(.NET)

Version

**Author:** , SIF Solutions Architect

**Revision:** 2.0

**Published:** May 2022

Copyright © 2022,

Table of Contents

1. Introduction 3

2. Pre-requisite libraries 3

2.1. SIF 3.0 Framework libraries 3

2.2. System.Runtime.Serialization 4

2.3. AutoMapper 4

2.4. log4Net, slf4Net, slf4Net.log4Net 4

2.5. NHibernate 4

2.6. SQLite 4

3. Project configuration 4

3.1. SIF 3.0 Framework configuration 4

3.2. Web API specific configuration 5

3.2.1. Global.asax.cs 5

3.2.2. WebApiConfig.cs 5

3.3. log4Net and slf4Net configuration 5

3.4. NHibernate configuration 5

4. Define the SIF Data Model Object 5

5. Implement the Service interface 6

6. Create the SIF Provider 6

7. Appendix A – Enabling SIF Events 7

7.1. Introduction 7

7.2. IEventIterator interface 7

7.3. IEventService 7

7.4. BroadcastEvents Web API route 7

7.5. SIF 3.0 Framework configuration 7

7.6. Scheduling SIF Events 8

# Introduction

This document outlines the steps necessary to implement a SIF Provider using the .NET version of the SIF Framework.

Before attempting the steps in this document, it is assumed that the following document has already been read:

* SIF Framework Setup Guide

# Prerequisites

In Visual Studio, create a new ASP.NET Core Web API project (.NET 6.0). *Authentication type* can be set to None. *Configure for HTTPS* and *Use controllers* can be ticked (leave every other option unticked).

Once done, using “Manage NuGet Packages…” add the latest version of the following packages to the project:

* Sif.Framework.AspNetCore
* Sif.Framework.EntityFrameworkCore
* Sif.Specification.DataModel.Au
* Tardigrade.Framework.EntityFrameworkCore
* An appropriate SQL database driver

In a Direct environment, the SIF Provider needs to connect with the same database as used by the Environment Provider. In this scenario, the SIF Provider and Environment Provider are treated as a single application.

In a Brokered environment, the SIF Consumer requires a database to store a session token used for managing its connection to an Environment Provider. The definition of the required *Sessions* database table can be found in one of the SQL scripts files under the *Scripts\SQL\Entity Framework Core\Sessions table* folder.

# Project configuration

## SIF Framework configuration

In the project, create an appsettings.json configuration file and ensure that the following properties are correctly set:

* Build Action – Content
* Copy to Output Directory – Copy if newer

#### Brokered environment

Add the following entries to appsettings.json for registering your Provider application to the Environment Provider:

* provider.environment.sharedSecret
* provider.environment.url
* provider.environment.template.applicationKey
* provider.environment.template.authenticationMethod
* provider.environment.template.dataModelNamespace
* provider.environment.template.instanceId (if known)
* provider.environment.template.solutionId (if known)
* provider.environment.template.userToken (if known)

The values associated with these entries should be provided by the SIF Administrator of the Environment Provider your application is connecting with.

#### Direct environment

Add the following entries to appsettings.json:

* provider.environment.template.dataModelNamespace

The values associated with these entries should be provided by the SIF Administrator of the Environment Provider your application is connecting with.

## Web API specific configuration

### Global.asax.cs

Replace the Global.asax.cs file of the project with the version from the Templates\Providers directory of the SIF 3.0 Framework.

In the “\*\*\* TO DO \*\*\*” section of the file, uncomment the lines of code and modify, replacing the SIF Data Model Object used with an appropriate one. Refer to the following section regarding a SIF Data Model Object for more information.

Without these lines, a StudentPersonal SIF Provider (for instance) would return a list of records with a root element of <ArrayOfStudentPersonal> rather than the required <StudentPersonals>.

### WebApiConfig.cs

Replace the WebApiConfig.cs file of the project with the version from the Templates\Providers directory of the SIF 3.0 Framework.

## slf4Net and log4Net configuration

Installation of the slf4Net and log4Net NuGet packages would have updated the App.config file with appropriate configuration settings. If an slf4net element has not been added, refer to the copy in a Consumer demo project.

Copy the log4net.config configuration file from the Templates\Consumers directory of the SIF 3.0 Framework into this project. Ensure the “Copy to Output Directory” file property is set to “Copy always”.

## NHibernate configuration

Copy the SifFramework.cfg.xml NHibernate configuration file from the Templates\Providers directory of the SIF 3.0 Framework into this project. Ensure the “Copy to Output Directory” file property is set to “Copy always”.

**IMPORTANT:** Within the configuration file, the “connection.connection\_string” property needs to reference the same database as that used by the EnvironmentProvider included in the SIF 3.0 Framework. When using the default SQLite database, the connection string defines a relative path from your SIF Provider.

# Define the SIF Data Model Object

For the implementation of a SIF Provider, a SIF Data Model Object class needs to be specified. A requirement for the use of this Framework is that the class must implement the IDataModel interface. For an example implementation (of a student), refer to one of the Provider demo projects.

It is important that the model object used will serialise (XML) to meet the SIF Data Model Specification for a locale. To assist with this requirement, the Sif3Specification Solution can be used as a reference for any data model objects used. Of particular importance are the namespaces used for types.

# Implement the Service interface

Create a Service class that implements the IBasicProviderService interface with the previously defined SIF Data Model Object class as the generic type. Implement the methods of the interface that are required for the SIF Provider. For an example Service class implementation (for a student), refer to one of the Provider demo projects.

# Create the SIF Provider

To implement the SIF Provider, create a new class that extends the BasicProvider class with the previously defined SIF Data Model Object class as the generic type. In the default constructor, call upon the “base” constructor of the BasicProvider class that takes the Service class created. For an example SIF Provider implementation (for a student), refer to one of the Provider demo projects.

As the Web API specification relies heavily on coding convention, some important points to consider when implementing a Provider are the following:

1. The prefix (name) to the “Provider” class defines both the SIF Data Model Object used and the Web Service URL, and MUST therefore be named appropriately, i.e. SchoolInfo**s**Provider not SchoolInfoProvider. The SIF Framework has been configured to treat classes ending in “Provider” that extend the BasicProvider as API Controllers.
2. As the creation of a single object in SIF does not follow the normal RESTful conventions, specific routing for this situation needs to be declared in the newly created SIF Provider. For instance, for the creation of a single StudentPersonal object, the POST action would use a route of “[Route("~/api/StudentPersonals/StudentPersonal")]”. This requirement is necessary to cater for multiple object operations.
3. If the Provider is going to broadcast events, the instructions specified in Appendix A need to be applied. If the Provider is not going to broadcast events, the [NonAction] attribute needs to be applied to the BroadcastEvents(string, string) method. This also means that this method needs to be overridden in the Provider implementation. For instance:

[NonAction]

public override IHttpActionResult BroadcastEvents(string zoneId = null, string contextId = null)

{

return base.BroadcastEvents(zoneId, contextId);

}

# Appendix A – Enabling SIF Events

## Introduction

To enable the broadcast of SIF Events for a SIF Provider, the following changes need to be implemented:

1. Create a SIF Events iterator class that implements the IEventIterator<TMultiple> interface.
2. In addition to implementing the IBasicProviderService<T> interface, the Service class needs to implement the IEventService<TMultiple> interface.
3. Define a specific route in the SIF Provider that exposes the BroadcastEvents REST endpoint.
4. Update the SifFramework.config file to specify the use of a BROKERED environment.

## IEventIterator interface

The class implementation of the IEventIterator interface forms the core logic called upon by the SIF Framework for the broadcast of SIF Events. It is the responsibility of the developer to return appropriate change records through the GetNext() method.

For an example, refer to the StudentPersonalIterator class of the Sif.Framework.Demo.Au.Provider project.

## IEventService

The IEventService interface provides the hook necessary for the SIF Framework to retrieve change records.

For an example, refer to the StudentPersonalService class of the Sif.Framework.Demo.Au.Provider project.

## BroadcastEvents Web API route

To configure an RPC-style REST endpoint for the broadcasting of SIF Events, a route attribute is required for the SIF Provider’s BroadcastEvents action (as I was unable to do so using a conventional route). For instance, for the broadcasting of StudentPersonal SIF Events, the BroadcastEvents action would use a route of “[Route("~/api/StudentPersonals/BroadcastEvents")]”.

For an example, refer to the StudentPersonalProvider class of the Sif.Framework.Demo.Au.Provider project.

## SIF 3.0 Framework configuration

In the SifFramework.config file, set the “provider.environmentType“ property to “BROKERED”. The SIF Framework does not support SIF Events in a “DIRECT” environment. If the BroadcastEvents action is called when the SIF Provider is running in a “DIRECT” environment, and error will be returned.

## Scheduling SIF Events

As the implementation of SIF Events in the SIF Framework is done by using a REST endpoint, it is the responsibility of the developer to schedule calls to execute the broadcast of change records. The mechanism used for scheduling SIF Events is therefore outside the scope of this document.

An example mechanism that could be used would be Azure WebJobs.