SIF Framework (.NET)

Version 6.0.0

Implementing a SIF Provider

**Author:** Rafidzal Rafiq, SIF Solutions Architect

**Revision:** 2.0

**Published:** May 2022

Copyright © 2022, Systemic Pty Ltd

Table of Contents

1. Introduction 3

2. Prerequisites 3

3. Project configuration 3

3.1. Brokered environment 3

3.2. Direct environment 4

4. Define the SIF Data Model object 4

5. Implement the service interface 4

6. Create the SIF Provider 5

7. ASP.NET Core Web API specific configuration 5

7.1. Configure XML serialization 5

7.2. Enabling method override 6

8. Appendix A – Enabling SIF Events 7

8.1. Introduction 7

8.2. IEventIterator interface 7

8.3. IEventService 7

8.4. BroadcastEvents Web API route 7

8.5. SIF Framework configuration 7

8.6. 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
* 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 Provider 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

Configuration of a SIF Provider differs depending on whether it is using a Direct or Brokered environment.

The DefaultConnection database connection string property is not SIF Framework specific but will be used for the instantiation of a service that accesses the session token from the SIF Provider database.

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

## Brokered environment

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

* ConnectionStrings:DefaultConnection
* provider.environment.sharedSecret
* provider.environmentType
* 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)

## Direct environment

Add the following entries to appsettings.json:

* ConnectionStrings:DefaultConnection
* provider.environment.template.dataModelNamespace

# Define the SIF Data Model object

For the implementation of a SIF Provider, a data model class needs to be created that represents a known SIF Data Model object. It is suggested to extend one of the SIF Data Model objects that are available from the Sif3Specification Solution. This Solution consists of projects that provide reference data model implementations that meet the SIF Specification. Of particular importance are the namespaces used for these classes.

A requirement of the SIF Framework is that the created data model class must implement the IDataModel interface. It is also important that the data model class used will serialise (XML) to meet the SIF Data Model Specification for a locale. For an example implementation of the data model class, refer to one of the SIF Provider demonstration projects.

# 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, refer to one of the SIF Provider demonstration 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. Implement the constructor to simply call upon the “base” constructor of the BasicProvider class that takes the service class created. For an example SIF Provider implementation, refer to one of the SIF Provider demonstration projects.

As the Web API specification relies heavily on coding convention, some important points to consider when implementing a SIF 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., StudentPersonal**s**Controller not StudentPersonalController. If the plural form (StudentPersonal**s**) is not used, the SIF Consumer will not be able to call the SIF Provider REST endpoints.
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 SIF Provider is going to broadcast events, the instructions specified in Appendix A need to be applied. If the SIF 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 SIF Provider implementation. For instance:

[NonAction]

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

{

return base.BroadcastEvents(zoneId, contextId);

}

# ASP.NET Core Web API specific configuration

## Configure XML serialization

By default, ASP.NET Core serializes message bodies in JSON. To support XML, XML serializers need to be explicitly added as a service.

In addition, XML serialization of collections results in a root element of name <ArrayOf…>. This behaviour does not comply with the SIF Specification in relation to collections of SIF Data Model objects. For a collection of SIF Data Model objects, the root element name is the pluralized form of the SIF Data Model itself, e.g., the root element for a collection of student objects is <StudentPersonals>.

To ensure compliance with the SIF Specification, the SIF Provider needs to be configured to serialize message bodies appropriately using input and output formatters defined in the SIF Framework. These formatters are applied when adding services for Controllers in the Program.cs class.

The ArrayOfInputFormatter and ArrayOfOutputFormatter classes take a SIF Data Model object as the generic type and is only applicable for message bodies of that SIF Data Model. The SifInputFormatter is used specifically for SIF Infrastructure Model objects (specifically deleteRequestType as required for deletion of multiple records).

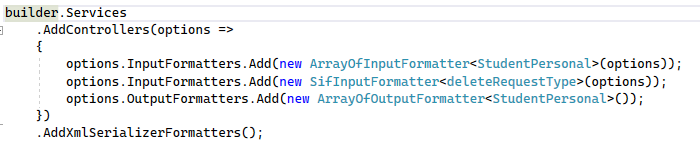


Figure 1: Configuration of input and output formatters

## Enabling method override

HTTP method override is used by the SIF Framework for managing REST calls for multiple creation (POST) and deletion (DELETE) operations. The SIF Framework includes a middleware class (MethodOverrideMiddleware) that needs to be called in the Program.cs class of the SIF Provider.

The MethodOverrideMiddleware class needs to be injected as a service and then call using the UseMethodOverrideMiddleware extension method (in addition to the .NET provided UseRouting extension method).

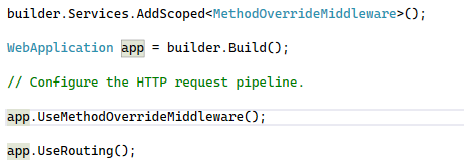


Figure 2: Method override configuration

# 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 appsettings.json configuration 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.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.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. 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.Provider project.

## SIF Framework configuration

In the appsettings.json configuration 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.