Connect SIF3 Framework Consumers to RICOne API

# Section 1: RICOne Environment

To be able to connect your consumer to the RICOne API you must have a connection/endpoint to a RICOne OAuth Server as well as the actual endpoint URL to the RICOne API (SIF3 Provider). You will need to contact a RICOne representative to give you the information pieces, listed below. They will provide you with information about their OAuth Server and RICOne API Endpoint so that you are able to connect your consumer, written with this SIF3 Framework, to the RICOne environment:

* OAuth Server endpoint or login. This will most likely be something of the form of https://auth.test.ricone.org/login
* Client ID (also referred to as Application Key)
* Password for OAuth Server
* Provider ID
* RICOne API Environment ID (This is the SIF Environment ID and will be a UUID)
* RICOne SolutionID (This is the SIF Solution ID and will be some identifier)
* RICOne API Base URL. This will most likely be something of the form of https://sandbox.ricone.org/api

Once you have the above information you are able to configure your consumer to connect to the RICOne API. The next section states where each information piece listed above, is placed to make your consumer connect using the SIF3 Framework.

Finally you need the RICOne Client API Library. This library will be provided to you with the above information. The rest of this document assumes that you have all of that.

# Section 2: Configure SIF3 Framework

To configure the SIF3 Framework to connect to the RICOne API environment listed in previous section you need to do the following steps:

**Step 1**: Set RICOne Solution ID in Environment Template

In your project you should have a directory where the consumer’s SIF3 Environment templates are stored. This would generally be at the following location <installDIR>/config/environments/consumer/template. There is most likely an environment template called RICOne.xml, demo.xml or devLocal.xml. If the RICOne.xml doesn’t exist then perform action 1 below before you go to action 2 :

1. Create a copy of one of these templates in the same directory and name it something like RICOne.xml
2. Open the RICOne.xml file and put the value of the “RICOne SolutionID” as seen from section 1 into the <solutionId> node. Set the <authenticationMethod> node to “**Bearer**” Your RICOne.xml file should look something like this:

<environment xmlns="http://www.sifassociation.org/infrastructure/3.1">

<solutionId>***{RICOne SolutionID}***</solutionId>

<authenticationMethod>**Bearer**</authenticationMethod>

<instanceId/>

<userToken/>

<consumerName></consumerName>

<applicationInfo>

<applicationKey></applicationKey>

<supportedInfrastructureVersion>3.1</supportedInfrastructureVersion>

<dataModelNamespace>http://www.sifassociation.org/datamodel/na/3.3</dataModelNamespace>

<transport>REST</transport>

<applicationProduct>

<vendorName>…</vendorName>

<productName>Test Driver</productName>

<productVersion>0.1alpha</productVersion>

</applicationProduct>

</applicationInfo>

</environment>

You can change the values under the <applicationProduct> node to any value that is applicable to you. Leave everything else as is.

1. Save your RICOne.xml file.

**Step 2**: Configure the consumer’s properties file.

The consumer’s property file can be found in the directory <installDIR>/config/consumers. Let’s assume you have a properties file called RicOneConsumer.properties. You need to set a few properties in that properties file with the values given to you by the RICOne contacts as mentioned in section 1 of this document. The table below states the name of the property and what it must be set to.

|  |  |
| --- | --- |
| **Property Name** | **Value** |
| env.xml.file.name | Name of xml file created in Step 2 (i.e. RICOne.xml) |
| env.application.key | Value of “Client ID” from Section 1 |
| env.pwd | Value of “Password for OAuth Server” from Section 1 |
| env.baseURI | Value of “RICOne API Base URL” from Section 1. Add the following to this base URL: /environments/environment. The final URL should look something like: https://sandbox.ricone.org/api/environments/environment |
| env.authentication.method | Bearer |
| env.use.existing | true |
| env.existing.environmentURI | Value of “RICOne API Base URL” from Section 1 and append /environments/<environment ID> from Section 1. The final URL should look something like this: https://sandbox.ricone.org/api/environments/<environmentID> |
| env.create.conflictIsError | false |

**Step 3**: Connect OAuth Server to your Consumer.

For the SIF3 Framework to be able to connect to the RICOne OAuth server the framework must implement an instance of the abstract class called AbstractSecurityService using the RICOne Client API (see also section 5.10.3 of the Developer’s Guide of the SIF3 Framework for details about external security services). An example of such an implementation is provided with this document and is called RICOneSecurityService. You can use this class with your project. You may want to place it into a different Java package, though. Ensure that you have the RICOne Client API library and its dependent 3rd Party libraries as well and place them into the appropriate directory of your Java project. The RIC One representative can provide you with these libraries.

Now you need to configure the consumer properties file to use that security class. Set the following properties in the RicOneConsumer.properties file:

|  |  |
| --- | --- |
| **Property Name** | **Value** |
| adapter.security.service | Fully qualified class name of the RICOneSecurityService class. I.e. systemic.sif3.demo.security.RICOneSecurityService |
| security.service.property.ricOne.authUrl | Value of “OAuth Server endpoint” from Section 1. This will look something like this: https://auth.test.ricone.org/login |
| security.service.property.ricOne.providerId | Value of “Provider ID” from Section 1 |

**Step 4**: Start your Consumer

After you have applied the configurations in the previous steps you should be able to start your consumer. Please ensure that your consumer uses the correct properties file (the one you changed in step 2 & 3 above). Verify this by looking at the consumer executable and ensure that the ConsumerLoader is initialised with the correct property file (example below):

ConsumerLoader.initialise("**RicOneConsumer**");

Note: You **MUST NOT** provide the “.properties” file extension!