

Artigo

[Danusa Calixto](#) · Out. 27, 2022 4min de leitura

## Usando OAuth 2.0 / OIDC para login único em uma aplicação REST no IRIS

Digamos que eu esteja desenvolvendo uma aplicação web que usa o IRIS como back-end. Estou trabalhando nela com acesso não autenticado. Está chegando a hora em que eu gostaria de implantar para os usuários, mas preciso adicionar a autenticação primeiro. Em vez de usar a autenticação padrão de senha do IRIS, quero que os usuários façam login com o Login Único (single sign-on - SSO, na sigla em inglês) da minha organização ou outro provedor de identidade popular, como Google ou GitHub. Li que o OpenID Connect é um padrão de autenticação comum e compatível com o IRIS. Qual é a maneira mais simples de colocar isso em funcionamento? ### Exemplo 1: uma app CSP simples A documentação [aqui](<https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls...>) apresenta uma opção bastante fácil para usar uma aplicação CSP como um cliente OpenID Connect. Estas são as etapas: 1. Configure o servidor OAuth 2.0 e a configuração do cliente no IRIS. Veja mais informações na seção "Caché configuration" do [excelente artigo do Daniel Kutac](<https://community.intersystems.com/post/intersystems-iris-open-authorized...>). 2. Copie a rotina OAUTH2.ZAUTHENTICATE do [repositório de amostras no GitHub](<https://github.com/intersystems/Samples-Security/blob/master/rtn/OAUTH2....>) para o namespace %SYS e renomeie como ZAUTHENTICATE. 3. Ative a autenticação delegada no sistema inteiro. 4. Crie uma página de login personalizada que seja uma extensão de %OAuth2.Login e substitua o método DefineParameters para especificar o nome do aplicativo OAuth 2.0 e os escopos:

```
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. Ative a aplicação web para a autenticação delegada e defina MyOAuth2.Login.cls como a página de login personalizada. 6. Um truque final: para a página de login personalizada funcionar, o usuário CSPSystem no IRIS precisa ter o acesso específico de LEITURA à base de dados em que está MyOAuth2.Login.cls. Neste momento, o login deve "simplesmente funcionar": a visita a uma página CSP nesta aplicação web redirecionará para a página de login no provedor de identidade. Depois de fazer o login, o usuário terá uma sessão CSP autenticada. O \$username será igual ao identificador de sujeito do SSO/Google/GitHub/etc., então posso usar a autorização integrada do IRIS para determinar a que dar acesso. ### Exemplo 2: o problema com REST E se a aplicação web estiver usando um manipulador REST? O processo acima não funciona. Se uma aplicação web estiver habilitada para REST, não é possível definir uma página de login personalizada. Descobri que são necessárias mais algumas etapas para contornar isso. 7. Crie uma aplicação web separada sem REST habilitado. O caminho nessa app precisa começar com o caminho do app REST. Por exemplo, se o nome da app REST for "/csp/api", você poderá nomear essa nova app como "/csp/api/login". Ative a autenticação delegada e defina a página MyOAuth2.Login.cls como a página de login personalizada. 8. Defina o Caminho do Cookie da Sessão nessa nova app para o mesmo da app REST: por exemplo, "/csp/api". Assim, ambas as aplicações compartilharão uma

sessão CSP. 9. Adicione uma página CSP a essa nova app para servir como uma "página inicial". Um usuário precisa primeiro acessar essa página para estabelecer sua sessão. Veja este exemplo que redireciona para um endpoint na API REST após o 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. Certifique-se de que a classe do manipulador REST tenha o parâmetro UseSession substituído por 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
}

}
```

Nesse ponto, o login na app REST também deverá "simplesmente funcionar". O usuário visitará a página inicial, será redirecionado para o login SSO e, finalmente, retornará a app REST, onde tem uma sessão CSP autenticada. Até onde sei, essa é a maneira mais fácil de adicionar o OpenID Connect a um app IRIS REST. Outra opção é usar a amostra "REST.ZAUTHENTICATE" do repositório de amostras de segurança. Com isso, espera-se que o front-end anexe um bearer token OAuth 2.0 a cada solicitação. No entanto, não há uma forma definida para o front-end obter esse token de acesso. Você terá que implementar esse fluxo OAuth por conta própria em JavaScript (ou usar uma biblioteca como [angular-oauth2-oidc](<https://github.com/manfredsteyer/angular-oauth2-oidc>)). Você também precisará se certificar de que o app JavaScript e o back-end IRIS estejam de acordo em todos os itens de configuração, como o endpoint do emissor do servidor de autorização, o ID do cliente OAuth 2.0 etc. Descobri que essa não é uma tarefa simples. Tenho curiosidade para saber se mais alguém está usando o OpenID Connect para autenticar uma app IRIS. Há uma maneira ainda mais simples? Ou vale a pena usar a abordagem mais complicada com bearer tokens? Comente aqui embaixo.

[#OAuth2](#) [#REST API](#) [#InterSystems IRIS](#)

---

URL de origem: <https://pt.community.intersystems.com/post/usando-oauth-20-oidc-para-login-%C3%BAnico-em-uma-aplicacao-%C3%A7%C3%A3o-rest-no-iris>

---