Article
Daniel Kutac · Feb 11, 2019 4m read

Using Oauth2 with SOAP (Web)Services

Hi guys,

Couple days ago, a customer approached me with the wish to enhance their existing legacy application, that uses SOAP (Web)Services so it shares the same authorization with their new application API based on REST. As their new application uses OAuth2, the challenge was clear; how to pass access token with SOAP request to the server.

After spending some time on Google, it turned out, that one of possible ways of doing so was adding an extra header element to the SOAP envelope and then making sure the WebService implementation does what is needed to validate the access token.

Luckily, we do have mechanism how to provide custom headers with SOAP request. So I just followed the documentation – see here for details – and came out with following classes.

The custom header class

```
Class API.SOAP.OAuth2Header Extends %SOAP.Header
{

Property accessToken As %String(MAXLEN = "");

}
```

Very simple, indeed. All I need is to pass access token but it can be extended to pass other information, too.

The WebService implementation class

```
/// API.SOAP.MyService
Class API.SOAP.MyService Extends %SOAP.WebService [ ProcedureBlock ]
{

/// Name of the WebService.
Parameter SERVICENAME = "MyService";

/// TODO: change this to actual SOAP namespace.
/// SOAP Namespace for the WebService
Parameter NAMESPACE = "http://tempuri.org";

/// Namespaces of referenced classes will be used in the WSDL.
Parameter USECLASSNAMESPACES = 1;
```

```
/// TODO: add arguments and implementation.
/// GetVersion
Method GetAccountBalance(pAccNo As %String) As API.SOAP.DT.Account [ WebMethod ]

{
  #define APP       "ANG RESOURCES"
  try {
    #dim tAccessTokenHeader as API.SOAP.OAuth2Header=..HeadersIn.GetAt("OAuth2Header"
)

    $$$THROWONERROR(tSC,##class(%SYS.OAuth2.AccessToken).GetIntrospection($$$APP,tAcc
essTokenHeader.accessToken,.jsonObjectAT))

    /* service specific check */
    // check whether the request is asking for proper scope for this service
    if '(jsonObjectAT.scope["account") set reason="scope not supported" throw

    if '(##class(%SYS.OAuth2.Validation).ValidateJWT($$$APP,tAccessTokenHeader.access
Token,,,.jsonObjectJWT,.securityParameters,.tSC)) {
      set reason="unauthorized access attempt"
      throw
    }
    set tAccountObject=##class(API.SOAP.DT.Account).%New()
    set tAccountObject.accno=pAccNo
    set tAccountObject.owner=jsonObjectJWT."acc-owner"
    set tAccountObject.balance=$random(200000)
  } catch (e) {
    set fault=..MakeFault($$$FAULTServer,"SECURITY",reason)
    Do ..ReturnFault(fault)
  }
  Quit tAccountObject
}

XData AdditionalHeaders

{
<parameters xmlns="http://www.intersystems.com/configuration">
<request>
<header name="OAuth2Header" class="API.SOAP.OAuth2Header"/>
</request>
</parameters>
}
}
```

Let ’ spend a few words about checking the access token. As you can see, the very first task we do is retrieving access token from the custom header and deserialize it into an object representation.

Then, it ’ s up to us, whether we check for scope, of we perform further validations, like validating JWT token (this depends on how we set-up the OAuth2 authorization server and whether we use OpenID which I highly recommend!)

Let ’ s have a look at client side now.

I won ’ t explain in details how you client receives access token for your Oauth2 authorization server, for this you can see my other articles; instead we just look at how we provide the access token to a Web Service client. (You generate WebService client class(es) by running standard SOAP Wizard / Client

option from your Atelier or Studio IDE)

Here is the code snippet from my client

```
set tWSClient=##class(Web.WSC.MyServiceSoap).%New()
set tWSHeader=##class(Web.WSC.s0.OAuth2Header).%New()

set tWSHeader.accessToken=accessToken
do tWSClient.HeadersOut.SetAt(tWSHeader,"access-token")
#dim tAccountObject as Web.WSC.s0.Account=tWSClient.GetAccountBalance(tAccNo)
```

Security setup considerations at the resource server

The easiest, but risky, was of setting up your CSP application at the resource server (WebServer) is make it allow unauthenticated users. They, it is always up to you to check in every service and every method for access token and validate it. If no access token is present or is not valid, you MUST return SOAP Fault.

The alternate way, but better, is to use delegated authentication and have *ZAUTHENTICATE* routine retrieved access token, assigning some deliberate username (can be taken from JWT if OpenID profile supplied with scope within access token request) with some minimal set of roles needed to execute webserver method.

#Best Practices #Security #SOAP #Caché #InterSystems IRIS
