Article
Vinicius Maranh... · Apr 2, 2020 4m read

Securing your APIs with OAuth 2.0 in InterSystems API Management – Part 1

Introduction

Nowadays, there is a lot of applications that are using Open Authorization framework (OAuth) to access resources from all kinds of services in a secure, reliable and efficient manner. InterSystems IRIS is already compatible with OAuth 2.0 framework, in fact, there is a great article in the community regarding OAuth 2.0 and InterSystems IRIS in the following link here.

However, with the advent of API Management tools, some organizations are using it as a single point of authentication, preventing unauthorized requests to arrive at downstream services and decoupling authorization/authentication complexities from the service itself.

As you may know, InterSystems has launched its API Management tool, called InterSystems API Management (IAM), which is available with IRIS Enterprise license (not IRIS Community Edition). Here is another great post in the community introducing InterSystems API Management.

This is the first part of 3-part series of articles that will show how you can use IAM to simply add security, according to OAuth 2.0 standards, to a previously unauthenticated service deployed in IRIS.

In this first part, will be provided some OAuth 2.0 background together with some IRIS and IAM initial definitions and configurations in order to facilitate the understanding of the whole process of securing your services.

After the first part, this article series will approach two possible scenarios to secure your services with IAM. In the first scenario, IAM will only validate the access token present in the incoming request and will forward the request to the backend if the validation succeeds. In the second scenario, IAM will both generate an access token (acting as an authorization server) and validate it.

Therefore, the second part will discuss and show in detail the steps needed to configure the scenario 1 and the third part will discuss and demonstrate the configurations for scenario 2, together with some final considerations.

If you want to try IAM, please contact your InterSystems Sales Representative.

OAuth 2.0 Background

Every OAuth 2.0 authorization flow basically consists of 4 parties:

1. User
2. Client
3. Authorization Server
4. Resource Owner

For the sake of simplicity, this article will use the " Resource Owner Password Credentials " OAuth flow, but you can use any OAuth flow in IAM. Also, this article will not specify any scope.
Note: You should only use Resource Owner Password Credentials flow when the client app is highly trusted, as it directly handles user credentials. In most cases, the client should be a first-party app.

Typically, the Resource Owner Password Credentials flow, follows these steps:

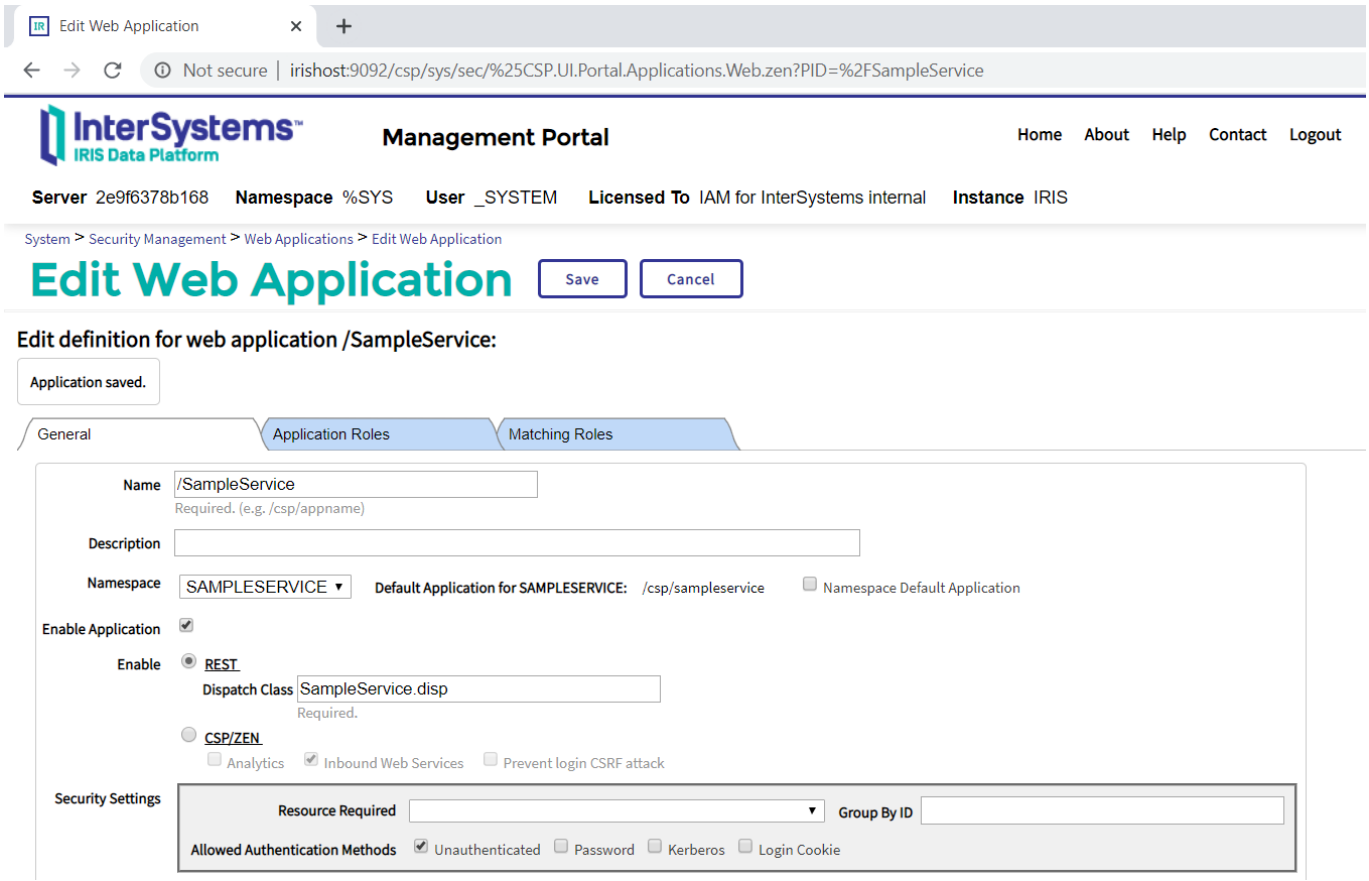
1. The user enters their credentials (for example username and password) in the client app
2. The client app sends the user credentials together with its own identification (client id and client secret, for example) to the authorization server. The authorization server validates the user credentials and client identification and returns an access token
3. The client uses the token to access resources on the resource server
4. The resource server validates the access token received before returning any information to the client

With that in mind, there are two scenarios where you can use IAM to deal with OAuth 2.0:

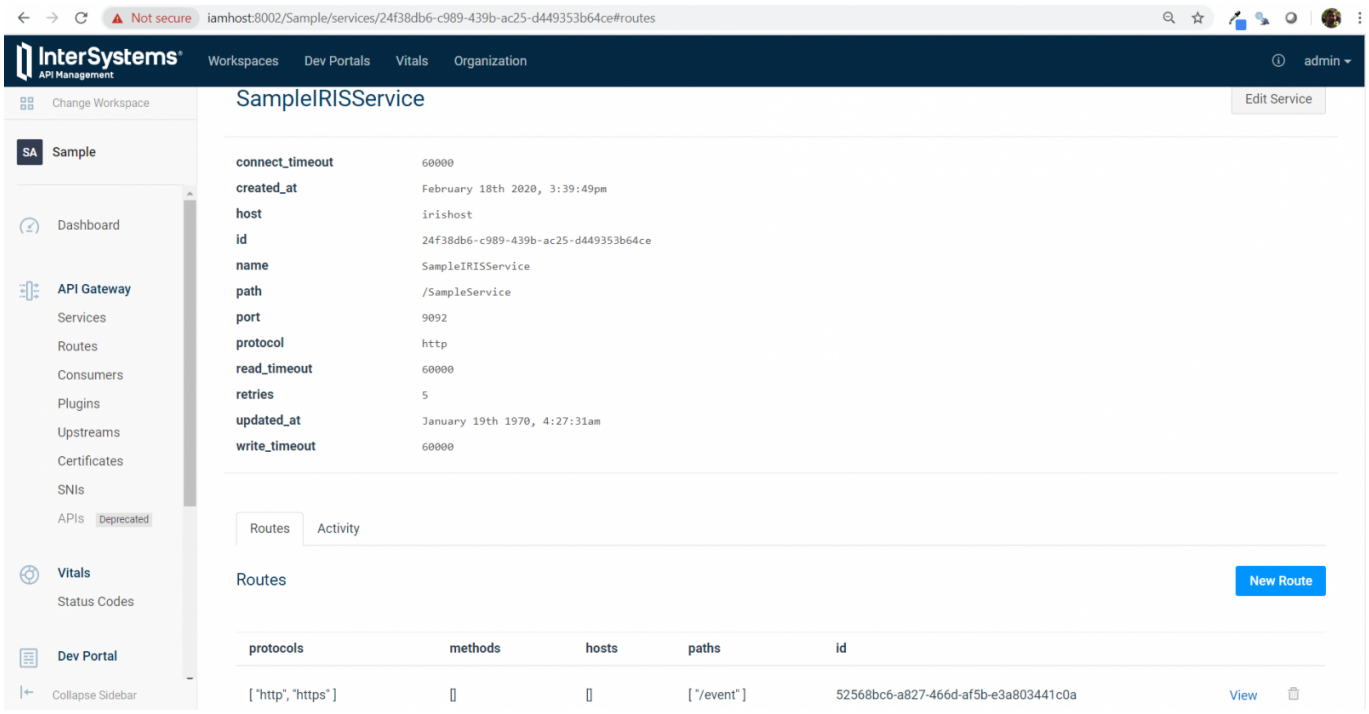
1. IAM acting as a validator, verifying the access token provided by the client app, forwarding the request to the resource server only if the access token is valid; In this case the access token would be generated by a third-party authorization server
2. IAM acting both as an authorization server, providing access token to the client, and as an access token validator, verifying the access token before redirecting the request to the resource server.

IRIS and IAM definitions

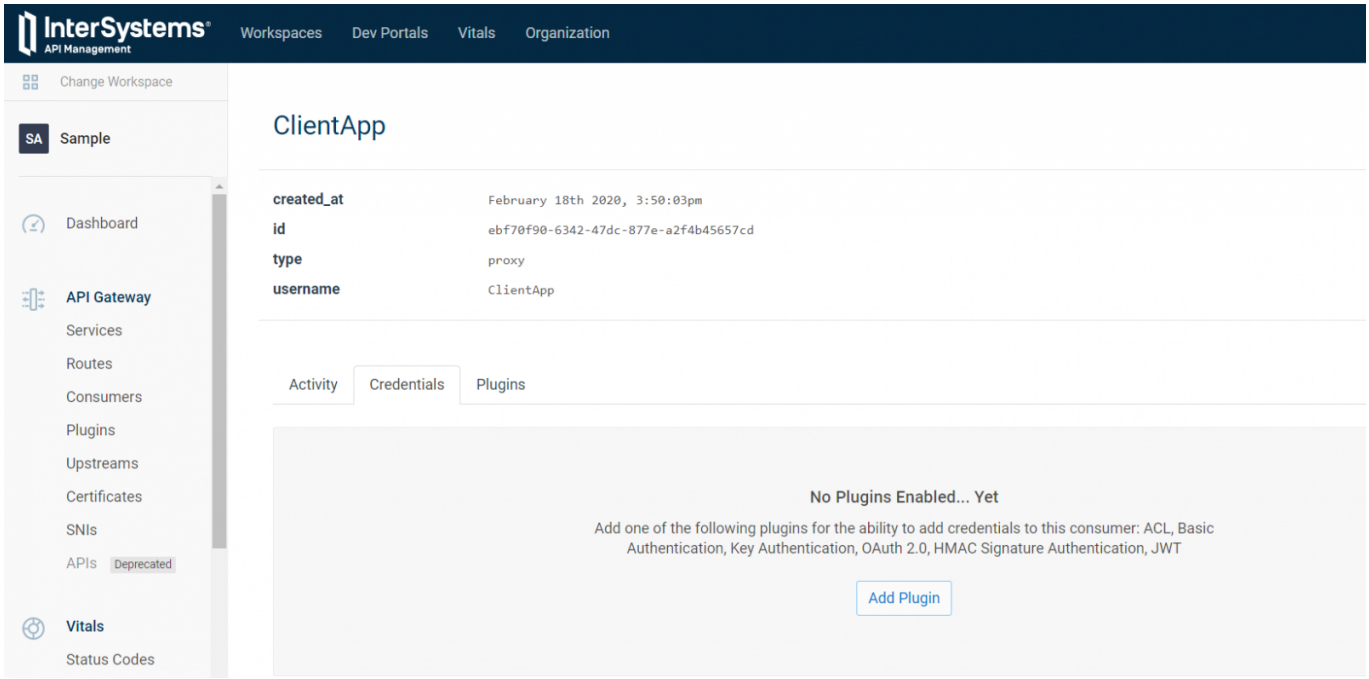
In this post, it will be used an IRIS Web Application called " /SampleService " . As you can see from the screenshot below, this is an unauthenticated REST service deployed in IRIS:



Furthermore, in IAM side is configured a Service called " SampleIRISService " containing one route, as you can see in the screenshot below:



Also, in IAM is configured a consumer called " ClientApp " , initially without any credentials, to identify who is calling the API in IAM:



With the configurations above, IAM is proxying to IRIS every GET request sent to the following URL:

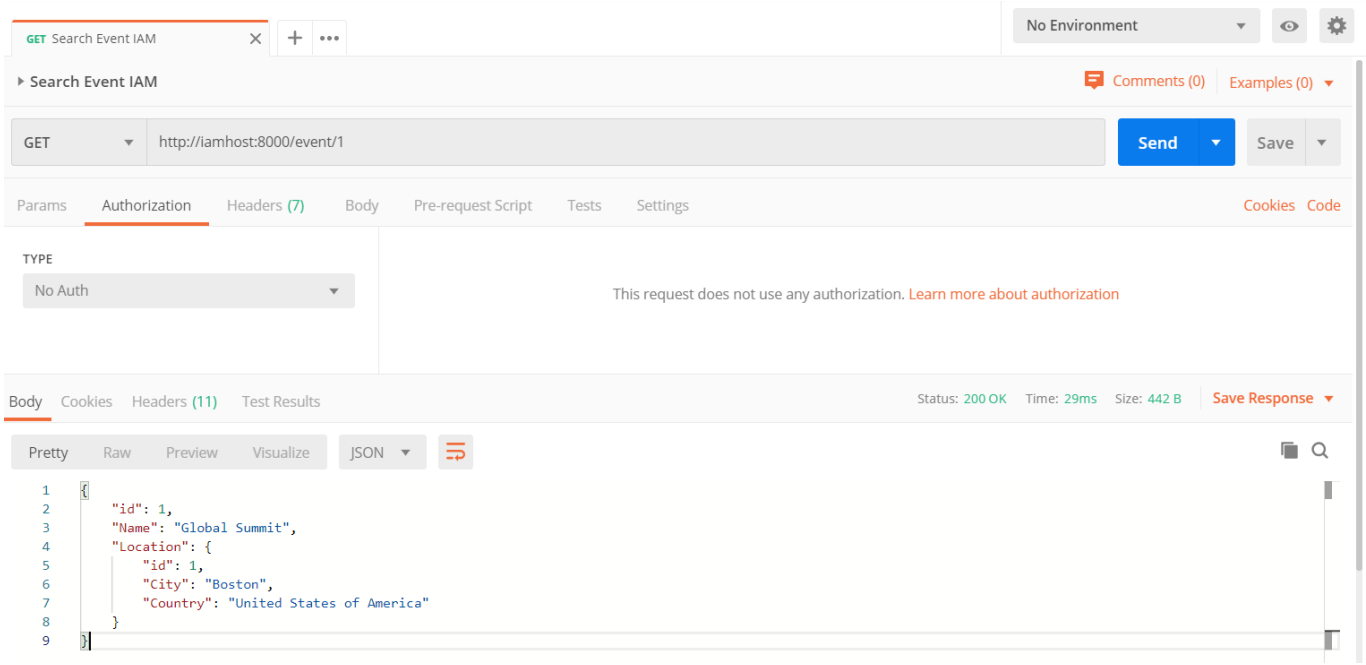
http://iamhost:8000/event

At this point, no authentication is used yet. Therefore, if we send a simple GET request, with no authentication, to the URL

http://iamhost:8000/event/1

we get the desired response.

In this article, we are going to use an app called " PostMan " to send requests and check the responses. In the PostMan screenshot below, you can see the simple GET request together with its response.



Continue reading to the second part of this series to understand how to configure IAM to validate access tokens present in the incoming requests.

#API #OAuth2 #REST API #Security #InterSystems IRIS

Source URL:https://community.intersystems.com/post/securing-your-apis-oauth-20-intersystems-api-management-%E2%80%93-part-1