Article
Pravin Barton · Sep 1, 2022 4m read

Using OAuth 2.0 / OIDC for single sign-on to an IRIS REST application

Say I've been developing a web application that uses IRIS as the back end. I've been working on it with unauthenticated access. It's getting to the point where I would like to deploy it to users, but first I need to add authentication. Rather than using the default IRIS password authentication, I'd like users to sign in with my organization's Single Sign On, or some other popular identity provider like Google or GitHub. I've read that OpenID Connect is a common authentication standard, and it's supported by IRIS. What is the simplest way to get up and running?

Example 1: a plain CSP app

The documentation here shows a pretty straightforward option for using a CSP application as an OpenID Connect client.

The steps for that look like:

1. Set up the OAuth 2.0 server and client configuration in IRIS. See the "**Caché configuration"** section of Daniel Kutac's great article for more information.

2. Copy the OAUTH2.ZAUTHENTICATE routine from the samples repo in GitHub into your %SYS namespace, and rename it to ZAUTHENTICATE.

3. Enable delegated authentication system-wide.

4. Create a custom login page that extends from %OAuth2.Login, and override the DefineParameters method to specify the OAuth 2.0 application name and scopes:

```
Class MyOAuth2.Login Extends %OAuth2.Login
{

ClassMethod DefineParameters(Output application As %String, Output scope As %String,
Output responseMode As %String)
{
    Set application="my application name"
    Set scope="openid profile email"
    Set responseMode=..#RESPONSEMODE
    Quit
}

}
```

5. Enable the web application for delegated authentication, and set the custom login page to MyOAuth2.Login.cls.

6. A final trick: In order for the custom login page to work, the CSPSystem user in IRIS needs to be specifically granted READ access to the database that MyOAuth2.Login.cls lives in.

At that point, login should "just work" - visiting a CSP page in that web application will redirect to the login page on the identity provider. After logging in the user will have an authenticated CSP session. Their $username will be equal to their subject identifier from SSO/Google/GitHub/wherever, so I can use IRIS's built-in authorization to

determine what to give them access to.

Example 2: the trouble with REST

What if the web application is using a REST handler? The above process doesn't work. If a web application is enabled for REST, there's no way to define a custom login page. I've found you need a few more steps to work around this.

7. Create a separate web application that does not have REST enabled. The path on that application must begin with the path of the REST application. For example, if the REST application is named "/csp/api", you could name this new application "/csp/api/login". Enable delegated authentication, and set the custom login page to your MyOAuth2.Login.cls page.

8. Set the Session Cookie Path on this new application to the same as that of the REST application: for example, "/csp/api". This will allow both applications to share a CSP session.

9. Add a CSP page to this new application that will act as a "home page". A user must first hit this page to establish their session. Here is an example that redirects to an endpoint on the REST API after login:

```
Class App.Home Extends %CSP.Page
{

ClassMethod OnPage() As %Status [ ServerOnly = 1 ]
{
    &html<<script type="text/javascript"> window.location="/csp/api/test" </script>>
    return $$$OK
}

}
```

10. Ensure the REST handler class has the UseSession parameter overridden to true.

```
Class API.REST Extends %CSP.REST
{

Parameter UseSession As BOOLEAN = 1;

XData UrlMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]
{
<Routes>
<Route Url="/test" Method="GET" Call="Test" Cors="true"/>
</Routes>
}

ClassMethod Test() As %Status
{
    write { "username": ($username) }.%ToJSON()
    return $$$OK
}

}
```

At this point, logging into the REST application will also "just work". The user will visit the home page, be redirected to SSO login, and finally go back to the REST app where they have an authenticated CSP session. As far as I can tell, this is the easiest way to add OpenID Connect to an IRIS REST app.

Another option is to use the "REST.ZAUTHENTICATE" sample from the security samples repo. This expects the front end to attach an OAuth 2.0 bearer token to every request. However, there's no defined way for the front end to get this access token. You will have to implement that OAuth flow yourself in JavaScript (or use a library like angular-oauth2-oidc.) You will also need to make sure the JavaScript app and the IRIS back end agree on all configuration items like the authorization server's issuer endpoint, the OAuth 2.0 client id, etc. I've found this is not a simple task.

I'm curious if anybody else is using OpenID Connect for authenticating an IRIS application. Is there an even simpler way? Or is it worth it to use the more complicated approach with bearer tokens? Let me know down below.

#Best Practices #OAuth2 #REST API #InterSystems IRIS
