記事
Shintaro Kaminaka · 2020年9月11日 8m 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2

開発者の皆さん、こんにちは。

今回の記事では前回の記事に引き続き、IRIS for Health上で、FHIRリポジトリ＋OAuth2認可サーバ/リソースサーバを構成する方法をご案内します。

(注意：2020.4以降のバージョンではこの記事に記載されているスコープ指定では正しくリソースが取得できません。詳細はこちらの記事をご覧ください。)

パート1では、事前準備と、OAuth2認可サーバを構成し、アクセストークンを取得するとこまでをご紹介しました。
このパート2では、FHIRリポジトリの構築方法と、OAuth2クライアント/リソースサーバの構成方法をご紹介していきます。

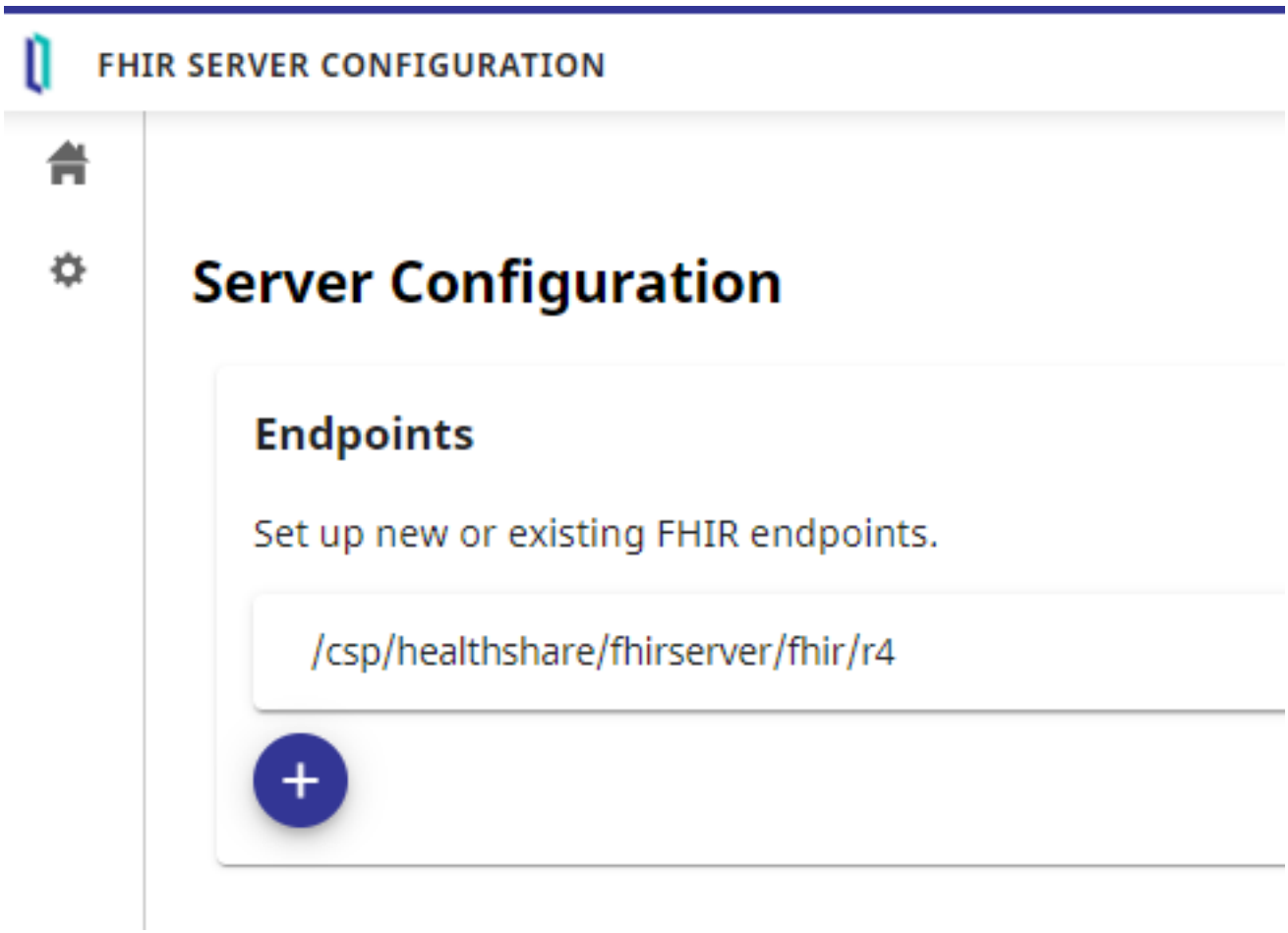
今日構成する、FHIRリポジトリおよび、OAuth2クライアント/リソースサーバの構成は、前回パート１で構成したOAuth2認可サーバのIRISインスタンスと分けることもできますし、同じインスタンスに同居させることもできます。
この記事の中では前回と同じインスタンス上に構成していきます。

FHIRリポジトリの構築とOAuth Client Nameの指定

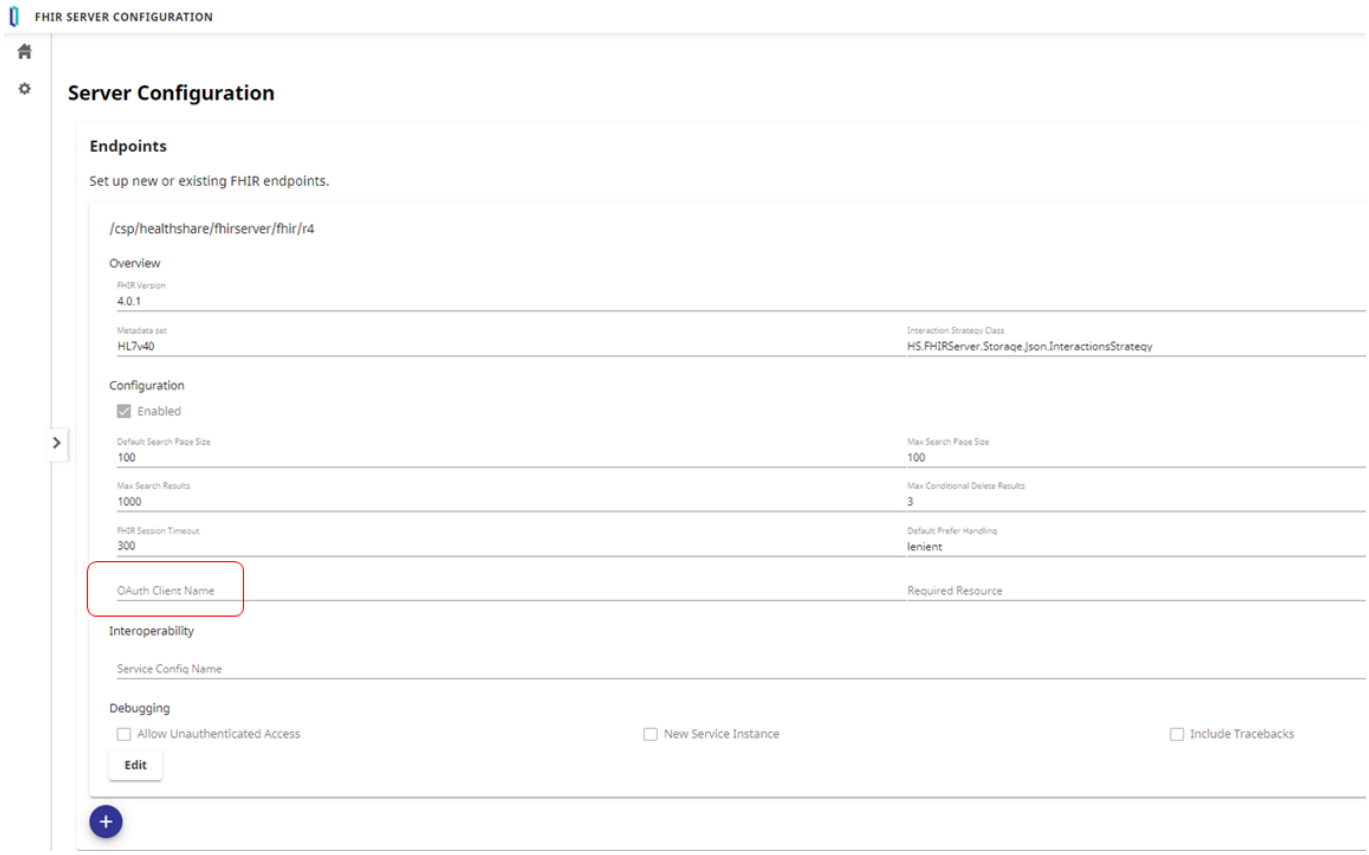
FHIRリポジトリの構築方法は、過去の記事「Azure上でIRIS for Healthをデプロイし、FHIRリポジトリを構築する方法」で紹介しています。

管理ポータルにアクセスし、ネームスペース/データベースを作成、さらにHealthメニューからFHIRリポジトリを構築する手順はこの記事を参考にして進めてください。
この記事の中でもネームスペース/データベース=FHIRSERVER , FHIR endpointを/csp/healthshare/fhirserver/fhir/r4として構成しています。

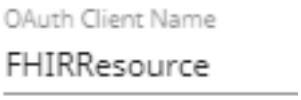
構築後の以下の画面で、endpoint URL /csp/healthshare/fhirserver/fhir/r4 をクリックして構成画面を開きます。



構成画面で **OAuth Client Name** を欄に、これから作成するOAuth2クライアントの構成名を入力しておきます。
先にOAuth2クライアントを構成している場合はその名前に合わせてください。



ここでは、「FHIRResource」という文字列にしておきます。変更するには上記画面で「Edit」ボタンを押して変更し、「Update」ボタンで保存します。



OAuth2クライアントの構成

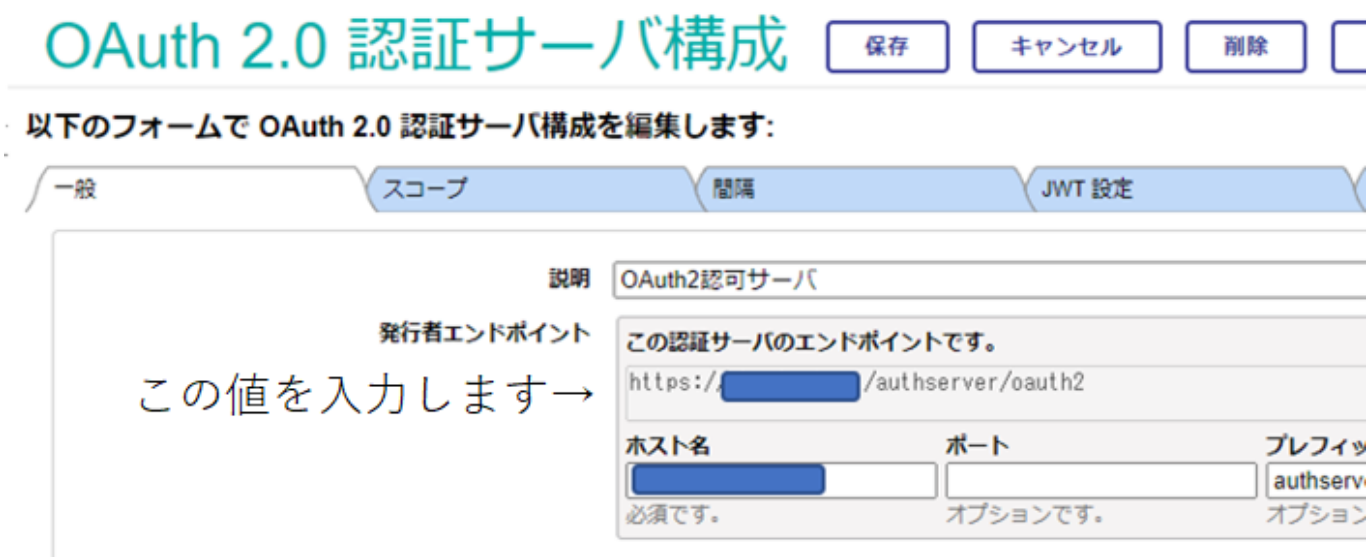
次は、OAuth2クライアント構成を作成していきましょう。

管理ポータルのシステム管理→セキュリティ→OAuth2.0
と進み、前回パート１とは異なり、「サーバ」ではなく、「クライアント」を選択します。

次の画面では「サーバの説明を作成」をクリックして、OAuth2認可サーバへ接続するための構成を作成します。

サーバデスクリプション
ページで発行者エンドポイントには、パート1で構成した認可サーバのエンドポイントを入力します。

以下はパート１で構成したOAuth2認可サーバの構成画面です。



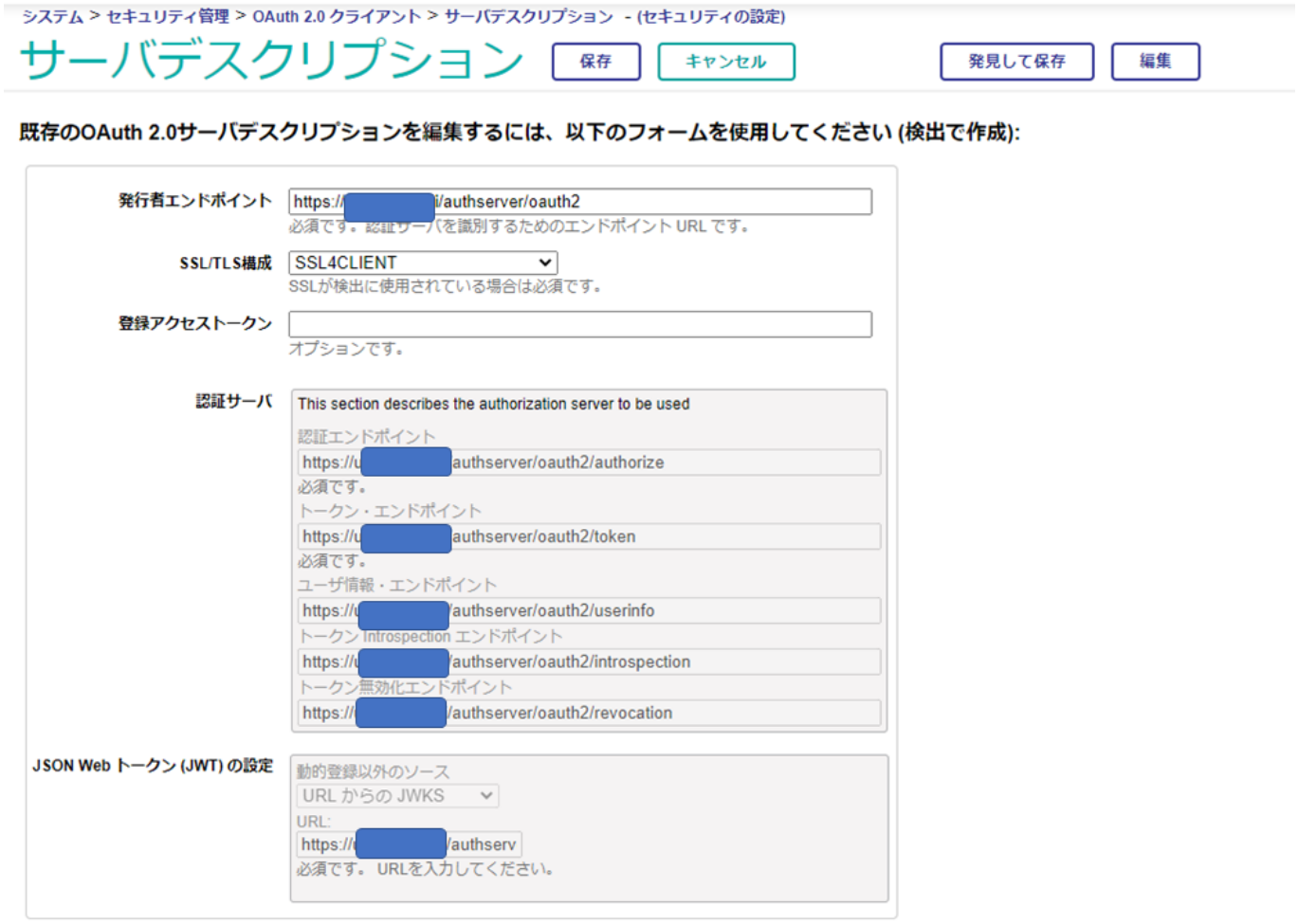
SSL/TLS構成には、パート1の事前準備で作成した、SSL/TLS構成「SSL4CLIENT]を入力します。

項目を入力したら「発見して保存」を実行して、OAuth2認可サーバから情報を取得します！

アクセスに成功すると以下のように取得できた情報が表示されます。
パート1事前準備で用意したホスト名を指定したSSL証明書が正しく作成・認識されていない場合、この過程でエラーが発生することがありますので、ご注意ください。

注意：この連載のパート1でDLしたdocker-containerファイルを使っている場合でも、IRISコンテナ→Apacheコンテナへホスト名を指定したアクセスがうまくいかない場合があります。この場合、以下のようにextrahostsとして、docker-
compose.ymlファイルに自分のマシンのホスト名とIPアドレスを入力すると解決できることがあります。

```
    extrahosts:
      - <yourhostname>:<your ip address>
```



The following is a list of server metadata properties:

名前	値
issuer	https://[redacted]i/authserver/oauth2
authorization_endpoint	https://[redacted]i/authserver/oauth2/authorize
token_endpoint	https://[redacted]i/authserver/oauth2/token
userinfo_endpoint	https://[redacted]i/authserver/oauth2/userinfo
revocation_endpoint	https://[redacted]i/authserver/oauth2/revocation
introspection_endpoint	https://[redacted]i/authserver/oauth2/introspection
jwks_uri	https://[redacted]i/authserver/oauth2/jwks
registration_endpoint	https://[redacted]i/authserver/oauth2/register
scopes_supported	openid, profile, email, address, phone, scope1, scope2
response_types_supported	code
response_modes_supported	query, fragment, form_post
grant_types_supported	authorization_code, jwt_authorization, refresh_token
id_token_signing_alg_values_supported	HS256, HS384, HS512, RS256, RS384, RS512
id_token_encryption_alg_values_supported	none, RSA1_5, RSA-OAEP, A128KW, A192KW, A256KW, dir
id_token_encryption_enc_values_supported	none, A128CBC-HS256, A192CBC-HS384, A256CBC-HS512
userinfo_signing_alg_values_supported	none, HS256, HS384, HS512, RS256, RS384, RS512
userinfo_encryption_alg_values_supported	none, RSA1_5, RSA-OAEP, A128KW, A192KW, A256KW, dir
userinfo_encryption_enc_values_supported	none, A128CBC-HS256, A192CBC-HS384, A256CBC-HS512
access_token_signing_alg_values_supported	none, HS256, HS384, HS512, RS256, RS384, RS512
access_token_encryption_alg_values_supported	none, RSA1_5, RSA-OAEP, A128KW, A192KW, A256KW, dir
access_token_encryption_enc_values_supported	none, A128CBC-HS256, A192CBC-HS384, A256CBC-HS512
request_object_signing_alg_values_supported	none, HS256, HS384, HS512, RS256, RS384, RS512
request_object_encryption_alg_values_supported	none, RSA1_5, RSA-OAEP, A128KW, A192KW, A256KW, dir
request_object_encryption_enc_values_supported	none, A128CBC-HS256, A192CBC-HS384, A256CBC-HS512
token_endpoint_auth_methods_supported	client_secret_post, client_secret_basic, client_secret_jwt, private_key_jwt
token_endpoint_auth_signing_alg_values_supported	HS256, HS384, HS512, RS256, RS384, RS512
claims_supported	preferred_username, email, email_verified, name, phone_number, phone_number_verified, iss, sub, aud, exp, auth_time, jti
ui_locales_supported	de, en, en-us, es, fr, it, ja, ko, nl, pt-br, ru, uk, zh-cn
claims_parameter_supported	true
request_parameter_supported	true
request_uri_parameter_supported	true

「保存」を押して構成を保存すると、以下のページに戻りますので、続いて「クライアント構成」を選択してFHIRリポジトリ用の構成を作成していきます。

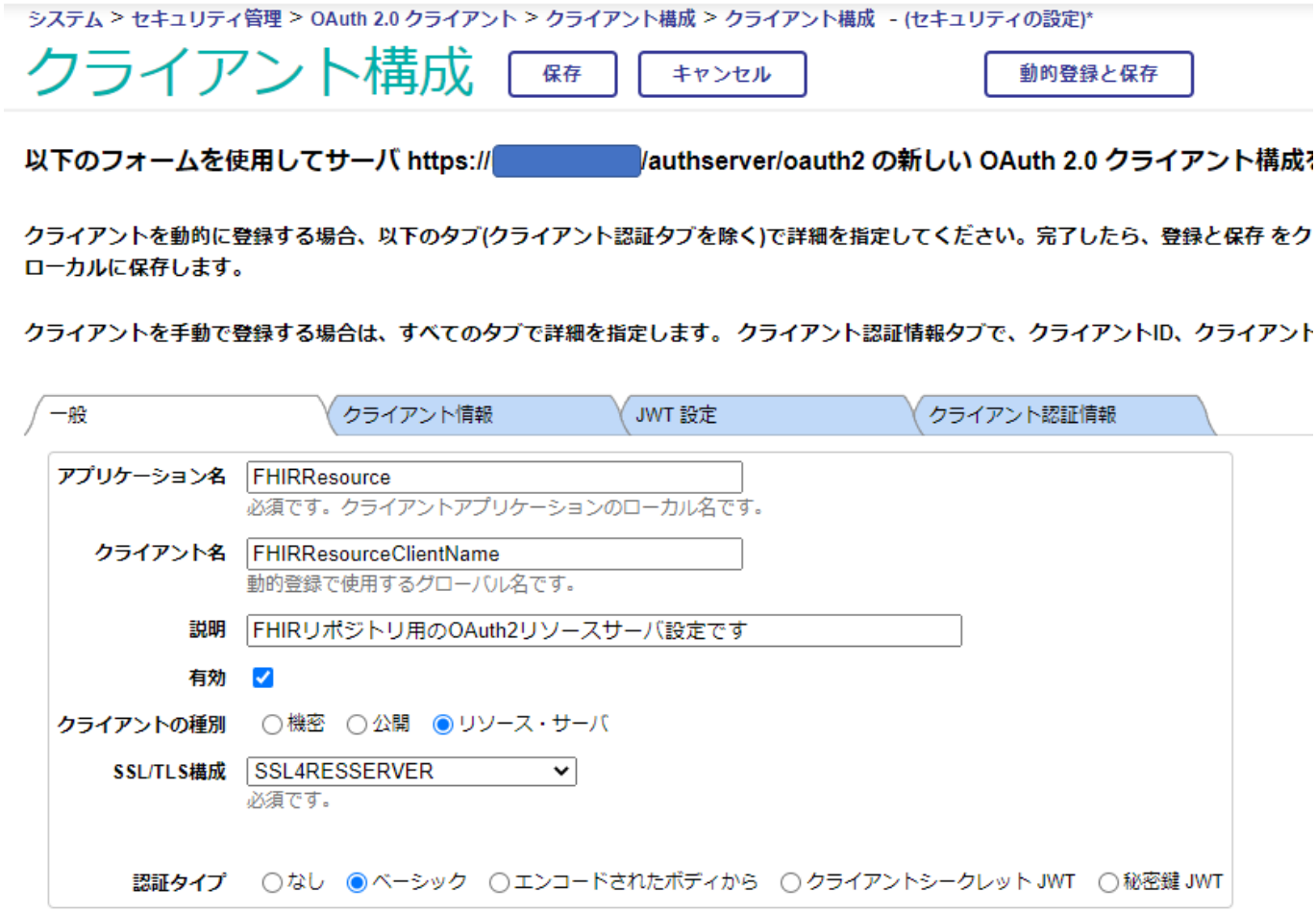
OAuth2クライアントにクライアント構成を追加する

ややこしいタイトルですが、次は今作成したOAuth2クライアント設定（どのOAuth2認証サーバに接続するかという情報をもつ）に、クライアント構成（OAuth2クライアントとしてOAuth2認可サーバに接続したい、具体的なFHIRリポジトリやCSPアプリケーションなどの情報）を追加します。

次の画面では「クライアントの構成を作成」をクリックして以下の画面を表示し、必要な項目を設定していきます。

最初に、クライアントの種別＝リソース・サーバ を選択すると、下記入力画面と同じになります。

アプリケーション名	FHIRResou
クライアント名	OAuth2認可 は違う名前
説明	この構成の
クライアントの種別	「リソース
SSL/TLS構成	パート1事前



入力が完了したら、「動的登録と保存」ボタンをクリックして保存とサーバへの登録を行います。
（ちょっとわかりにくいですが）ボタンの表示が「動的登録と保存」から「更新メタデータを取得して保存」に変わったら登録が成功しています。

OAuth2認可サーバ側の構成情報を見て、本当に登録されているか確認してみましょう。

管理ポータル→システム管理→セキュリティ管理→OAuth2.0→サーバ
の画面で「クライアントデスクリプション」をクリックすると以下のように登録されていることがわかります。

名前がクライアント名で指定した名前になっていることが確認できます。

パート1ではPostmanからアクセステストする際は、このクライアントデスクリプション画面をさらに進んで表示される、クライアント認証情報（クライアントIDとくらい案tの秘密鍵）を手動でコピーしましたが、今回は動的登録の過程でこれらの情報はクライアント側に受け渡されています。

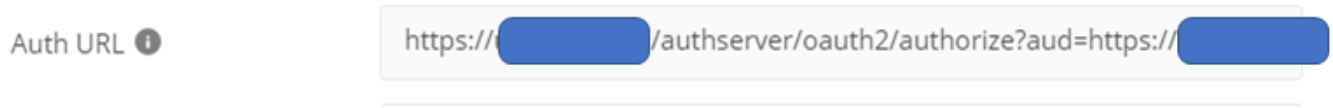
PostmanからOAuth2アクセストークンを使って、FHIRリポジトリへアクセスする

それではいよいよ、Postmanからアクセスしてみましょう！

まずアクセストークンを取得します。基本はパート1の最後で取得した方法と同じですが、アクセストークンの発行先を表すaudienceパラメータを追加する必要があります。

```
aud=https://<hostname>/csp/healthshare/fhirserver/fhir/r4
```

Postmanで具体的に追加するには、以下のようにAuthorization Codeのendpoint URLにパラメータとして追加します。
（Postmanの画面の都合上パラメータの全体が見えませんが、上記の aud=https://<hostname>/csp/healthshare/fhirserver/fhir/r4 をすべて記載してください）



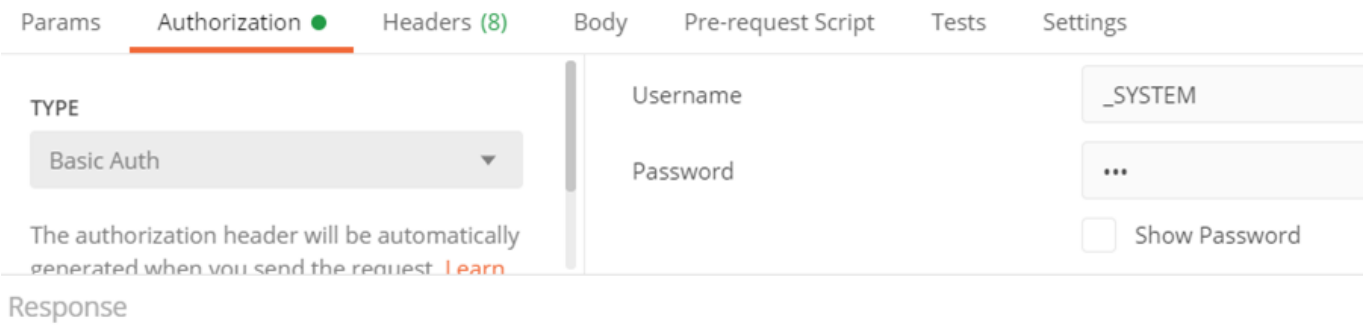
注意：Postmanに入力するClient IDやClient Secretは先ほどのリソースサーバの動的登録で発行されたものに**変更する必要はありません。**パート1で追加したpostman用に発行されたクライアントIDと秘密鍵を使用します。

アクセストークンが取得できたら、その内容をコピーしておいてください。

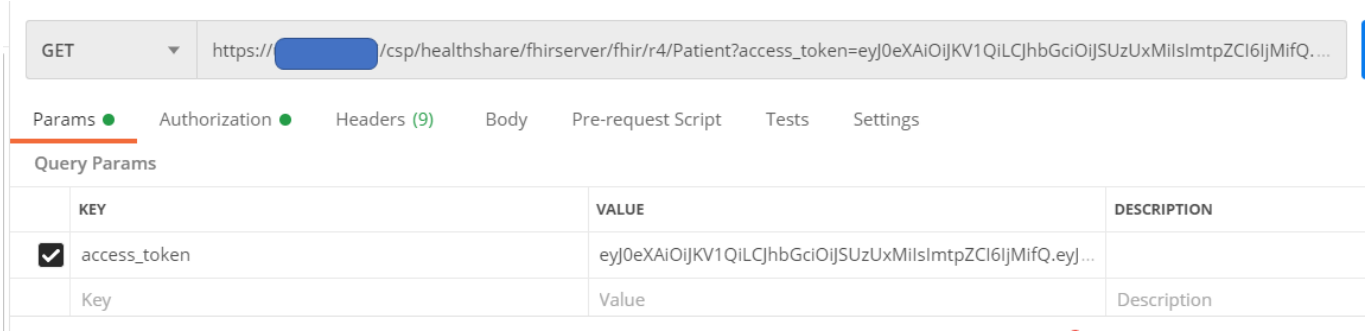
PostmanではこのままAuthorizationのTYPEをOAuth2にしておくと、アクセストークンを送信する機能がありますが、IRIS for HealthのFHIRリポジトリでは、OAuth2のアクセストークンだけではなく、Basic認証のユーザ・パスワード情報も送信する必要があります。

そのため、Postmanからアクセスする場合は、（ちょっと手間ですが）AuthorizationのTYPEはBasic Authにして、ユーザ名パスワードを入力し、アクセストークンはFHIRリポジトリへのRESTリクエストのParameterとして送信する必要があります。

具体的には、まず以下の画面のようにユーザ名・パスワードを入力します。このユーザ情報は、アクセストークンのsub内に含まれるユーザ情報と一致しているかの確認が行われるため、必ずアクセストークン取得時に入力したユーザ情報と同じユーザである必要があります。



次に、Paramsタブで、 **accesstoken** にパラメータに先ほどのアクセストークン値を入力します。



FHIRリポジトリを構築したばかりであれば、リポジトリには何のデータもはいってはいませんが、Patientデータをリクエストしてみましょう！

Request URL には https://<hostname>/csp/healthshare/fhirserver/fhir/r4/Patient
を入力し、HTTPのメソッドはGETを選択します（上の図のようになります）

Sendボタンを押してリクエストを投げてみましょう！以下のようにFHIRのBundleが取得できれば、アクセストークンを使用したFHIRリポジトリへのアクセスは成功です！

FHIRリポジトリへのデータの登録や検索の方法については、IRIS for Healthのドキュメントやコミュニティの記事をご参照ください。

IRIS for Health 2020.1 日本語ドキュメント：リソースリポジトリ

IRIS for Health 2020.3 英語ドキュメント：Resource Repository

(この記事を執筆した段階では、2020.3はPreview Editionです。)

いかがでしたか？FHIRリポジトリへのアクセスが成功したでしょうか？

この連載で紹介した構成内容は最も単純な構成ですが、実際のFHIRプロジェクトではユーザの認めたスコープによってどの範囲のデータまで返すように実装するか？といった検討と実装が必要になってきます。

開発者コミュニティでは引き続きFHIRに関する情報を発信していきたいと思います。

#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%AA%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%882