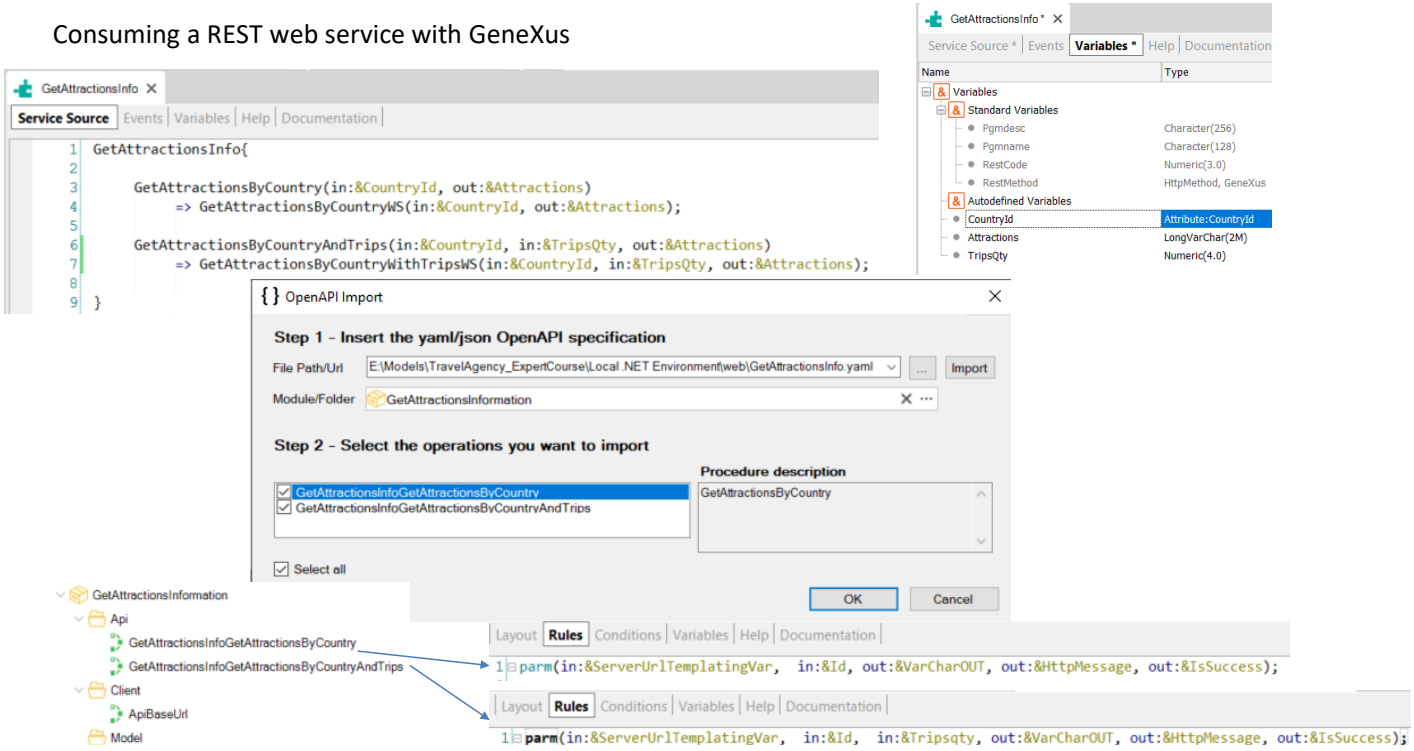
Web Services. Advanced topics

Consuming REST services with GeneXus

GeneXus™

In this video, we will see how to consume REST services with GeneXus using the OpenAPI protocol made by third parties or published from GeneXus.
In particular, how we can invoke HTTP methods of REST services, or how to consume a secure REST service.

Consuming a REST web service with GeneXus



We are going to consume the GetAttractionsByCountry service, which we built by publishing the GetAttractionsByCountryWS procedure as a service, using the GetAttractionsInfo API object that we saw earlier.

To import the REST service definition, we go to Tools / Application Integration / OpenAPI import and write the URL or the file path of the JSON file with the Swagger specification of the REST service. Swagger is a set of open source software tools for designing, building, documenting, and using REST services that was developed by SmartBear Software and includes automated documentation, code generation, and test case generation.
The file that we are going to import with this specification can have a .JSON or .YAML extension, which is a superset of JSON.

If, when publishing our REST service, we set the Generate Open API interface property to True, available in the API object or in the procedure exposed as REST, the Swagger specification file with .YAML extension is automatically generated in the Environment Web folder.
The Swagger file that we import can have an OpenAPI specification version 2 or 3. As of version 17 upgrade 6, GeneXus supports both version 2 and version 3 of the OpenAPI specification.

Continuing with the example, in the dialog box where we are asked for the path, we look for the file GetAttractionsInfo.yaml in the Web folder of our active environment. In Module/Folder, we write the name of a module that we created before to contain everything that we import. This is good practice, in case we import an object that has the same name as an existing object in the KB.

We press the Import button and see that the wizard finds the two services that we had exposed with the API object.
We click on Select All and then OK.

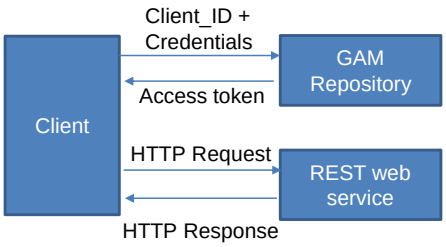
If we open the module, we see that there are 3 folders, one called API containing two procedure objects with the name of the services, which we are going to execute to invoke the services. Also, a Client folder containing a procedure called APIBaseURL that returns the Base URL that will be used to invoke the service, and that we can change if we want, as well as a Model folder that in this case is empty, because the previous methods do not return any SDT.

In the rules of the procedures, we find the input parameters, where the &ServerUrlTemplatingVar variable is present in all the procedures that are consumed, and the others are the ones we recognize. As output parameters, we have the &VarCharOUT variable that will contain the requested information and the &HttpMessage and &IsSuccess variables that we can use to have information on the execution of the service.

Consuming secure REST web services with GeneXus

- Client identification
- Users & Passwords
- Permissions

GeneXus Access Manager




```
//First get the access_token
&httpclient.Host= &server
&httpclient.Port = &port

&GAMOauthAdditionalParameters.ClientId = "f719771ad52a42919a221bc796d0d6b0"
&GAMOauthAdditionalParameters.ClientSecret = &ClientSecret
&GAMLoginAdditionalParameters.AuthenticationTypeName = !"local"
&AccessTokenSDT = GAMRepository.GetOauthAccessToken(!"admin", !"admin123",&GAMLoginAdditionalParameters,&GAMOauthAdd
itionalParameters,&GAMSession,&GAMErrors)

//call DPProduct web service
&httpclient.BaseUrl = &urlbase + '/rest/'
&httpclient.AddHeader("Content-Type", "application/json")
&httpclient.AddHeader('Authorization','OAuth ' + &AccessTokenSDT.access_token)
&httpclient.AddHeader("GENEXUS-AGENT","SmartDevice Application")
&httpclient.Execute('GET','DPProduct')
```

Just as we stressed the importance of security for SOAP services, we must do the same for REST services.

Secure REST services are based on the Oauth security scheme and this involves defining the Client; that is, the application, the users (UserId and UserPassword) and the permissions (Read, Write, FullControl, etc.).
In GeneXus, this is implemented through the GAM, with authentication based on Oauth version 2.0.

When we expose a procedure, a data provider, or a business component as a REST service, if GAM is applied, the REST service is identified as an application within the GAM repository. To give access to the service, we must configure the roles, users, and permissions of the service application and then provide the client identifier (Client_Id) of the application, username, and password to the consumer of the service.

Before invoking the service, the client must obtain an access token. For that, it must make a POST to the GAM repository with the Client_ID and the access credentials provided before. The GAM response will be a JSON with the access token and type of permission (FullControl, etc.).
This invocation to the GAM repository can be made using the GetOauthAccessToken() method of the GAM API.

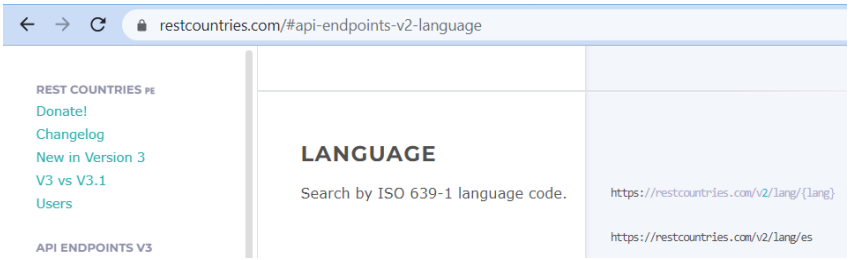
Once the token is obtained, a variable of HTTPClient type must be used to consume the REST service. Here is an example of consuming a secure REST service.

More information can be found in the wiki, in the article “HowTo: Develop Secure REST Web Services in GeneXus.”

Customizing the consumption of a REST web service

Customizing the consumption of a REST web service

&HttpClient.Execute("GET", ...)
& HttpClient.Execute("POST", ...)
& HttpClient.Execute("PUT", ...)
& HttpClient.Execute("DELETE", ...)




```
//REST API: https://restcountries.com/v3.1/all?fields=name
&httpclient.Host = "restcountries.com"
&httpclient.Port = 443 // for https
&httpclient.Secure = 1
&httpclient.BaseUrl = "/v2/"
&httpclient.AddHeader("Content-Type", "application/json")
&httpclient.Execute("GET","lang/es")

if &httpclient.StatusCode = 200
     &result = &httpclient.ToString()
else
     msg("Error: " + &httpclient.StatusCode.ToString())
endif
```

Just as we customized how SOAP services are consumed, let's see how we can customize the way a REST service is consumed.

Although it is recommended to import the definitions of a REST service with the Import OpenAPI wizard we saw, sometimes the service information file (.yaml extension) is not available. In these cases, it is possible to invoke the HTTP methods: GET, PUT, POST, and DELETE, using a variable of HTTPClient type.

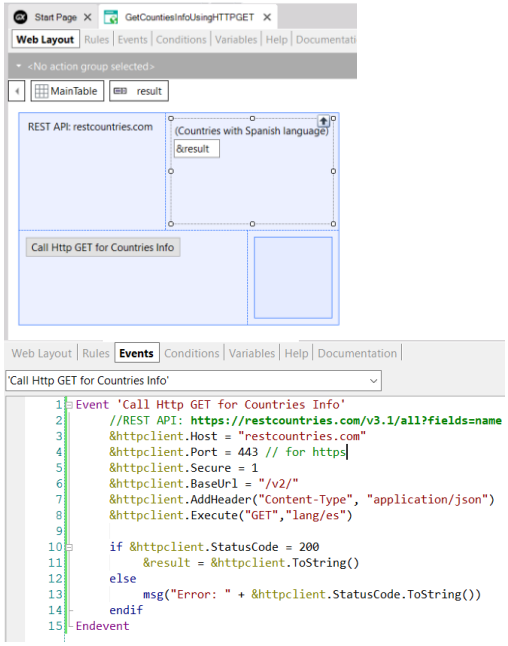
Let's see an example for invoking a public REST API that returns data about countries.

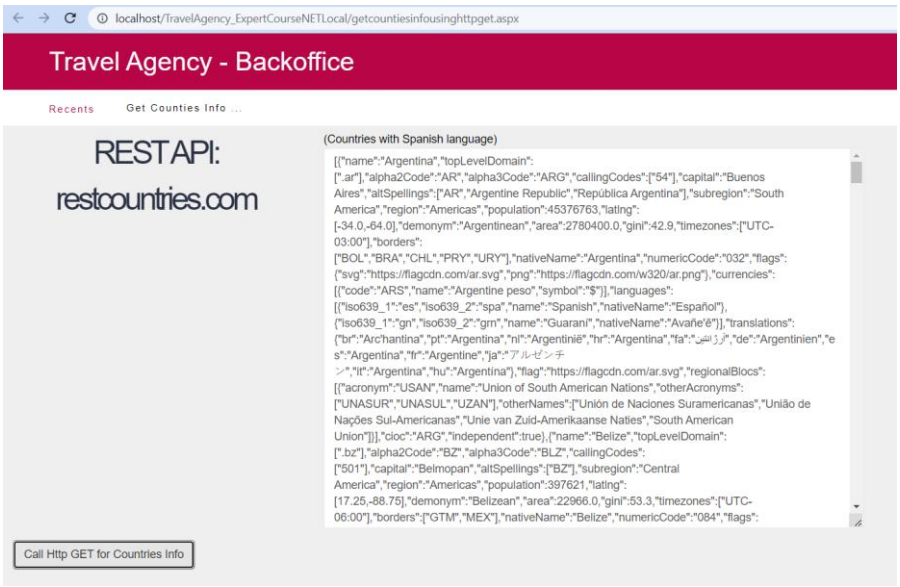
To invoke the web service, first we create the variable and then we assign the properties to it: Host, Port, Secure, and BaseUrl.
Next, we add the header of JSON type and invoke the Execute method, passing the method we want to use and the parameters required by the service. In this case, we are passing the language because we want to retrieve the Spanish-speaking countries.

After the invocation, we process the status code returned. If it is 200, we get the JSON string and otherwise we give an error message.

To use the other HTTP methods, we replace the parameters of the Execute method with the HTTP method we want and use the appropriate parameters according to the method. For example, in a DELETE we must pass the identifier of the record we want to delete.

Customizing the consumption of a REST web service





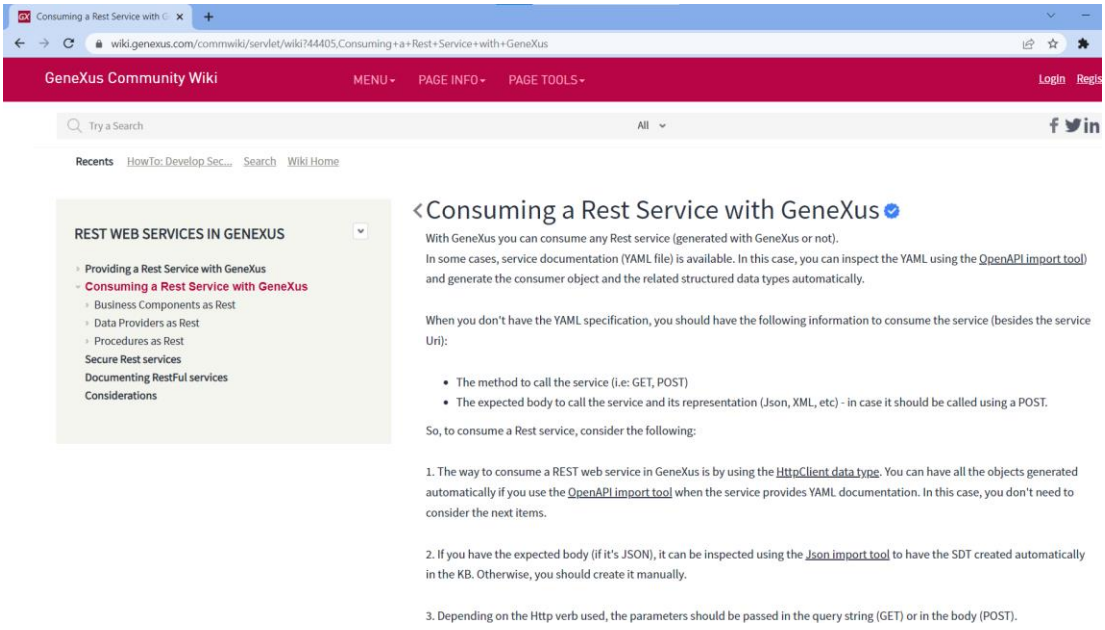
Let's run this example in GeneXus.

We have created the web panel GetcountriesInfoUsingHTTPGET, and included a button to invoke the service in the web layout, and a &result variable to show the JSON that we will obtain.

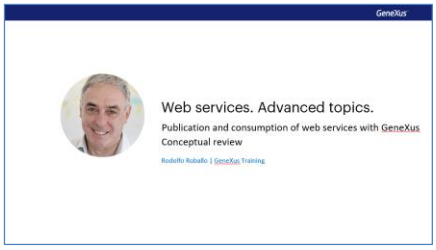
In the button event, we see the code that we already explained.

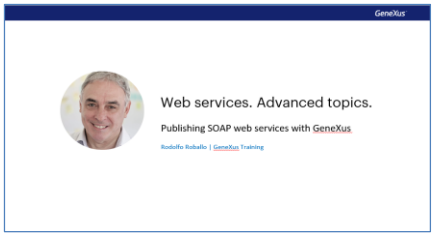
We execute the web panel that is set as main... we press the button and receive the information of the Spanish-speaking countries, as we wanted.

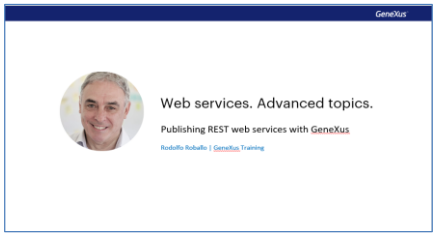
Customizing the consumption of a REST web service

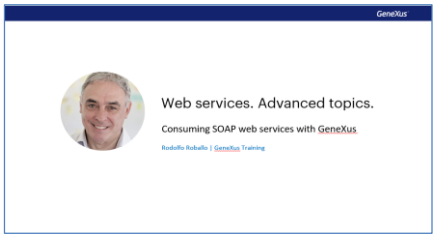


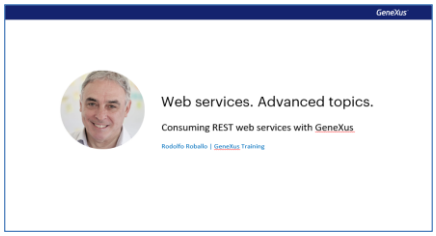
For more information, read the wiki article: “Consuming a Rest Service with GeneXus.”











In these videos about web services with GeneXus, we try to address the most common use cases of publishing and consuming both SOAP and REST services, using the simplest examples and in situations where customization is required.

Feel free to explore this and other related topics in more detail in our Wiki.

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications