

記事

[Shintaro Kaminaka](#) · 2021年4月15日 9m 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4(代行認証編)

開発者の皆さん、こんにちは。

以前の[記事](#)でIRIS for Health上でFHIRリポジトリを構築し、OAuth2認証を構成する方法をご紹介しました。

この代行認証編では、IRIS for HealthのFHIRリポジトリに組み込まれた認証機能ではなく、IRISの代行認証機能 + ZAUTHENTICATEルーチンを使用して認証を行う方法をご紹介します。

前回記事でご紹介したように、標準のFHIRリポジトリの認証機構では、アクセストークンの発行先を追加するためのAudienceの指定(`aud=https://`)や、アクセストークンだけではなくベーシック認証の情報を送付するなどの対応が必要でした。

スクラッチでFHIRクライアントを開発するのではなく、既成の製品やアプリケーションからアクセスする場合、上記のような処理の実装ができないことがあるかもしれません。そのような場合には、この代行認証 + ZAUTHENTICATEルーチンを使用して、カスタマイズした認証の仕組みを構築することができます。

この記事に含まれる情報のドキュメントについて

この記事で記載されている情報はIRIS for Healthのドキュメントにも含まれている内容をわかりやすく再構成したものです。

[RESTサービスの保護:RESTアプリケーションおよびOAuth2.0](#)

[OAuth 2.0 クライアントとしての InterSystems IRIS Web アプリケーションの使用法](#)

代行認証を有効にする

まず使用しているIRIS環境で「代行認証」機能を有効にし、アクセスするFHIRリポジトリの「Webアプリケーション設定」で「代行認証」機能を使える用に構成します。

認証/ウェブセッションオプション画面

まずシステムとして「代行認証」が使用できるように構成します。

管理ポータルでのシステム管理 セキュリティ システム・セキュリティ 認証/ウェブセッションと進み、「代行認証を許可」をチェックします。

「代行認証によるOS認証を許可」ではありませんのでご注意ください。

システム > セキュリティ管理 > 認証/ウェブセッションオプション - (セキュリティの設定)

認証/ウェブセッションオプション

保存

キャンセル

セキュリティ認証/ウェブセッションオプションを編集:

認証なしアクセスを許可	<input checked="" type="checkbox"/>
OS 認証を許可	<input checked="" type="checkbox"/>
代行認証によるOS認証を許可	<input type="checkbox"/>
LDAP認証によるOS認証を許可	<input type="checkbox"/>
パスワード認証を許可	<input checked="" type="checkbox"/>
代行認証を許可	<input checked="" type="checkbox"/>
Kerberos認証を許可	<input checked="" type="checkbox"/>
LDAP認証を許可	<input type="checkbox"/>
LDAPキャッシュ credentials 認証を許可	<input type="checkbox"/>
ログイン Cookie の作成を許可	<input type="checkbox"/>
ログイン Cookie の有効期間 (秒)	<input type="text" value="0"/>
0 は非永続 Cookie を意味します	
二要素タイムベースのワンタイム・パスワード認証を許可	<input type="checkbox"/>
二要素 SMS テキスト認証を許可	<input type="checkbox"/>

%ServiceWebGatewayサービス 画面

次に、CSPゲートウェイを経由したWebのアクセスに対して、「代行認証」が有効になるよう構成します。

管理ポータル の システム管理 セキュリティ サービス と進み、「%ServiceWebGateway」をクリックして、許可された認証方法の「代行」にチェックがついていることを確認します。もしチェックされていないければ、チェックして保存を実行してください。

FHIRリポジトリの ウェブアプリケーションの編集 画面

最後に、アクセスするFHIRリポジトリの ウェブ・アプリケーションの編集画面で「代行認証」を有効にします。

管理ポータル の 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、該当のFHIRリポジトリアプリケーションを選択します。
特に変更をしていなければ、/csp/healthshare/<namespace>/fhir/r4 となっています。

この画面で、セキュリティの設定：許可された認証方法の「代行」をチェックして保存します。

これで、「代行認証」を利用する準備はOKです。次は、実際に代行認証のためのロジックが記載されたZAUTHENTICATEルーチンを用意します。

ZAUTHENTICATEルーチンの入手とインポート

ZAUTHENTICATEルーチンのサンプルはInterSystemsのGitHubで公開されています。

[GitHub:Samples-Security](#)

この記事ではここで紹介されているREST.ZAUTHENTICATE.macルーチンを利用します。
GitHubのREADMEに記載されているこのルーチンの説明をここにも転載します。

- REST.ZAUTHENTICATE.mac is another sample routine that demonstrates how to authenticate a REST application using OAuth 2.0. To use this sample:
 1. Configure the resource server containing the REST application as an OAuth 2.0 resource server.
 2. Copy this routine to the %SYS namespace as ZAUTHENTICATE.mac.
 3. Modify value of applicationName in ZAUTHENTICATE.mac.
 4. Allow delegated authentication for %Service.CSP.
 5. Make sure that the web application for the REST application is using delegated authentication.

この記事では、先に手順の4.,5.を済ませているので、ルーチンのインポートを実施しましょう。
(上記READMEでは、%Service.CSPと記載されていますが、現在は%ServiceWebGatewayになっています。)

GitHubからルーチンをダウンロード

してインポートするか、あるいは、この[リンク](#)から直接ルーチンを表示し、中身をStudioやVS Codeのエディタを使ってコピーしてZAUTHENTICATEルーチンをつくることもできます。%SYSネームスペースに作成します。

(注意

: 2021/4/16時点ではこのルーチンをスタジオからインポートするとエラーが発生してしまいます。お手数ですが、ファイルの中身をコピーしてZAUTHENTICATEルーチンを作成する方法で回避してください。)

ZAUTHENTICATEルーチンを作成したら、applicationNameを変更します。これは前回の記事で記載したOAuth2クライアントアプリケーションのクライアント構成画面で作成した「アプリケーション名」を指定します。

ここでは前回の記事にならない「FHIRResource」としています。コードの一部を紹介します。

```
// Usually you will need to modify at least the roles to be assigned.
set roles="%DB_DEFAULT,%Operator"

$$$SysLog(3,"OAuth2",[ZAUTHENTICATE],"ServiceName="_ServiceName_", Username="_Username_", roles="_roles")

// MUST BE MODIFIED FOR EACH INSTANCE WHICH USES OAuth 2.0 for REST.
// The application name used to define the authorization server to this resource server.
set applicationName="FHIRResource"

set loginSuccessful=0
set errorText=""
```

コードを変更したらコンパイルを実行します。

このZAUTHENTICATEルーチンで重要なのは以下のコード部分です。
GetAccessTokenFromRequestメソッドを使用してHTTPリクエストからアクセストークンを取り出し、ValidateJWTメソッドを使用してValidationを実施し正しいアクセストークンであることを確認しています。

```
// This ZAUTHENTICATE routine will support OAuth 2.0 based
```

```
// delegated authentication for subclasses of %CSP.REST.
set accessToken=##class(%SYS.OAuth2.AccessToken).GetAccessTokenFromRequest(.sc)

// Check if authorized.
// if the access token is not a JWT, we would need to validate the access token
// using another means such as the introspection or userinfo endpoint.
if $$$ISOK(sc) {
    set valid=##class(%SYS.OAuth2.Validation).ValidateJWT(applicationName,accessToken
    ,,,,jsonObject,,,sc)
}
```

POSTMANからのテスト

それでは前回同様、RESTクライアントツールのPOSTMANからテストしてみましょう。
前回同様、まずはアクセストークンを取得します。

前回とは異なり、Auth URLにaudパラメータを追加する必要はありません。トークンを取得できたら、「Use Token」ボタンをクリックし、そのトークンを使用できるようにします。

次は、FHIRリポジトリへのアクセスです。今回は前回と異なり、ベーシック認証と組み合わせる必要はありませんので、そのままFHIRリポジトリにアクセスするRESTのURLのみを入力し、実行します。

FHIRリソースが取得できたら成功です。

2020.4以降の対応

IRIS for Health

2020.4ではこちらの[記事](#)

に掲載したように、FHIRリポジトリ上でアクセストークンのスコープ情報がチェックされるようになりました。このため、セキュリティ要件にも依存しますが、ZAUTHENTICATEルーチンで必ずしもアクセストークンのValidationチェックを行う必要はありません。

これまでこのシリーズで紹介してきましたように、IRIS for HealthがOAuth2認可サーバの役割も兼ねている場合、2020.4上で動かす最も単純な方法は、ZAUTHENTICATEルーチンのGetCredentialsラベルで、アクセストークンを取得する際にも指定したIRISパスワードユーザを返すようにすることです。

例：アクセストークンを取得した際のユーザと

同じユーザを返すようにする。(このdaikoユーザには%All権限を与えています)

```
GetCredentials(ServiceName,Namespace,Username>Password,Credentials) Public {
    if ServiceName="%Service_WebGateway" {
        // Supply user name and password for authentication via a subclass of %CSP.RE
ST
        set Username="daiko"
        set Password="xxxxxx"
    }
    quit $$$OK
}
```

こちらの[代行認証に関するドキュメント](#)

に記載があるように、GetCredentialsラベルで実在するIRISパスワードユーザが返された場合はそのユーザに認証が行われるため、ZAUTHENTICATEルーチンで実行されていたアクセストークンのValidationチェックのロジックは実施されなくなります。

ただし、アクセストークンの検証はその後FHIRリポジトリ上で実施されるため不正なアクセストークンでアクセ

