SIF 3.0 Framework (.NET)

Version

**Author:** Rafidzal Rafiq, SIF Solution Engineer

**Revision:** 1.0

**Published:** Nov 2014

Copyright © 2014,

Table of Contents

1. Introduction 3

2. Exercise 1: SIF 3 Framework Installation & Verification 3

2.1. Download the SIF 3 Framework 3

2.2. Load the SIF 3 Framework into Visual Studio 3

2.3. Create an Environment 4

2.4. Start the Environment Provider 4

2.5. Start the demo K12Student Provider 4

2.6. Run the demo K12Student Consumer 5

2.7. Review expected behaviour 5

3. Exercise 2: First Consumer 5

3.1. Consumer pre-requisites and configuration 5

3.1.1. log4net 5

3.1.2. AutoMapper 5

3.1.3. SIF 3 Framework libraries 6

3.1.4. SIF 3 Framework configuration 6

3.2. Define the K12Student data model 6

3.3. Implement the K12Student Consumer 6

3.4. Implement a console application 7

3.5. Test the K12Student Consumer 7

3.6. Advanced exercise 7

4. Exercise 3: CRUD Provider – School 7

4.1. Consumer pre-requisites and configuration 7

4.1.1. NHibernate 7

4.1.2. SQLite 7

4.1.3. AutoMapper 8

4.1.4. SIF 3 Framework libraries 8

4.1.5. Web API specific configuration 8

4.2. Define the School data model 8

4.3. Implement the School service interface 9

4.4. Implement the School Provider/Controller 9

4.5. Create a new Environment for the School Provider 9

4.6. Start the School Provider 9

4.7. Test the School Provider 9

4.7.1. Postman (Chrome Plugin) 9

4.7.2. Implement a School Consumer (optional) 11

4.8. Advanced Exercise 11

5. Important notes 11

5.1. Issues with the Environment Provider 11

# Introduction

Before attempting the exercises in this document, it is assumed that the following documents have already been read:

* Sif3Framework .NET Developer’s Guide
* Sif3Framework .NET Demo Usage Guide

The exercises presented here run the demo projects included with the SIF 3 Framework, and copy heavily from the information presented in the above documents.

# Exercise 1: SIF 3 Framework Installation & Verification

**Task:**

* Download the SIF 3 Framework
* Load the SIF 3 Framework into Visual Studio
* Create an Environment
* Start the Environment Provider
* Start the demo StudentPersonal Provider
* Run the demo StudentPersonal Consumer

## Download the SIF 3 Framework

The SIF 3 Framework can be downloaded from GitHub. To do so, use the following link and click on the “Download ZIP” button.

<https://github.com/nsip/Sif3Framework-dotNet>

Once downloaded, extract to an appropriate directory.

## Load the SIF 3 Framework into Visual Studio

The Visual Studio Solutions for this framework can be found under the Code directory. For the demo projects to be run, the Sif3Framework and Sif3FrameworkDemo Solutions need to be loaded into Visual Studio and built. If loaded for the first time, Visual Studio may require time to retrieve associated third party libraries that have been referenced via NuGet.

Once opened in Visual Studio, no additional configuration should be required to run the demo projects. This framework was developed using Visual Studio 2013, but should load in the Express version.

## Create an Environment

Before a Service Consumer and Object Service Provider can interact, an Environment must be created to manage the session information between them. Once an Environment has been defined, then the Service Consumer and Object Service Provider are able to register to that Environment. This registration is managed by an Environment Provider.

Environment definition generally falls under the domain of a SIF Administrator. However, creation of an initial Environment for this demo is performed by running the *Scripts\BAT\Demo execution\DemoAuSetup.bat* script. This script uses the Sif.Framework.Demo.Setup project to create and populate a demo database with an initial Environment definition. A SQLite database is used as no configuration changes are required to be able to run this demo out of the box.

All mandatory information associated with the applications (e.g. applicationKey, sharedSecret) have been predefined for these demos.

## Start the Environment Provider

Open the Sif3Framework Solution in Visual Studio and simply run it (by clicking on the Internet Explorer button shown below).

Internet Explorer button

If the Internet Explorer button is not visible, it will be necessary to make the Sif.Framework.EnvironmentProvider project of the Solution the single start-up project.

The Sif.Framework.EnvironmentProvider project has been configured to run in IIS Express on a particular port (as specified in the Visual Studio Project properties). This port is referenced in the Environment configured from the previous section.

For the moment, ignore the “HTTP Error 403.14 – Forbidden” message that appears in the popped up Internet Explorer window on start-up. As this is a Web API service, the fact that it does not have a home page is not relevant for this demo.

## Start the demo StudentPersonal Provider

Load the Sif3FrameworkDemo Solution into Visual Studio, set the single start-up project to Sif.Framework.Demo.Au.Provider and simply run it (by clicking on the Internet Explorer button shown below).

Internet Explorer button

As with the Sif.Framework.EnvironmentProvider, the Sif.Framework.Demo.Au.Provider project has been configured to run in IIS on a particular port.

## Run the demo StudentPersonal Consumer

Once the Environment Provider and StudentPersonal Provider have been successfully started, run the *Scripts\BAT\Demo execution\DemoAuConsumer.bat* script to start the StudentPersonal Consumer. This script runs the Sif.Framework.Demo.Au.Consumer Project of the Sif3FrameworkDemo Solution.

## Review expected behaviour

The scenario illustrated in this demo is one where a Consumer is making a call for student data in a Direct Environment. As such, once this data has been consumed, the demo has been completed.

In this case, the StudentPersonal Consumer simply prints the (10) student details to the console. Once the details have been printed, the Environment Provider and StudentPersonal Provider instances can be stopped.

# Exercise 2: First Consumer

**Task:**

* Implement a StudentPersonal Consumer from scratch

## Consumer pre-requisites and configuration

Even though the demo projects that come with the SIF 3 Framework already include a StudentPersonal Consumer, it is useful to implement your own to understand the third party libraries and configuration required for implementing a Consumer from scratch. To this end, in Visual Studio create a new Solution and new Console Application Project.

### log4net

Using “Manage NuGet Packages…”, add the log4net package to this project. Instructions on the use of log4net fall outside the scope of this document. However, the following paragraphs summarise log4net configuration required for log output.

Add the following line of code to the end of the AssemblyInfo.cs class.

[assembly: log4net.Config.XmlConfigurator(Watch = true)]

Add appropriate log4net Appenders and Loggers to App.config. For those new to log4net, it would be simpler if the App.config file from the demo Consumer project is copied.

### AutoMapper

Under normal circumstances, the AutoMapper package would be added in the same manner as the log4net installation. However, due to an issue encountered when using the current release of AutoMapper (3.2.1), the version used in this framework is AutoMapper 3.1.1. To add this version of AutoMapper to new projects, within Visual Studio select the Consumer project and run the following command in the Package Manager Console.

PM> Install-Package AutoMapper -Version 3.1.1

### SIF 3 Framework libraries

Add the following libraries from the SharedLibs directory included with this Framework:

* Sif.Framework 0.15.0\Sif.Framework.dll
* Sif.Specification.Infrastructure 3.0.1\Sif.Specification.Infrastructure.dll
* Sif.Specification.DataModel.Au 1.3\Sif.Specification.DataModel.Au.dll

### SIF 3 Framework configuration

For a Consumer to run, certain configuration settings are required. These settings identify the Consumer to the Environment Provider.

The easiest way to define these settings is to copy the SifFramework.config file from the demo Consumer project and reference it in the App.config file using:

<appSettings file="SifFramework.config">

</appSettings>

For these exercises, leave the settings as they are since they already correspond to the Environment information stored in the demo database (used by the Environment Provider). For more information on these configuration settings, refer to the “Sif3Framework .NET Demo Usage Guide” document.

## Define the StudentPersonal data model

For implementing a Consumer, a SIF data object needs to be specified. In this case, the StudentPersonal data model will be used. A requirement for the use of this Framework is that the model object used has to implement the IPersistable interface as well as be a SIF data model. For an example implementation, refer to the Consumer demo project.

Create a new class (e.g. StudentPersonal.cs) and implement the IPersistable interface (where the Id property equates to the RefId of the SIF data object).

It is important that the model object used will serialise (XML) to meet the AU 1.3 Data Model Specification. To assist with this requirement, the Sif.Specification.DataModel.Au.v1\_3 Project can be used as a reference for any AU 1.3 data model objects used.

## Implement the StudentPersonal Consumer

To implement the Consumer, create a new class (e.g. StudentPersonalConsumer.cs) that extends the GenericConsumer class with the previously defined data model class (e.g. StudentPersonal.cs) as the generic type. Implement the constructors to simply call upon the “base” constructors of the GenericConsumer class.

## Implement a console application

Create a class with a Main() method that will be run the StudentPersonal Consumer. In the Main() method, instantiate the new Consumer class (e.g. StudentPersonalConsumer.cs) by calling the constructor that takes multiple string parameters, and pass it a parameter of “Sif3DemoApp” (to match the Environment settings used by the existing demo Provider project).

Call the Register(), Retrieve() and then Unregister() methods, writing the student details retrieved to the console.

## Test the StudentPersonal Consumer

To test the newly created StudentPersonal Consumer, repeat Exercise 1 replacing the demo Consumer with this Consumer.

## Advanced exercise

If you have successfully completed this exercise and have more time, why don’t you try to “Update” a student?

# Exercise 3: CRUD Provider – SchoolInfo

**Task:**

* Implement a SchoolInfo Provider (or any other object of your choice)
* Test using Postman, or implement a SchoolInfo Consumer

## Consumer pre-requisites and configuration

In Visual Studio create a new Solution and new ASP.NET Web Application Project (an empty project set for Web API). For those new to Web API, refer to the following web page:

<http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api>

### NHibernate

Using “Manage NuGet Packages…”, add the NHibernate package to this project. Instructions on the use of NHibernate fall outside the scope of this document.

Copy the SifFramework.cfg.xml NHibernate configuration file from the demo Provider project into this project. For the sake of this exercise, leave the SifFramework.cfg.xml unchanged to use the existing SQLite demo database (defined using relative paths).

### SQLite

Using “Manage NuGet Packages…”, add the SQLite package to this project. Instructions on the use of SQLite fall outside the scope of this document.

### AutoMapper

Under normal circumstances, the AutoMapper package would be added in the same manner as the NHibernate and SQLite installations. However, due to an issue encountered when using the current release of AutoMapper (3.2.1), the version used in this framework is AutoMapper 3.1.1. To add this version of AutoMapper to new projects, within Visual Studio select the Provider project and run the following command in the Package Manager Console.

PM> Install-Package AutoMapper -Version 3.1.1

### SIF 3 Framework libraries

Add the following libraries from the SharedLibs directory included with this Framework:

* Sif.Framework 0.15.0\Sif.Framework.dll
* Sif.Specification.Infrastructure 3.0.1\Sif.Specification.Infrastructure.dll
* Sif.Specification.DataModel.Au 1.3\Sif.Specification.DataModel.Au.dll

### Web API specific configuration

Due to the different methods the .NET Framework uses to XML serialise objects, the Global.asax.cs file that comes with the Project needs to be updated to specify a particular serialisation method. This is achieved by the following lines of code:

XmlMediaTypeFormatter formatter =

GlobalConfiguration.Configuration.Formatters.XmlFormatter;

formatter.UseXmlSerializer = true;

In addition, due to the default behaviour of the inherent XML serialiser used by Web API, the additional lines of code (below) are also required in the Global.asax.cs. Without these lines, the SchoolInfosController would return a list of school records with a root element of <ArrayOfSchoolInfo> instead of the required <SchoolInfos>.

ISerialiser<List<SchoolInfo>> schoolInfosSerialiser =

SerialiserFactory.GetXmlSerialiser<List<SchoolInfo>>

(new XmlRootAttribute("SchoolInfos"));

formatter.SetSerializer<List<SchoolInfo>>((XmlSerializer)schoolInfosSerialiser);

## Define the SchoolInfo data model

For implementing a Provider, a SIF data object needs to be specified. In this case, the SchoolInfo data model will be used. A requirement for the use of this Framework is that the model object used has to implement the IPersistable interface as well as be a SIF data model. For an example implementation (of a student), refer to the Provider demo project.

Create a new class (e.g. SchoolInfo.cs) and implement the IPersistable interface (where the Id property equates to the RefId of the SIF data object).

It is important that the model object used will serialise (XML) to meet the AU 1.3 Data Model Specification. To assist with this requirement, the Sif.Specification.DataModel.Au.v1\_3 Project can be used as a reference for any AU 1.3 data model objects used.

## Implement the SchoolInfo service interface

Create a class (e.g. SchoolInfoService) that implements the IGenericService interface with the previously defined data model class (e.g. SchoolInfo.cs) as the generic type. For these exercises, it would be sufficient to only implement the Retrieve() method.

## Implement the SchoolInfo Provider/Controller

To implement the Provider, create a new class (e.g. SchoolInfosController.cs) that extends the GenericController class with the previously defined data model class (e.g. SchoolInfo.cs) as the generic type. In the default constructor, call upon the “base” constructor of the GenericController class that accepts a service interface (in this case, the newly created SchoolInfoService).

As the Web API specification relies heavily on coding convention, a couple of very important points to take into account when implementing a Provider:

1. The Web Service class must end in “Controller”; and
2. The prefix to “Controller” defines the Web Service URL, and therefore the Controller MUST be named SchoolInfosController (not SchoolInfoController).

## Create a new Environment for the SchoolInfo Provider

Open the Project properties of this SchoolInfo Provider and copy the Project Url from the Web section.

Edit the *Data files\AU\EnvironmentResponse.xml* file of the Sif.Framework.Demo.Setup Project and replace the *requestsConnector* URL with the URL of the new SchoolInfo Provider (keeping /api). Re-run the *Scripts\BAT\Demo execution\DemoAuSetup.bat* script.

## Start the SchoolInfo Provider

To test the newly created SchoolInfo Provider, repeat Exercise 1 replacing the demo Provider with this Provider and ignoring the running of a Consumer.

## Test the SchoolInfo Provider

### Postman (Chrome Plugin)

To test the SchoolInfo Provider without an actual Consumer, a REST Client is required. For these exercises, the Postman plugin for Chrome will be used (<https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm?hl=en>). The following steps need to be performed:

* Create an Environment
* Retrieve all Schools

#### Create an Environment

Use Basic Auth with a Username of “Sif3DemoApp” and a Password of “SecretDem0”. These authentication settings have been pre-defined in the demo database using the Sif.Framework.Demo.Setup project (refer to the *Data files\AU* XML files).

Use a URL of <http://localhost:62921/api/environments/environment>.

Use the POST HTTP operation.

Provide the following payload with the HTTP request (as pre-defined in the Sif.Framework.Demo.Setup project).

<?xml version="1.0"?>

<environment xmlns="http://www.sifassociation.org/infrastructure/3.0.1">

<solutionId>Sif3DemoSolution</solutionId>

<authenticationMethod>Basic</authenticationMethod>

<instanceId></instanceId>

<userToken></userToken>

<consumerName>Sif3DemoStudentPersonalConsumer</consumerName>

<applicationInfo>

<applicationKey>Sif3DemoApp</applicationKey>

<supportedInfrastructureVersion>3.0.1</supportedInfrastructureVersion>

<dataModelNamespace><http://www.sifassociation.org/au/datamodel/1.3>

</dataModelNamespace>

<transport>REST</transport>

</applicationInfo>

</environment>

Specify the following headers:

* Authorization - Basic U2lmM0RlbW9BcHA6U2VjcmV0RGVtMA==
* Content-Type – application/xml
* Accept – application/xml

The Authorization header specified above is generated from Basic Auth with the provided Username and Password from above.

Once ready, send the POST request and save a copy of the response payload (which should match the *Data files\AU\EnvironmentResponse.xml* file).

#### Retrieve all Schools

From the response payload of the Environment POST operation (above), store the <sessionToken> value and the requestsConnector URL.

Using Basic Auth, set the Username to this <sessionToken> value and a Password of “SecretDem0”.

For the URL, use that returned as the requestsConnector URL, e.g. [http://localhost:<port>/api/SchoolInfos](http://localhost:%3cport%3e/api/SchoolInfos).

Use the GET HTTP operation.

Specify the following headers:

* Authorization – as generated from Basic Auth with the provided Username and Password from above
* Content-Type – application/xml
* Accept – application/xml

Once ready, send the GET request. The response payload should contain a list of schools.

### Implement a SchoolInfo Consumer (optional)

Implement a SchoolInfo Consumer as per the StudentPersonal Consumer created in Exercise 2 and run against the newly created SchoolInfo Provider.

## Advanced Exercise

If you have finished the above exercise successfully and have more time, implement remaining methods in the SchoolInfo Provider class.

# Important notes

## Issues with the Environment Provider

There is an existing issue with the Environment Provider whereby if an error occurs with data consumption or provision, an entry in the demo database is not being properly removed. As a workaround, should an exception occur during these exercises, it is highly recommended that the demo database be re-created before starting the Provider or running the Consumer. This can be performed by executing the *Scripts\BAT\Demo execution\DemoAuSetup.bat* script.