのコンテナIMAGE名 : <directoryname>web  
IRIS for Healthのコンテナ : store/intersystems/irishealth-community:2020.3.0.200.0 (あるいはそれぞれ指定したIMAGE)

次は、以下の3つの方法で管理ポータルにアクセスしてみてください。最後の3番目の方法でうまくアクセスできれば、Apache Webサーバ経由のSSL構成は成功です！

`http://<hostname>:52773/csp/sys/UtilHome.csp` : このURLはIRIS コンテナのPrivate Apache経由のアクセスです。構成したApacheは経由しません。

`http://<hostname>/csp/sys/UtilHome.csp` :  
このURLは構成したApacheを経由して管理ポータルにアクセスしています。

`https://<hostname>/csp/sys/UtilHome.csp` :  
このURLは構成したApache経由でSSL接続を使用して管理ポータルにアクセスしています。

## SSL構成の作成

それでは、IRIS for Healthが起動し、管理ポータルにアクセスできるようになったので、最後の準備としてSSL構成を作成しましょう。

管理ポータル システム管理 セキュリティ SSL/TLS構成  
と進み、先ほど作成した3ペアの証明書キーを使って、SSLの構成を3つ作成しておいてください。

名前は自由に決めることができますが、この記事では過去のOAuth2関連記事にならって、SSL4AUTH/SSL4CLIENT/SSL4RESSERVER としています。

サーバ 595804d6fc04 ネームスペース %SYS ユーザ \_SYSTEM ライセンス先 InterSystems IRIS Community インスタンス IRIS

システム > セキュリティ管理 > SSL/TLS 構成 > 新規 SSL/TLS 構成 - (セキュリティの設定)

## 新規 SSL/TLS 構成

保存

キャンセル

テスト

以下のフォームを使用して新しいSSL/TLS構成を作成します:

構成名	<input type="text" value="SSL4AUTH"/> <small>必須です。</small>								
説明	<input type="text"/>								
有効	<input checked="" type="checkbox"/>								
タイプ	<input checked="" type="radio"/> クライアント <input type="radio"/> サーバ								
サーバ証明書の検証	<input checked="" type="radio"/> なし <input type="radio"/> 必須								
信頼済み認証局の証明書を含むファイル	<input type="text"/> <input type="button" value="参照..."/>								
このクライアントの認証情報	<p><small>注意: このクライアントがサーバから認証を受ける必要がある場合のみ必要です。</small></p> <p>このクライアントの証明書を含むファイル <input type="text" value="/ISC/sslkeys/auth.cer"/> <input type="button" value="参照..."/></p> <p>関連づけられた秘密鍵を含むファイル <input type="text" value="/ISC/sslkeys/auth.key"/> <input type="button" value="参照..."/></p> <p>秘密鍵タイプ <input checked="" type="radio"/> RSA <input type="radio"/> DSA</p> <p>秘密鍵パスワード <input type="password" value="....."/></p> <p>秘密鍵パスワード(再入力) <input type="password" value="....."/></p>								
暗号方式設定	<table><tr><td>Minimum Protocol Version</td><td><input type="text" value="TLSv1.2"/></td></tr><tr><td>Maximum Protocol Version</td><td><input type="text" value="TLSv1.3"/></td></tr><tr><td>Enabled cipherlist (TLSv1.2 and below)</td><td><input type="text" value="ALL:!aNULL:!eNULL:!EXP:!SSLv2"/></td></tr><tr><td>Enabled ciphersuites (TLSV1.3)</td><td><input type="text" value="TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_"/></td></tr></table>	Minimum Protocol Version	<input type="text" value="TLSv1.2"/>	Maximum Protocol Version	<input type="text" value="TLSv1.3"/>	Enabled cipherlist (TLSv1.2 and below)	<input type="text" value="ALL:!aNULL:!eNULL:!EXP:!SSLv2"/>	Enabled ciphersuites (TLSV1.3)	<input type="text" value="TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_"/>
Minimum Protocol Version	<input type="text" value="TLSv1.2"/>								
Maximum Protocol Version	<input type="text" value="TLSv1.3"/>								
Enabled cipherlist (TLSv1.2 and below)	<input type="text" value="ALL:!aNULL:!eNULL:!EXP:!SSLv2"/>								
Enabled ciphersuites (TLSV1.3)	<input type="text" value="TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_"/>								

\*ホストとコンテナのディレクトリ共有について

docker-composeファイルに記載されている以下のvolumes

指定によって、ホストの現在のディレクトリ位置 = コンテナ内の /ISC ディレクトリ となっています。  
上記の設定等で証明書ファイルを指定する場合は、このディレクトリを活用してください。

volumes:  
- ./ISC

このディレクトリには、ファイルだけでなくIRISのデータベースファイルや構成ファイルも配置されます。  
詳しくはドキュメント「[永続インスタンス・データを保存するための永続的な %SYS](#)」をご覧ください。

## IRIS for HealthでOAuth2の構成

いよいよIRIS for HealthにOAuth2を使ってアクセスするための具体的な準備に入ります！

### OAuth2認可サーバ構成

まずはOAuth2認可サーバの構成です！

管理ポータル システム管理 セキュリティ OAuth 2.0 サーバ と進みます。

以下の内容に従って設定します。

「一般」タブ設定内容	発行者エンドポイント：ホスト名
	: プレフィックス

サポートする許可タイプ
SSL/TLS構成

システム > セキュリティ管理 > OAuth 2.0 認証サーバ構成 - (セキュリティの設定)\*

## OAuth 2.0 認証サーバ構成

保存

キャンセル

削除

クライアントデスク립ション

以下のフォームで OAuth 2.0 認証サーバ構成を編集します:

一般 スcope 間隔 JWT 設定 カスタマイズ

説明 OAuth2認可サーバ

発行者エンドポイント この認証サーバのエンドポイントです。  
https://[redacted]/authserver/oauth2

ホスト名	ポート	プレフィックス
[redacted]		authserver
必須です。	オプションです。	オプションです。

オーディエンスが必要です

ユーザセッションをサポート

リフレッシュ・トークンを返す OpenID 接続に必要なもののみ

サポートする許可タイプ (少なくともひとつをチェック)  認証コード  
 暗黙的  
 リソース所有者のパスワード認証情報  
 クライアントの認証情報  
 JWT 承認

OpenID プロバイダのドキュメント

サービスのドキュメントURL	
ポリシー URL	
サービス URL の使用条件	

SSL/TLS構成 SSL4AUTH

「スコープタブ」では、「サポートしているスコープを追加」をクリックして追加します。  
あとで、表示される認可コード (Authorization Code) のログイン画面ではここで記述した「説明」が表示されます。

以下のフォームで新しい OAuth 2.0 認証サーバ構成を作成します:

Scope	説明	
scope1	Scope 1です。	編集
scope2	Scope 2です。	編集

少なくともひとつのサポートされたScopeが必要です。  
サポートしているScopeを追加

サポートしていないScopeを許可  
デフォルトのScope

「間隔」タブはデフォルトから変更する内容はありません。  
「JWT」タブでは署名アルゴリズムに「RS512」を選択してみましょう。

以下のフォームで新しい OAuth 2.0 認証サーバ構成を作成します:

JSON Web トークン (JWT) の設定

X509 認証情報から JWT 設定を作成

署名アルゴリズム  
RS512: RSASSA-PKCS1-V1\_5 using SHA-512

鍵管理アルゴリズム

コンテンツの暗号化アルゴリズム

最後の「カスタマイズ」タブでは、トークン・クラスを生成 の指定を %OAuth2.Server.JWT に変更します。

以下のフォームで新しい OAuth 2.0 認証サーバ構成を作成します:

カスタマイズのオプション

認証クラス  
%OAuth2.Server.Authenticate  
必須です。

ユーザクラスを検証  
%OAuth2.Server.Validate  
必須です。

セッション維持クラス  
OAuth2.Server.Session  
必須です。

トークン・クラスを生成  
%OAuth2.Server.JWT  
必須です。

ネームスペースのカスタマイズ  
%SYS  
必須です。

カスタマイズしたロール (少なくともひとつ選択)  
利用可能 選択済み

入力したら、「保存」ボタンを押して、構成を保存します。

これで、IRIS for HealthがOAuth2認可サーバとして稼働する基本の設定ができました。早速Postmanからアクセスしてアクセストークンが取得できるか試してみましょう！

しかし、その前にあと2つ構成を行わなければいけません。

## クライアントデスクリプションを追加する

まずはOAuth2クライアントとしてアクセスするPOSTMANの情報を追加します。OAuth2クライアント登録は動的登録などで追加されることもあります。

サーバ構成画面の「クライアントデスクリプション」をクリックして先に進みます。

システム > セキュリティ管理 > OAuth 2.0 認証サーバ構成 - (セキュリティの設定)

### OAuth 2.0 認証サーバ構成

保存    キャンセル    削除    クライアントデスクリプション

以下のフォームで OAuth 2.0 認証サーバ構成を編集します:

一般    スコープ    間隔    JWT 設定    カスタマイズ

「クライアントデスクリプションを作成」をクリックしてエントリを追加します。

以下の内容に従ってクライアントデスクリプションを作成します。

「一般」タブ設定内容	名前
	クライアントの種別
	リダイレクトURL
	サポートする許可タイプ
	認証署名アルゴリズム

システム > セキュリティ管理 > OAuth 2.0 認証サーバ構成 > OAuth 2.0 サーバ > クライアントデスク립ション

## クライアントデスク립ション

保存

キャンセル

以下のフォームを使用して、OAuth 2.0 認証サーバに登録されている新しいクライアントデスク립ションを作成します:

一般

クライアント認証情報

クライアント情報

JWT 設定

名前	<input type="text" value="postman"/>
	必須です。
説明	<input type="text" value="POSTMANからアクセステストするためのエントリです。"/>
クライアントの種別	<input checked="" type="radio"/> 機密 <input type="radio"/> 公開 <input type="radio"/> リソース・サーバ
	必須です。
リダイレクト URL	<input type="text" value="https://www.getpostman.com/oauth2/callback"/>
	少なくともひとつの URL が必要です。リストのアイテムをクリックして編集または削除してください。
	<input type="button" value="URL を追加"/>
	<input type="button" value="OK"/> <input type="button" value="削除"/> <input type="button" value="キャンセル"/>
サポートする許可タイプ (少なくともひとつをチェック)	<input checked="" type="checkbox"/> 認証コード <input type="checkbox"/> 暗黙的 <input type="checkbox"/> リソース所有者のパスワード認証情報 <input type="checkbox"/> クライアントの認証情報 <input checked="" type="checkbox"/> JWT 承認
サポートされている応答タイプ (少なくとも1つをチェックする)	<input checked="" type="checkbox"/> コード <input checked="" type="checkbox"/> id_token <input checked="" type="checkbox"/> id_token トークン <input checked="" type="checkbox"/> トークン
認証タイプ	<input type="radio"/> なし <input checked="" type="radio"/> ベーシック <input type="radio"/> エンコードされたボディから <input type="radio"/> クライアントシークレット JWT <input type="radio"/> 秘密鍵 JWT
認証署名アルゴリズム	<input type="text" value="RS512"/>

入力できたら「保存」ボタンを押してクライアントデスク립ションを保存します。

「クライアント認証情報」タブをクリックすると、このエントリに対するクライアントIDとクライアントの秘密鍵を確認することができます。  
POSTMANからテストするときはこのIDと秘密鍵が必要になります。

システム > セキュリティ管理 > OAuth 2.0 認証サーバ構成 > OAuth 2.0 サーバ > クライアントデスク립ション

## クライアントデスク립ション

保存

キャンセル

以下のフォームを使用して、OAuth 2.0 認証サーバに登録されている既存のクライアントデスク립ションを編集します:

一般

クライアント認証情報

クライアント情報

JWT 設定

クライアント ID	<input type="text" value="-GIKMcY [REDACTED] sMako"/>
クライアントの秘密鍵	<input type="text" value="FJQg5R [REDACTED] 2HEKqngd_vYHtdT5kun_yFgXWayZzBqaQOkRQadMGIE0fr6Ow"/> <input type="button" value="隠す"/>

### ウェブアプリケーションの追加

POSTMANからアクセスする前にもう一つ大事な設定を加える必要があります。  
OAuth2認可サーバ構成画面で、この構成のエンドポイントが `https://<hostname>/authserver/oauth2` と決定されました。

このエンドポイントへのアクセスが正しくIRISで処理されるために、このURLパスに対するウェブアプリケーションの追加を行う必要があります。

システム管理 セキュリティ アプリケーション ウェブ・アプリケーション  
と進み、「新しいウェブ・アプリケーションを作成」をクリックします。

システム > セキュリティ管理 > ウェブ・アプリケーション

## ウェブ・アプリケーション

新しいウェブ・アプリケーションを作成

以下が現在定義されているウェブ・アプリケーションの一覧です:

名前	ネームスペース	ネームスペースのデフォルト	有効	タイプ	リソース	認証方法
<a href="#">/api/atelier</a>	%SYS	いいえ	はい	CSP	%Development	パスワード
<a href="#">/api/deepsee</a>	%SYS	いいえ	はい	CSP		パスワード

OAuth2のウェブ・アプリケーションのテンプレートが用意されているので、まず「コピー元」から「/oauth2」を選択します。

「ウェブ・アプリケーションの編集」設定項目	コピー元	
	名前	/authserver/oauth2
	有効	「REST」のラジオボタン

各値を入力したら保存します。

システム > セキュリティ管理 > ウェブ・アプリケーション > ウェブ・アプリケーションの編集 - (セキュリティの設定)

## ウェブ・アプリケーションの編集

保存

キャンセル

以下のフォームを使用して新しいウェブ・アプリケーションを作成します:

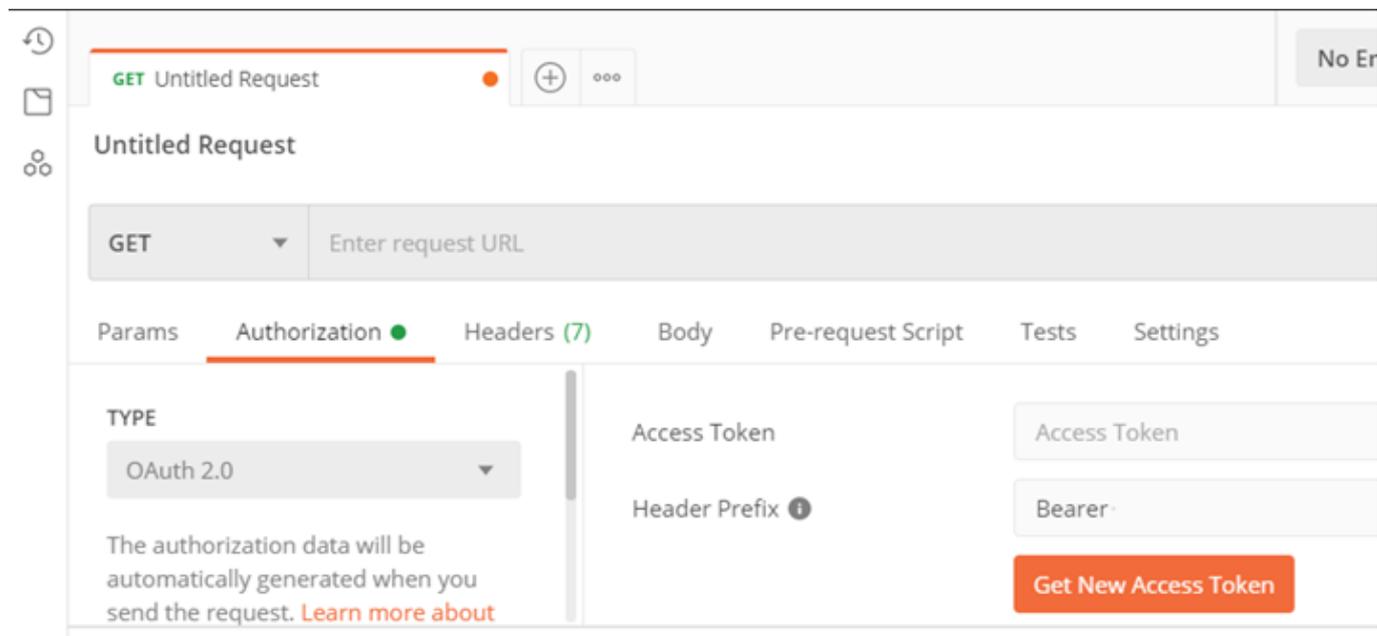
名前	<input type="text" value="/authserver/oauth2"/>	必須です。(例: /csp/appname)
コピー元	<input type="text" value="/oauth2"/>	← 最初に選択します
説明	<input type="text" value="OAuth 2.0 Authorization Server."/>	
ネームスペース	<input type="text" value="%SYS"/>	%SYSのデフォルト・アプリケーション: /csp/sys <input type="checkbox"/> ネームスペースのデフォルト・アプリケーション
アプリケーション有効	<input checked="" type="checkbox"/>	
有効	<input checked="" type="radio"/> REST ディスパッチ・クラス <input type="text" value="OAuth2.Server.REST"/> 必須です。	
	<input type="radio"/> CSP/ZEN <input type="checkbox"/> アナリティクス <input type="checkbox"/> 善い Web サービス <input type="checkbox"/> ログイン CSRF 攻撃を防ぐ	
セキュリティの設定	必要なリソース <input type="text"/> IDでグループ化 <input type="text"/>	
	許可された認証方法 <input checked="" type="checkbox"/> 認証なし <input type="checkbox"/> パスワード <input type="checkbox"/> Kerberos <input type="checkbox"/> ログイン Cookie	
セッションの設定	セッションタイムアウト <input type="text" value="900"/> 秒 イベントクラス <input type="text"/> .cls	
	セッションにクッキーを使用する <input type="text" value="なし"/> セッションクッキーパス <input type="text" value="/authserver/oauth2/"/>	

## POSTMANからOAuth2のテストを行う

それではPOSTMANからテストしてみましょう。  
テストの実行は他のツールや実際のプログラムから行うこともできます。

POSTMANの詳しい説明はこの記事では触れませんが、一点注意点として、POSTMANのSettingでSSL certificate verificationはOFFに変更するようにしてください。

POSTMANで新しいリクエストを作成後、AuthorizationタブのTYPEで「OAuth 2.0」を選択し、「Get New Access Token」をクリックします。



次の画面で、以下の内容に従って値を入力していきます。

「GET NEW ACCESS TOKEN」設定項目	Token Name
	Grant Type
	Callback URL
	Auth URL
	Auth Token URL
	Client ID
	Client Secret
	Scope
	State

### GET NEW ACCESS TOKEN ✕

Token Name

Grant Type

Callback URL ⓘ

Authorize using browser

Auth URL ⓘ

Access Token URL ⓘ

Client ID ⓘ

Client Secret ⓘ

Scope ⓘ

State ⓘ

Client Authentication

---

パラメータ入力後、「Request Token」ボタンを押すと、以下のようなログイン画面が表示されたでしょうか？



管理ポータルにアクセスしているユーザ情報 (SYSTEM等) でログインしてみてください。

ログイン後の次の画面では、このアプリケーションに権限の許可を与えるかどうか決定できます。  
「許可」をクリックしたあと、次の画面で以下のようにAccess Tokenが表示されれば、アクセストークン取得テストは成功です！





いかがでしたか？ Access Token そして idtokenが取得できたでしょうか？

証明書等、少々手間がかかる準備もありますが、データベースプラットフォームであるIRIS for Health上でもこれだけ簡単にOAuth2認可サーバを構築することができました。

この連載の次のパートでは、いよいよFHIRリポジトリを構築し、FHIRリポジトリのOAuth2リソースサーバ登録、そして再びPOSTMANからOAuth2アクセストークンを使用したFHIRリポジトリへRESTアクセスする方法を紹介していきます。

[#FHIR #OAuth2 #InterSystems IRIS for Health](#)

ソースURL:<https://jp.community.intersystems.com/post/iris-health-%E4%B8%8A%E3%81%A7fhir-%E3%83%A%E3%83%9D%E3%82%B8%E3%83%88%E3%83%AA%EF%BC%8Boauth2-%E8%AA%8D%E5%8F%AF%E3%82%B5%E3%83%BC%E3%83%90%E3%83%AA%E3%82%BD%E3%83%BC%E3%82%B9%E3%82%B5%E3%83%BC%E3%83%90%E6%A7%8B%E6%88%90%E3%82%92%E6%A7%8B%E7%AF%89%E3%81%99%E3%82%8B-%E3%83%91%E3%83%BC%E3%83%881>