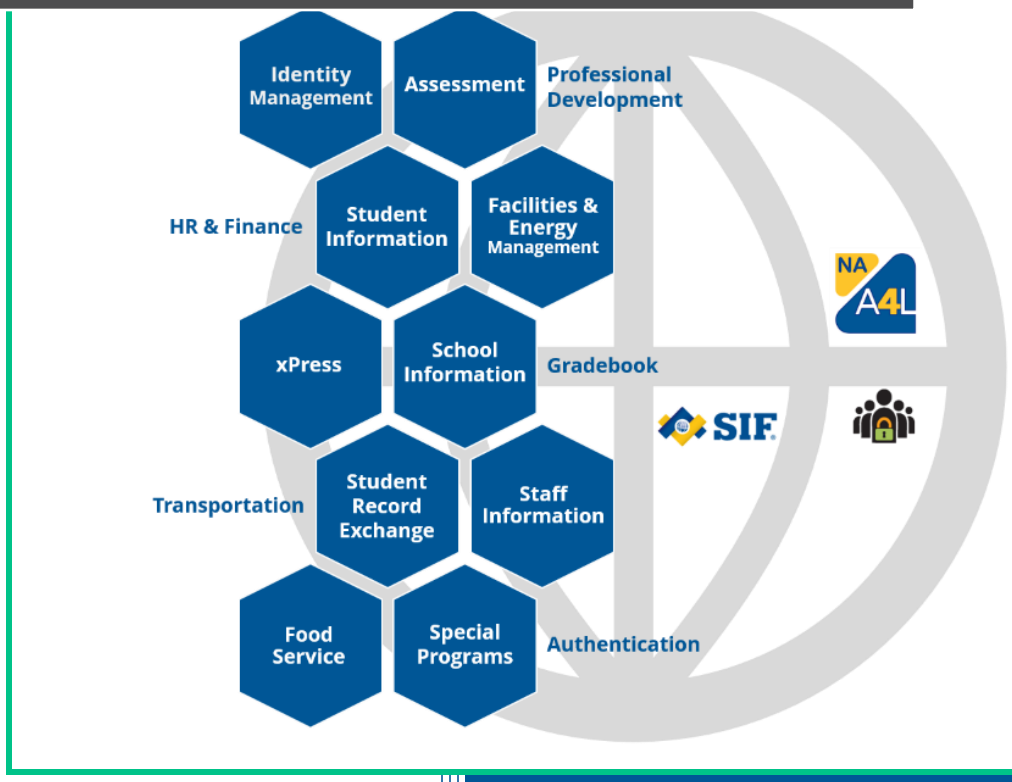
Access 4 Learning Community

SIF Specification

Unity Adoption Guidebook



https://www.A4L.org

Feb 2022 v0.9

Contents

What is Unity? ***4***

Unity's Ecosystems ***7***

Introduction 7
Conventions 7
Direct Solutions 9
Brokered Solutions 10
Data Hub Solutions 11
Hybrid Solutions 12
Conclusions 12

Technical Handbook ***13***

Why is a roster important? 14
Component 1: Authentication 15
Component 2: Infrastructure 17
Component 3: Data 25
User Stories 26
Going Beyond Roster 30
Adding Enterprise Features 30
As Roster Matures 30

Where Privacy Comes In ***31***

The Communities Toolbox 31
The Place for Effective Privacy 32
Privacy Obligation Document (POD) Components 32
Privacy for the Service Provider 33
Privacy for a Data Processor 33

SIF Data Model Specification: Comparison ***35***

Before You Write Code ***38***

Need something fast? 38
Prefer support from SIF experts? 38
Want to share development and maintenance costs? 39
Concerned about the future? 39
Still want to build? 39

Java Open Framework ***41***

Introduction 41

Downloading the Framework 41
Opening the Framework Workspace 43
Building the Framework 44
Demo Project 49
Including in Your Workspace 62
Further Reading 62

.NET Open Framework 63
Introduction 63
Downloading the Framework 63
Opening the Framework Solution 64
Building the Framework 65
Specification Project 67
Demo Project 67
Further Reading 71

Next Steps 72

Thank you to the A4L Community membership, specifically the following, for their help in generating this Guidebook:

Group Leader:
- *Eric Adams, Kimono*

Authors:
- *Rick Borage, Tyler Technologies*
- *Kathy Walter, Nsoma*
- *Larry Fruth, A4L*
- *John W. Lovell, A4L*

Editors:
- *Steve Setzer, Kimono*
- *Jeff Simons, WSIPC*
- *Elycia Hansen, WSIPC*
- *Mike Reynolds, CedarLabs*
- *Penny Murray, A4L*

Framework Experts:
- *Joerg Huber, Systemic*
- *Rafidzal Rafiq, Systemic*

What is Unity?

The A4L Community has taken the best of the best from 20 years of integration (i.e., interoperability specifications), addressing school and state agency needs in a new Specification – 'Unity'[1].

This volunteer developed blueprint has been created using open, non-proprietary, and transparent processes. It contains the most comprehensive K12 data model and modern transport technologies to securely move data to the right person at the right time under local data privacy policies.

The Unity interoperability blueprint:

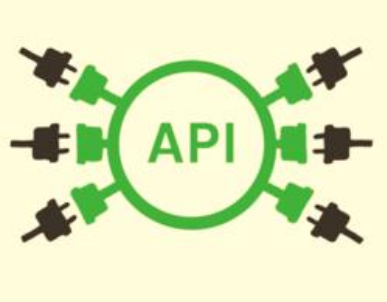
- …supports greater privacy and security controls including the work of the Student Data Privacy Consortium (SDPC)[2].
- …is designed to support API development and standardization.
- …is an implementation blueprint that works with other technical standards, to NOT replicate them, but let you take advantage of 'best of breed' standards (i.e., CEDS, PESC, LTI, Caliper, Ed-Fi).
- …allows you to expect that what you build today will be used tomorrow.
- …allows for easy maturation to a more powerful infrastructure while keeping your data layer investments intact.

Unity supports greater privacy and security controls including the work of the Student Data Privacy Consortium (SDPC)

Introducing POD's (Privacy Obligation Document), Unity enables a deeper dive into what data is made available for which application within your implementation. PODs are a machine-readable set of rules enabling meta-data for privacy controls. Allowing School Districts to control restricted data to certain applications, you can be confident that your learner's data is secure, whilst streamlining your implementation and delivery of contracted expectations.

[1] *The 'Unity' Specification can be found here: http://data.a4l.org/sif-specifications/sif-specifications-north-america/*

[2] *Student Data Privacy Consortium (SDPC): https://privacy.A4L.org*



This Photo by Unknown Author is licensed under CC BY-SA-NC

Unity is designed to support API development and standardization

With updated API objects, Unity allows for the quick and easy implementation of rostering and IEP. The Unity Data Model builds on the existing SIF 2.x Data Model, enabling incremental moves (one application at a time) to update your existing integration.

Example Use Case: ***RICOne.org*** - RIC One is a family of services serving most of NY State, which enable districts to more efficiently and effectively utilize student data systems and to improve school operations and classroom instruction, while ensuring the privacy of sensitive student data. RIC One is a web service and data hub built on the CEDS data model, supporting interoperability for products that have adopted SIF, Ed-Fi and other data frameworks.

Example Use Case: Grade Passback - Unity's use of the SIF 3 (global) Infrastructure opens up a better path for the many districts moving grades from their Learning Management System (LMS), or Assessment software, back to their Student Information System (SIS). At its core, the main change is from a "best effort" approach that the SIS may or may not honor, to a system that allows for feedback and cooperation between services. Additionally, because the Unity Specification picks up where SIF 2 left off, modernizing these systems means significant improvements with a light lift.

The Community is developing implementation blueprints that work with other technical standards…

Unity offers the tightest and broadest technical alignment to the Common Education Data Standard (CEDS)[3] and the US Department of Education's work. With use case driven development, Unity addresses state reporting and interoperates with Generate, enabling mapping between standards easier. By working WITH other standards - like the recent collaboration with PESC developing the JSON Task Force - the A4L

[3] *Common Education Data Standards (CEDS): https://nces.ed.gov/programs/ceds/*

Community is focused on the broader marketplace needs. Unity has been designed so that it will effortlessly work with other standards, so you can choose the right one for you!

Easy maturation to a more powerful infrastructure while keeping your data layer investments intact

The Unity Specification allows you to leverage the SIF 2x Specifications. The Community has been diligent in their commitment to ensuring that marketplace implementers have the ability to add / build on existing SIF implementations incrementally - one application at a time. Ref IDs have been kept as one-to-one mappings, which supports old and new style Specifications.

You can expect that what you build today will be used tomorrow

Unity is a solid starting point, with proven components, to develop a data exchange solution for disparate software applications.

The A4L Community has committed to a clear and strategic direction, with growing commitment from Community members, whether they be vendors, districts, or states, to modernize and utilize the Unity Specification. There is also much excitement for developing Gradebook passback for a release in the near future.

Brief technical explanation: Due to 'collections' or 'groupings' used in the SIF 3 Infrastructure, users are now able to get data that has changed - not 'all' data or another 'event' - this dramatically increases the performance time for data exchanges.

Unity's Ecosystems

Introduction

Before you consider if you are going to build, buy, or partner to get your application into one or more Unity Ecosystems, you are going to want to figure out what role your software will play. In order to gain an appreciation for these options and expectations, this chapter lays out an overview of how integrations using the (global) SIF 3 Infrastructure may look. Based on experiences from our international members, you're going to want to make it all the way to the end before you make any decisions.

Conventions

The diagrams in this chapter use symbols the reader should know about. Along with these symbols, various roles are also explained. Overarching assumptions that form a lens for viewing the diagrams are also enumerated.

Components

Application

Whether a Service Consumer script pulling down data or the most sophisticated Service Provider on the planet, this symbol represents an integrated Application and its adaptor's REST API capabilities.

Connection(s)

While your Service Consumer may need multiple connections, you should expect it to be directed to a *bidirectional* REST API from a single network endpoint, regardless of the topology being discussed.

Broker

The software that (when present) connects agents together for a specific data scope by providing (global) SIF Infrastructure Services through REST APIs.

Operational Data Store

This represents an Operational Data Store (ODS) that readily provides and *accepts* the (global) SIF Infrastructure and Unity data through REST APIs.

Roles

Infrastructure Provider: The Provider of the (global) SIF Infrastructure's services to the integration.

Service Provider: One or more Applications that service Unity data requests.

Service Consumer: Applications that make Unity service requests without servicing them in return.[4]

Assumptions

- Requests for data from Service Provider systems may be made by Service Consumers.
- Request to create, update, and delete data in Service Provider systems may also be made by Service Consumers.
- All Service Consumer may participate in any one of these topologies *without change*, given that other systems are providing the (global) SIF Infrastructure and Unity data services needed.
- Integrated Components may be combined or extended over time to support multiple needs at once.
- Existing Components may take on different complementary or even reduced roles to better fit with the growing capabilities of other Applications and their impacts on the Ecosystem.
- Diagrams represent certain *ideal* topologies and are used to create common points of reference.
- Components in each diagram have been arranged so that data *predominately* flows from left to right.

[4]*The (global) SIF Infrastructure requires that all Applications participating in an integration be registered as Service Consumers, however for the purposes of this chapter we use a narrower definition in order to make a greater distinction between roles.*

Direct Solutions

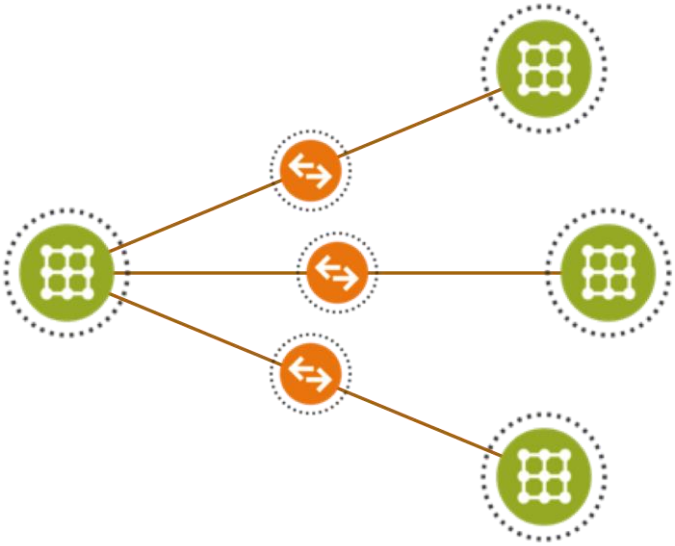
Both common in the world of the technology as the client-server model and largely inaccessible to the North American market of SIF solutions previously, direct solutions grant Service Consumers access to the Service Provider without the need for a middleman. This makes Unity appropriate for the smallest of integrations, empowering handling synchronous data requests in both directions. Direct Data Access means activities such as a teacher taking attendance on a tablet computer could be recorded on the Student Information System (SIS) with immediate feedback and recall.

Pros

- The simplest ecosystem possible
- No additional components necessary
- Enables simple interactions with apps
- Clear growth paths to take

Cons

- Only one Service Provider per ecosystem
- Only as strong as the Service Provider:
  - May burden the Service Provider
  - May limit the Service Consumers
- Growth requires additional components



Roles

Infrastructure Provider: Application on Left
Service Provider: Application on Left
Service Consumers: Applications on Right

In this diagram the Application on the left is the server playing the roles of both Infrastructure and Service Provider, while the client Applications on the right play the role of Service Consumers empowered to read and write certain data directly to the Service Provider. Sticking with our attendance example, imagine using your tablet with a Service Consumer application to load up the roster for your section and adding attendance information with both sets of data living on the Student Information System (SIS) as the Service Provider. A very simple example, however you can take this a long way as long as you only need one Service Provider.

Brokered Solutions

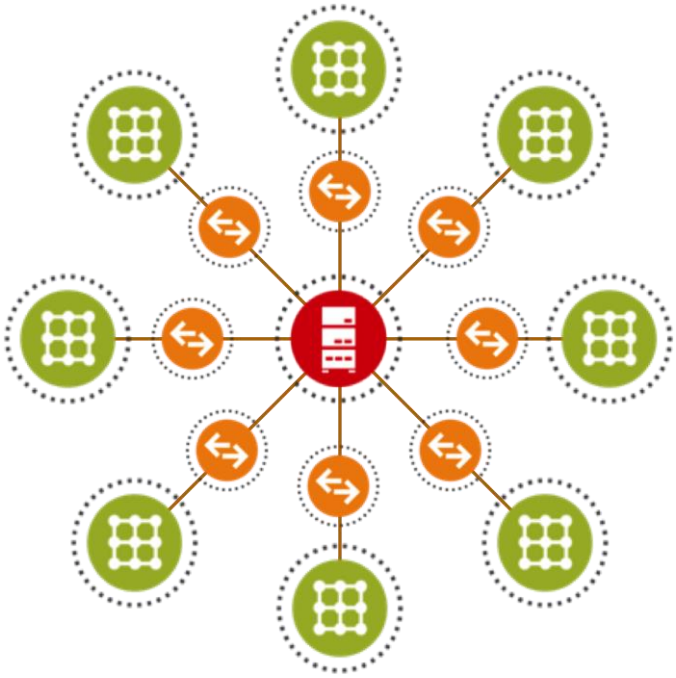
At the other extreme of the data integration world is the middleware model, which is the traditional topology for SIF integrations and is still the most capable. This makes Unity appropriate for large scale integrations as the Broker manages the other components including connections and security. Brokers allow States to flow data up for reporting and back down in order to positively impact administration and education. Schools and districts use them to enable the near real time sharing of data across their systems, ensuring things such as each student being picked up and dropped off at the right place each and every day.

Pros

- The dedicated Broker provides management of the entire integration
- All Applications may consume and/or provide data
- Mix, match, and configure Applications to create the strongest ecosystem

Cons

- Ensuring well matched quality Service Providers for each data domain, takes additional deliberate effort
- In order to maintain robustness, data may need to be exchanged asynchronously



Roles

Infrastructure Provider:	Broker in Middle
Service Provider:	Some Application(s)
Service Consumers:	Some Application(s)

This topology brings home the advantages of utilizing a Broker. One of these advantages being that from any one application's point of view the simplicity of a small integration is maintained. Instead of seven sets of certificates, credentials, services, access control lists, connections, and more, it has one set for the Broker. This encourages more Applications to participate in the integration. So, when the Transportation Department verifies and shares the best addresses for a student and their contacts, it doesn't just help bus drivers keep kids safe, it positively impacts other systems as well. Systems like Food Services, which now have a good address to send a lunch balance report to, which in turn leads to a student getting a good meal on a day they had state testing. Strong systems make everyone stronger and when it comes to large data integrations, the strongest have this topology.

Data Hub Solutions

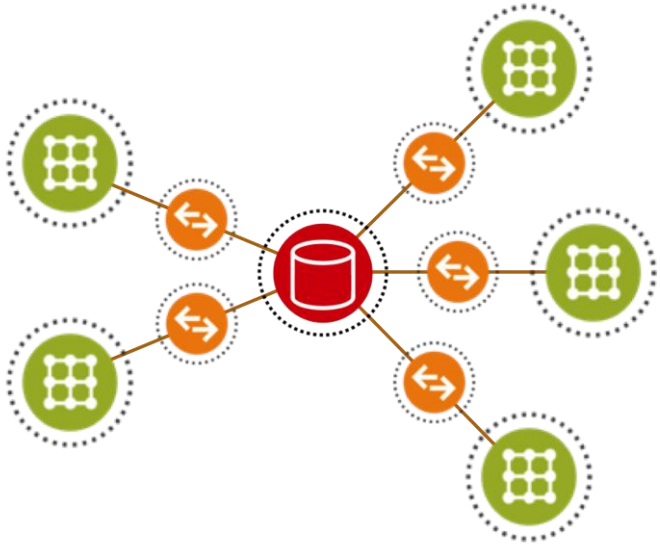
For a mid-sized integration, we have found that the (global) SIF Infrastructure lends itself well to a data pool style. We often refer to an integration of this topology as a Data Hub because it creates a place to put data so that it can be queried using the Data Hub as the source of truth. The ability to control the source data set for reporting is often a desired feature for helping ensure data quality. This makes Unity able to evolve to meet a growing integration's needs, as the ODS carries the load of being both the Infrastructure and Service Provider.

Pros

- No Application need be a Service Provider
- One source of truth for all queries
- Queries that span multiple Applications' data can be answered

Cons

- Query results are only as up to date as the data sent to the ODS by its Service Consumers



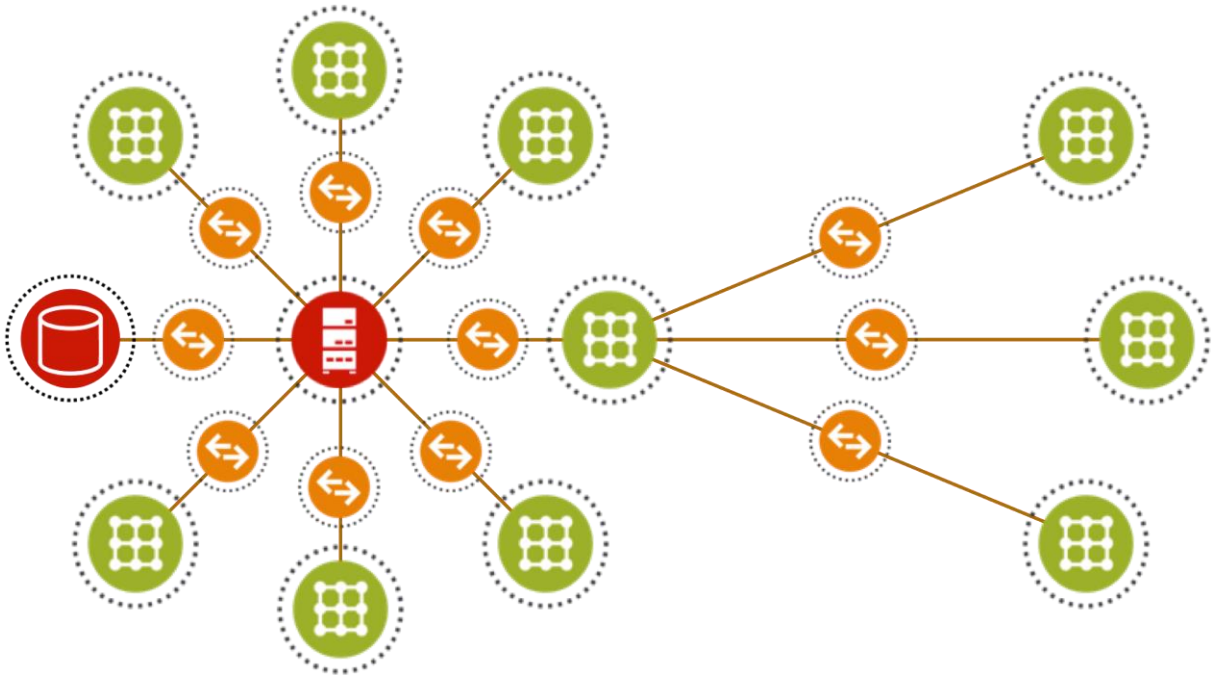
Roles

Infrastructure Provider:	ODS in Middle
Service Provider:	ODS in Middle
Service Consumers:	All Applications

The picture shown depicts the Service Consumers on the left supplying data through requests to the ODS in the middle which can then handle queries from the Service Consumers on the right. While these are artificial limits, it helps convey how an ODS can help create a 'Unity Ecosystem' by taking on the burdens of being the only Service Provider. When it comes to growth, notice that the right half of this example is exactly the same as our direct solution. At the same time, you can see that if the two Service Consumers on the left-hand side became Service Providers and the ODS was swapped out with a Broker, we would have a Brokered Solution, albeit not as large as the previous.

Hybrid Solutions

Having a common REST API and supporting services opens the door to many possibilities. When putting together solutions, one shouldn't be afraid to mix and match approaches in order to construct the best solutions possible. Here is what it might look like to combine all of the functionality discussed above.



Here we have an ODS that can store whatever data isn't provided elsewhere, linked by a Broker managing the integration, and a Service Provider hosting additional Service Consumers. While an integration like this one is best rolled out in phases, growing to this level of sophistication is a very real possibility. In fact, many projects, both here and abroad, have used brokers as Infrastructure Providers in front of one or more Service Providers to create a more modular Data Hub.

Conclusions

When looking to enter a 'Unity Ecosystem' there is one clear place to start and that is creating a solid Service Consumer. Not only is this the least amount of work, the Ecosystems you encounter may very well already have other Service Providers so you shouldn't assume this will be one of the roles your software will fulfill. In such a 'Unity Ecosystem', your Service Consumer may contribute to the data by making create, update, and delete requests. If it becomes necessary to beef up your Adaptor in order to fulfill the role of a Service Provider, you will be in a much better position to understand what that will take and what features are important for you to support.

Technical Handbook

This chapter is updated and adapted from the 'SIF Standards - Technical Handbook' which takes an OAuth approach to authentication.

If you are new to the SIF standard, this is a very good place to start. This document lays out all the pieces you need to get started and also talks through some practical examples where meaningful data exchanges happen without much effort using the SIF 3 Infrastructure[5] and featuring the North American xPress Roster objects[6].

A SIF implementation requires three components:

1. Authentication – proof that an interface is allowed to interface
2. Infrastructure – setup to exchange information with approved interface
3. Data – actual information exchanged between approved interfaces

We will walk through each of these areas in detail to foster understanding, implementation, integration and a technical foundation for addressing future education needs.

We will also peer into the future at enhancements that can make this baseline setup even more useful. The Access 4 Learning (A4L) Community's unique make-up of educators, vendors, and information technology professionals are building APIs, enabling access to the data and impacting learning through real world products, services, integrations, and solutions.

The purpose of this chapter

This chapter is designed to demonstrate the basic principles of a Service Consumer, e.g., data consumption. These include simple authentication, REST headers and payloads, and even data examples. It is not intended to be a complete distillation of all available authentication systems, integration with brokers, or a full-service cookbook, for a complete SIF implementation.

---

[5] *SIF Infrastructure Implementation Specification: http://data.a4l.org/sif-infrastructure/*

[6] *xPress objects can be found in the Unity Specification (also known as the SIF Data Model Implementation Specification (North America) 4.2): http://specification.sifassociation.org/Implementation/NA/4.2/*

Why is a roster important?

At the simplest level, a roster is a list of students/learners in a program, class/section, or organization: a school, district, or regional entity. In an educational software application, a roster is used to connect a student or learner with the related educator or teacher.

Rosters are important because any kind of instructional application or administrative application that supports the teacher in the classroom requires a link between the students in a class/section and the teacher. That link is sometimes obvious and implicit in the relationship of the teacher to the section, but at other times with software programs, or project-based learning, more explicit relationships need to be created between the students and the teacher. This roster of students is required by all software applications that support students and gives the teacher the ability to manage, coach, and teach the students.

This makes the roster in education one of the fundamental building blocks of any instructional, curricular, assessment, or administrative application.

Further reading on xPress Roster can be found here: https://xpressapi.org/

However, if you are looking to leverage more established objects (such as StudentPersonal) to access the same information, that can be accomplished through the same mechanisms demonstrated here.

Component 1: Authentication

To access our xPress Roster objects we will use SIF_HMACSHA256 as it is the default authentication method of the (global) SIF Infrastructure. SIF_HMACSHA256 has several important features that afforded it this honor:

- Starts simply with two pieces of information.
- Never sends the secret over the (global) SIF Infrastructure.
- Secured with a one directional hash algorithm.
- Confirmation includes identifying the Service Consumer.
- Salted with the current date and time in UTC ISO 8601 format (contained in the timestamp element) so it changes on every request to avoid replay attacks.

Additionally, we will be using pre-provisioned Environments as this is a great way to simplify getting started with SIF.

Note: While the (global) SIF Infrastructure makes provision to use any SSO Authentication method, interoperability can only occur if the authentication systems on the client and server sides of the connection match. In order to maximize compatibility, the infrastructure documentation includes guidance for the following:

- Basic Authentication
- SIF_HMACSHA256
- OAuth 2.0
- Client Certificates

Getting an Authentication Token

In order to generate a SIF_HMACSHA256 Authentication Token the following information is required.

- Timestamp: Date/Time in in ISO-8601 format
- Application Key/Session Token: Unique ID
- Shared Secret: Used to hash the above information.

So that the authentication method employed, and identity of the sender is not completely lost in the shuffle, there are four steps to creating a usable Authentication Token.

Operation	Example Value
Concatenate the applicationKey and date/time and separate them by a colon	RamseyPortal:2013-06-22T23:52-07
Calculate the HMAC SHA 256 value using the unsent Client Application shared secret, and then Base64 encode	6TVgYwbAhmQc2zALda8ZupfrzfqZ+XD7f2bMA0AzWRo=
Combine the *applicationKey* with this string and separate them by a colon	RamseyPortal:6TVgYwbAhmQc2zALda8ZupfrzfqZ+XD7f2bMA0AzWRo=)
Base64 encode the result and prefix with the authentication method and a space	SIF_HMACSHA256 bmV3OjZUVmdZd2JBaG1RYzJ6QUxkYThadXBmcnpmcVorWEQ3ZjJiTUEwQXpXUm89Cg==

Example JavaScript

The following code runs in Node.js and is used to generate all following examples, however the expectation is you will replace the “PUT_SESSION_TOKEN_HERE” and “PUT_SHARED_SECRET_HERE” with corresponding production values.

```
var CryptoJS = require("crypto-js");

var timeStamp = (new Date()).toISOString();
var sessionToken = "PUT_SESSION_TOKEN_HERE";
var sharedSecret = "PUT_SHARED_SECRET_HERE";
var valToHash = sessionToken + ":" + timeStamp;
var hash = CryptoJS.HmacSHA256(valToHash, sharedSecret).toString(CryptoJS.enc.Base64);
var authToken = "SIF_HMACSHA256 "
  + CryptoJS.enc.Base64.stringify(CryptoJS.enc.Utf8.parse((sessionToken) + ":"
  + hash));

console.log("Timestamp: " + timeStamp);
console.log("Authentication Token: " + authToken);
```

Using the Token

When you have computed your Authentication Token you **may** use it with the (global) SIF Infrastructure in the HTTP Authorization header. When servicing a request it is up to the software handling the request to verify the token’s validity and access level.

Example Header

```
Authorization : SIF_HMACSHA256
UFVUX1NFU1NJT05fVE9LRU5fSEVSRTpYdk0zbGpxV3VJMXV2UnhZVkttUkFsV3hlTFVhM2orbEdNUUwvW
k52NzdFPQ==
```

Component 2: Infrastructure

To get started, the infrastructure required is very simple - it uses a REST interface. Our examples will be even simpler, in that they only utilize READ operations.

If you are not familiar with REST, check out this tutorial 'What is REST?'.

Your returned data comes back, not as files, but as xPress Roster objects. Usually, data is stored in relational databases on both ends of the wire; although it is up to the implementer to choose their own approach to storing and/or viewing data. However, a good XPress Roster product serializes or parses the passed objects one page at a time.

More API details can be found in the examples below and in the latest SIF Infrastructure Implementation Specification.

Security & Other Supporting Standards

Just like OAuth 2.0, the SIF 3 Infrastructure relies heavily on existing Internet grade security. This includes the use of TLS (HTTPS) to prevent the reading of intercepted messages. Other standards are also leveraged to create a robust ecosystem. While this document further profiles the SIF 3 Product Standard, understanding the information and requirements within it is required when certifying a SIF xPress Roster solution.

Adaptors based on the SIF 3 Infrastructure fulfill different roles. For this document the **Service Provider** holds the data and returns it when requested while one or more **Service Consumers** makes requests of the Service Provider. While a great place to start, these roles may expand in powerful ways as more Infrastructure features are utilized.

One of the guiding principles when defining the SIF 3 Infrastructure is that it must be re-usable without change to support the Data Model of any SIF locale: in other words, be payload independent. This means not only do people use this infrastructure to move data designed for Australia, New Zealand, North America, and the United Kingdom but other formats such as comma separated values and encrypted opaque payloads. So, as you read the xPress Roster examples below, keep an open mind.

Retrieving Data

In this section do not worry about the data returned. In fact, you do not even have to consider why we get back the data that we do. Instead, direct your thoughts to how to build and/or service requests.

Below are samples of requests and responses, including additional field definitions. For a complete list of all field definitions, refer to the latest version of the SIF Infrastructure Implementation Specification.

Example GET

URL:

```
http://163.153.114.103/api/requests/xStudents.json?navigationPage=1&navigationPageSize=1
```

Headers:

```
Accept: */*
Accept-Encoding: gzip
Authorization: SIF_HMACSHA256
UFVUX1NFU1NJT05fVE9LRU5fSEVSRTpkSVA3c296NHFFc3lWZ2J1M0JlcU9wRDlobU1XRzdaVXYvV0xXOGsyeTJvPQ==
Timestamp: 2020-06-28T01:47:44.721Z
```

Parameters

Characteristics

The Request, Response, and Event columns use the standard SIF Characteristics. Tables used throughout this and other documents include one of the following primary (and mutually exclusive) element characteristics:

- **M – Mandatory**. The element must appear in every Create Event and, where not specifically excluded in a conditional Request, in every Response message issued by the Service Provide as well. If a Create Request does not specify one or more Mandatory elements, the request is erroneous.
- **Q – ReQuired**. If the element appears in the original Create Event or is eventually included in an Update Event (i.e. once it is known to the Service Provider), it must be returned in all corresponding queries as if it were Mandatory.
- **D – RecommenDed**. The element is optional, but we encourage systems to provide these fields to establish baseline communications.
- **O – Optional**. The element may or may not appear in any message relating to the object. The Service Provider need not support it.

One or more of the following qualifiers may also appear with the above characteristics:

- **C – Conditional**. The element is treated as the accompanying primary characteristic only if the specified conditions are satisfied. Otherwise the element is omitted from the message. Specifically:
  - **MC** – If conditions are such that the element can legally be included, it must be
  - **OC** – If conditions are such that the element can legally be included, it may be.
- **I – Immutable**. The value of the element cannot be changed once it is supplied.
- **U – Unique**. The value of this element for each object of this type must be unique (ex: ID)
- **N – Non-Queryable**. The element value is often calculated (ex: an aggregate) and cannot be used as a search key in a conditional Query Request.
- **R – Repeatable**. The element may appear more than once.

The Conveyed column using the following abbreviations:

- **H - HTTP Header**
- **Q - URL Query Parameter**
- **M - URL Matrix Parameter**
- **P - URL Path**

When more than one conveyance is utilized or a conditional is indicated, see the explanation for details of its use.

Likely Request Parameters

HTTP Header Field Name	Request	Response	Event	Conveyed	Explanation
Accept	O			HP	Used to indicate when JSON is expected in the response (application/json). If omitted, may also be indicated by including the ".json" extension in the URL's path. Otherwise results will be conveyed using the default, XML.

access_token	MC			Q	The token used to authenticate the sender of the message, authorizing the requested action. Usually the token/hash value of the Authorization header. This query parameter is only required when the Authorization header is not set or another authentication standard is leveraged.
navigationPage	O	MC		HQ	The number of the Page to be returned. If it is outside the range of results (which does not constitute an error) an HTTP Response with a code of 204 (No Content) will be returned. The first page is indicated with the value 1 (i.e. navigationPage=1).
navigationPageSize	MC	MC		HQ	This is included in every Paged Query Request and indicates the number of Objects to be returned in the corresponding Response Page. If the Page Size specified is too large for the Service or Environments Provider to supply, an Error with code 413 (Response too large) will be returned. When contained in the Response, it indicates the actual number of objects on the returned Page.
serviceType	MC	MC	MC	H	One of: UTILITY/OBJECT/FUNCTIONAL/SERVICEPATH/XQUERYTEMPLATE/ SERVICE. If not provided, it will default to OBJECT
Timestamp	MC	M	M	HQ	Date / Time of Event creation (in ISO-8601 format also used as the basis of xs:dateTime) If not needed for authentication, may be omitted in the request. If needed, only for requests, this value may be provided as a URL query parameter instead of a header.

Example Response

Status Line:

```
HTTP/1.1 200 OK
```

Headers:

```
Content-Type: application/json
navigationCount: 1
navigationLastPage: 1
timestamp: 2020-06-28T01:47:47.67Z
Server: Apache-Coyote/1.1
```

```
messageType: RESPONSE
Date: Tue, 27 Oct 2015 22:51:33 GMT
responseAction: QUERY
Content-Length: 1993
navigationPage: 1
navigationPageSize: 1
environmentURI: http://localhost:8080/api/environments/44524a80-b71b-49cc-
8bf2-250000b6712b
relativeServicePath: /xStudents.json
```

Body:


```
{
    "xStudents": {
        "xStudent": {
            "@refId": "D47B7B88-CE17-44FB-B94F-0000E5BA0532",
            "name": {
                "type": "LegalName",
                "familyName": "Pitts",
                "givenName": "Jennifer",
                "middleName": "X"
            },
            "localId": "471777",
            "stateProvinceId": "735668753",
            "address": {
                "addressType": "Mailing",
                "line1": "936 Cedar Drive",
                "city": "MOUNT VERNON",
                "stateProvince": "NY",
                "countryCode": "US",
                "postalCode": "10552"
            },
            "phoneNumber": {
                "phoneNumberType": "Cell",
                "number": "5552585105",
                "primaryIndicator": "false"
            },
            "email": {
                "emailType": "Organizational",
                "emailAddress": "JenniferPitts@JohnsonCityMS.edu"
            }
        }
    }
}
```

Likely Response Parameters

HTTP Header Field Name	Request	Response	Event	Conveyed	Explanation
content-Type	MC	M	M	HP	Tells the receiver how to parse the body of the message. Supported generally, however SIF data models are generally conveyed with types: application/json or application/xml (default). Must be conveyed whenever a body is present. May be omitted in a request. In that case the mime type is either: the mime type indicated on the URL (i.e., json) or XML if not defined on the URL or the HTTP Header.
environmentURI		OC		H	May be returned by the environment provider where the environment is pre-provisioned.
messageType	O	M	M	H	One of: EVENT/REQUEST/RESPONSE/ERROR. If not provided, it will default to REQUEST.
navigationCount		O		H	The total number of objects in the set of results generated by the initial Paged Query that is associated with the returned navigationId.
navigationLastPage		O		H	It is included as an aid for the Service Consumer in detecting when to stop issuing Paged Query Requests.
relativeServicePath		MC		H	Replicates all information contained in the segments of the Request URL following the Request Connector. This could include the Service name, eXtended Query Template name or Service Path defining the payload format, and any accompanying URL matrix parameters (Context and Zone). URL Query parameters are included. The Environment Provider places it into all delayed Responses (and would therefore not be supplied by a Service Provider in a Brokered Architecture) as an aid to stateless Service Consumers. It is optional for immediate Responses.
requestId	MC	MC		H	Only required for delayed Requests. A Service Consumer specified “token” that uniquely identifies every delayed Request issued by the Service Consumer. Used to correlate the delayed (asynchronous) Response with the original Request. It could be as simple as a monotonically increasing integer.
responseAction		M		H	This must exactly match the requestAction value contained in the HTTP header of the Request being responded to. Valid values are: CREATE/UPDATE/DELETE/QUERY/HEAD

serviceType	MC	MC	MC	H	One of: UTILITY/OBJECT/FUNCTIONAL/SERVICEPATH/XQUERYTEMPLATE/SERVICE. If not provided, it will default to OBJECT
Timestamp	MC	M	M	HQ	Date / Time of Event creation (in ISO-8601 format also used as the basis of xs:dateTime) If not needed for authentication, may be omitted in the request. If needed, only for requests this value may be provided as a URL query parameter instead of a header.

Identifying Success

Any of a number of 2XX and 3XX status codes may be returned in HTTP Responses to indicate that the requested action contained in the Service Consumer's (or in the case of publishing an Event, the Service Provider) HTTP Request was received, understood, accepted and processed "successfully".

HTTP Code	Meaning	Example of Use
200	OK	The standard HTTP response code for all successful HTTP requests, with the exceptions noted below
201	Objects Created	One or more objects have been successfully created
202	Accepted	The SIF Request contained in the HTTP request has been accepted for routing, but the processing has not been completed. This is the status code returned in the HTTP response to every delayed SIF Service Consumer Request, as well as every published SIF Event (by a Service Provider).
204	No Content	All change Requests have Responses with contents. This is the response to a Query for which no existing object qualified except where the query uses the object id notation (e.g., .../students/{studentId}). In such a case the HTTP Status 404 must be returned (see also description for HTTP Status 404).
304	Not modified	The specific response when a Query asks for objects which have changed, and none have

Possible Errors

There is also a range of standard HTTP Error Codes (4XX and 5XX) that will be returned in case of Error. Whenever a SIF Error object is returned in response to a Request, a known HTTP Error Code will be returned in the HTTP Status field. This field can have one of the following values:

HTTP Code	Meaning	Example of Use
400	Bad Request	XML error, version problems or error specific to a particular object type such as the omission of a mandatory element or unsupported query or unsupported order clause
401	Unauthorized	Illegal Service Consumer Authorization token accompanying the request
403	Forbidden	Service Consumer Authorization token is legal, but Service Consumer is not authorized to issue the requested operation
404	Not Found	Object ID does not correspond to an existing object. This can occur for Query as well as Update or Delete operations. No Service Provider has been found to match the parameters (Zone, Context, Service name) in the Request.
405	Method not Allowed	Paged Query Request issued to Object URL rather than Object List URL.
409	State Conflict	An attempt has been made to create an existing object.
410	Gone	Used for query functionality where the Changes Since opaque marker or Paged navigationId provided by the consumer is no longer valid.
412	Precondition Failed	An attempt has been made to modify an object when the Requester was not basing the modification on the latest version.
413	Response too large	A non-paged Query returning all objects was too large for the Service Provider (or Broker) to include in a single Response message.
500	Internal Service Error	An unexpected error occurred in servicing the Request.
503	Service Unavailable	Returned only for Service Consumer Requests requiring an immediate Response. This error indicates that the expected Service processing time for the Request is great enough that the Service Consumer must reissue it as a Request requiring a delayed Response.

Example Error

```
HTTP/1.1 401 Unauthorized
responseAction: QUERY
messageType: ERROR
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 156
Date: Fri, 23 Oct 2015 19:40:09 GMT
timestamp: 2020-06-28T01:47:56.31Z
relativeServicePath: /xStudents.json

{
      "error": {
      "@id": "e1e19242-0654-4f5f-bea7-50b4e6cb29b0",
      "code": "401",
      "message": "Not Authorized.",
      "description": "No or invalid Authorization Token provided"
      }
}
```

Component 3: Data

To help lay out details around data, let's look at two different user stories. First, we will consider Jack who needs information about the students in a district, so that they may access his company's subscription learning service. Then we will take a look at Jill who wants her class roster with her on a mobile device, as she wanders around her classroom.

More details about the API can be found in the examples below and also in the SIF Data Model Specification.

Minimums

In order to ensure all xPress Roster Service Providers add value, a baseline set of support has been established. While these tables layout a set of recommendations, the A4L Community fully recognizes that some use cases may require omitting certain data elements. Therefore, software meeting these minimums may be badged differently than those that do *not*.

Further reading: xPress Roster – Data Guidance

This document can be found in the Technical Support section: http://data.a4l.org/technical-support/

User Stories

We've reviewed how to properly access data by asking permission, formatting information requesting, and processing the data available. Let's walk through the two examples to illustrate how these processes come together.

User Scenario 1: Provisioning Educational Systems

Jack sells online learning for "Earth Sci 9H", a popular online earth science curriculum. Planet School District purchased Jack's curriculum only for its students who are taking earth science and the teacher teaching it. The school's online learning platform uses XPress to securely connect the district and its content publishers. "Earth Sci 9H" has been given XPress Roster credentials to the district's data. Jack wants to make sure that only teachers and students actually taking earth science will be set up with "Earth Sci 9H" accounts.

First Jack's application requests all the courses and finds all courses of interest.

URL:

```
http://163.153.114.103/api/requests/xCourses.json
```

Request Headers:

```
Accept: */*
Accept-Encoding: gzip
Authorization: SIF_HMACSHA256
UFVUX1NFU1NJT05fVE9LRU5fSEVSRTpDR3kveDFSU1cyUExHb1R5WWE4a3hBcWJmdzZGTHdaSzZXNEpWS
09oZ05nPQ==
Timestamp: 2020-06-28T01:52:38.280Z
```

Response Headers:

```
responseAction: QUERY
messageType: RESPONSE
Server: Apache-Coyote/1.1
environmentURI: http://localhost:8080/api/environments/823c6dfd-e356-4d3e-b23a-
46b7a9d92ad0
Content-Type: application/json
Content-Length: 14282
Date: Mon, 02 Nov 2015 22:12:41 GMT
```

```
timestamp: 2020-06-28T01:52:41.94Z
relativeServicePath: /xCourses.json
```

Response Snippet:

```
{
    "@refId": "A2258F48-7B8C-4406-A00E-462DCDCD3CE3",
    "schoolRefId": "66667705-6C51-4C30-A22A-77CEA0FBCF53",
    "schoolCourseId": "JMS0115",
    "courseTitle": "Earth Sci 9H",
    "description": "Test Description 1",
    "subject": "Science",
    "applicableEducationLevels": {
        "applicableEducationLevel": "11"
    }
}
```

Next, Jack's application uses the course's unique identifier obtained from the course request to associate the course with all the people involved, including students and teachers.

Then, the application sets up tracking for the students and teacher in his systems utilizing one of the unique identifiers provided by the learning system or the student names, through xRoster objects (see example below). Once that information is obtained, the application can store the unique identifiers, the student names or both.

URL:

```
http://163.153.114.103/api/requests/xCourses/A2258F48-7B8C-4406-A00E-462DCDCD3CE3/xRosters.json
```

Request Headers:

```
Accept: */*
Accept-Encoding: gzip
Authorization: Token: SIF_HMACSHA256
UFVUX1NFU1NJT05fVE9LRU5fSEVSRTorenZYRUk3cjJUOXAxTHowdWhGMFNGZ29BazY0dy9GRXRtTlRBR
DZ2SFBZPQ==
serviceType: SERVICEPATH
Timestamp: 2020-06-28T02:00:33.817Z
```

Many rosters (one for each section of the specified course) are returned and accounts can be built from the results.

Response Headers:

```
responseAction: QUERY
messageType: RESPONSE
```

```
Server: Apache-Coyote/1.1
environmentURI: http://localhost:8080/api/environments/823c6dfd-e356-4d3e-b23a-
46b7a9d92ad0
Content-Type: application/json
Content-Length: 20840
Date: Mon, 02 Nov 2015 22:15:45 GMT
timestamp: 2020-06-28T02:00:36.73Z
serviceType: SERVICEPATH
relativeServicePath: /xCourses/A2258F48-7B8C-4406-A00E-462DCDCD3CE3/xRosters.json
```

Abbreviated Response:


```
{
    "xRosters": {
        "xRoster": [
            {
                "@refId": "2127E79B-CFA9-4CE9-B277-11CA5E0001FC",
                "courseRefId": "A2258F48-7B8C-4406-A00E-462DCDCD3CE3",
                "courseTitle": "Earth Sci 9H",
                "subject": "Science",
                "schoolRefId": "66667705-6C51-4C30-A22A-77CEA0FBCF53",
                "schoolSectionId": "JMS0115:7",
                "schoolYear": "2014",
                "sessionCode": "S1-1415",
                "schoolCalendarRefId": "B0FD06FD-5F35-4D96-B2EA-AA96CD2D0F38",
                "meetingTimes": {
                    "meetingTime": {
                        "timeTableDay": "AB",
                        "classMeetingDays": {
                            "bellScheduleDay": "M,T,W,Th,F"
                        },
                        "timeTablePeriod": "7",
                        "roomNumber": "295",
                        "classBeginningTime": "15:00:00",
                        "classEndingTime": "15:55:00"
                    }
                },
                "students": {
                    "studentReference": [
                        {
                            "refId": "3A80F017-CCAB-4B9A-B54C-01A351041BD9",
                            "localId": "428537",
                            "givenName": "Ima",
                            "familyName": "Peterson"
                        },
                        {
```

```
                            "refId": "DD8D0728-9A59-4CC1-84C3-05588B10C3FE",
                            "localId": "366654",
                            "givenName": "Pamela",
                            "familyName": "Dorsey"
                        }
                    ]
                },
                "primaryStaff": {
                    "staffPersonReference": {
                        "refId": "35A20CE9-E563-41EA-B023-003D765941F1",
                        "localId": "345773374",
                        "givenName": "Allegra",
                        "familyName": "Gallegos"
                    },
                    "teacherOfRecord": "true"
                }
            }
        ]
    }
}
```


User Scenario 2: The Mobile Teacher

Jill is a sophisticated teacher; she's never separated from her tablet and always on top of things. However, it is the beginning of the school year and she is having trouble remembering all her students' names. She soon finds that IT has installed an app that gives Jill the class list she needs, right on her tablet.

Behind the scenes, there is a server application that is servicing Jill's request that is surprisingly similar to Jack's software. After all, both are collecting rosters of the people they serve. The difference is that the server application is responsible for multi-user authentication. Teacher Jill already has an account in the school's own system. When Jill logs in, the software servicing her requests accesses the authentication system, finds out her refId and then makes the following call for her.

Known Information:

```
"xStaffRefId" : "813C565F-366F-4E03-8598-00424085D17A"
```

This allows her server to retrieve all her rosters and let her select the section she is currently teaching.

URL:

```
http://163.153.114.103/api/requests/xStaffs/813C565F-366F-4E03-8598-00424085D17A/xRosters.json
```

Request Headers:

```
Accept: */*
Accept-Encoding: gzip
Authorization: SIF_HMACSHA256
UFVUX1NFU1NJT05fVE9LRU5fSEVSRTp5ejcwa1ViTVMrcE9RcFhpYlRxM3p3T2hJYWs0OUtuTXVjKzZML
zgxUm9rPQ==
serviceType: SERVICEPATH
Timestamp: 2020-06-28T02:22:17.365Z
```

Attendance

An enhancement Jill will soon realize she would really like with her mobile class roster, is the ability to take attendance. We have objects to make marking attendance possible, simple, and efficient. Additionally, we are mapping out how to capture and share other kinds of marks just as directly related to learning as being present.

Going Beyond Roster

Adding Enterprise Features

The SIF Specifications include hundreds of objects from which solutions can be designed and built. Anyone can start to build a product or service modeled on these to provide the marketplace with options that go beyond any single xPress API. In addition to this, there are many Infrastructure advantages, such as better scaling and advanced privacy controls, that can be realized above and beyond this particular API, all of which can be found in the latest SIF Infrastructure Implementation Specification

As Roster Matures

It is important to recognize that many things grow better over time. This indeed has already happened to the API laid out in this document. Enhancing them to leverage not only SIF 3's ability to read data but also to write it. The Access 4 Learning Community invites you to learn along with us.

Where Privacy Comes In

Ok, the contract is all signed between marketplace provider and customer, the deliverables are clearly defined, and everything is outlined in the roles and responsibilities of each party.

Ready to go, right? No.

In most cases the next call between the two parties to answer the question "How do you want us to deliver on X, Y and Z?". This is especially true when it comes to student data privacy issues which usually outlines the need for vendors to use "industry established best practices/standards". The issue is that for the education vertical, no practices or standards exist.

The Communities Toolbox

The Global Education Privacy Standard, GEPS[7], is a PK-20 global set of data privacy obligations (obligations) that can be aligned to contractual clauses as well as technical control benchmarks. GEPS includes open XML code (PODS) to transfer privacy obligations between controllers and processors bridging the gap in understanding education data protection expectations. GEPS allows for organizations to choose the SDPC standard suggestions or use other existing standards, (i.e., IEEE, NIST, ISO, etc.) to set their own expectations between vendors and customers on managing student data.

We suggest if you are an end user OR marketplace provider, that you get involved proactively in this work to assure your needs are met[8]. With the growth of the Alliance, and demands for better data privacy oversight, this work will be the foundation of how all products will validate their controls over student data.

[7] *For more information about GEPS, start here: https://privacy.a4l.org/geps/*

[8] *For further information on how to get involved, please go to: https://privacy.a4l.org/get-involved/*

The Place for Effective Privacy

Schools, administration officials, classroom teachers and education authorities are faced with an overwhelming array of applications to assist in a student's learning, school administrative functions and community communication tools. Data may travel from inhouse on premises, highly trusted sources to external applications which may not be as trusted. A paper or digital contract is the only primitive protection for this data exchange.

With the release of SIF Infrastructure Specification 3.3, an approach has been introduced that specifies a machine-readable form of the contract and provides a mechanism that governs the process by which an external application requests data. By referencing the correct version of the contract in each request, appropriate filters are able to be applied to the data returned on each request; thereby ensuring the external application receives only the information that it should. By referencing the digital contract, the external application is also acknowledging certain obligations it must adhere to when it comes to handling the returned data. This digital contract is called a Privacy Obligation Document or POD.

Privacy Obligation Document (POD) Components

1. POD - Privacy Obligation Document: An artifact derived from a paper contract which contains details of the parties involved, the data which can be transferred from one party to another, details of the technical benchmarks which must be adhered to (e.g., encryption levels) and details of any additional parties which may handle the data.

2. POD Lookup Service – Officially the "Privacy Obligations Registry Utility Service" this provides a means by which external applications request and obtain the current POD that applies to them

3. POD Enforcer – Officially the "Data Protection Enforcer Service" this service:
    a. Checks that any incoming requests from external applications are referencing their correct POD
    b. Uses the rules from the applicable POD to clean the raw data being returned in a request, ensuring that a 'cleansed' data set is returned to the requesting external application.
    c. Is placed and configured to honor all payload encryption requirements.

Privacy for the Service Provider

A standard contract (paper or otherwise) is signed off. This forms the agreement between the provider of data (typically the school or school district) – known as the Data Controller and the downstream consumer of the data known as the Data Processor.

1. A POD is then created from this contract by breaking the contract down into the various contract clauses, linking these to obligations, before finally defining benchmarks which call out the standards a Data Processor is expected to honor when they handle the data. In the U.S. please contact the SDPC for details on contract processing tools, in the AU please contact the National Schools Interoperability Program (NSIP)

2. Once a POD has been created, it can then be enforced in any environment. In SIF enabled environments, the POD lookup service and the POD Enforcer are needed to ensure the right POD is being referenced and that the filtering is being applied. In the simple example below, a Data Processor (e.g., 3rd party attendance package) is assumed to have already obtained the correct POD and makes a request of the Data Controller (e.g., School District).
    a. The POD is registered with the POD Lookup Service, with its identifying Data Privacy Marker (DPM). (Using an HTTP PUT request to the Privacy Obligations Registry Service)
    b. The POD is made available to the POD Enforcer, with its identifying Data Privacy Marker (DPM)
    c. Data requests from data processors include the applicable DPM and are mediated by the POD Enforcer.
    d. For each data request received from a data processor, POD Enforcer ensures:
        i. The POD identified by the DPM provided in the data request, is the most up-to-date and current POD for that data processor.
        ii. Applies the data cleansing rules specified in the applicable POD, to the data received from the Service Provider, before the cleansed dataset is returned to the data processor

Privacy for a Data Processor

A Data Processor (e.g., 3rd party attendance package) is assumed to have already obtained the correct POD and its identifying DPM and makes a request of the Data Controller (e.g., School District).

1. A data request is made including a DPM that identifies the applicable POD.

2. The POD is checked by making a call to the POD lookup service (using the DPM).

3. The data request is forwarded to appropriate backend systems and raw data is extracted. The POD Enforcer applies the data cleansing rules from the applicable POD, ensuring the data conforms to the fields and other conditions the Data Processor is entitled to receive.

4. Filtered data is returned to the Data Processor

SIF Data Model Specification: Comparison

Moving from one version of anything to a new version can always have scary pitfalls. The central question always revolves around "what's new" and "is it worth it for what I want"? This is true whether you are making your initial "leap" into SIF or looking to enhance the functionality of your product by a new version of the Specification. Below is a quick "cheat sheet" for you to determine which version of the SIF Specification might be right for your use case. This comparison sheet will continue to be updated as the Unity data model continues its maturation and alignment to Common Education Data Standards (CEDS) federal work.

Feature / Description	SIF Data Model Specification (North America)		
	2.7M	2.8	Unity
Assessment *Assessment concentrates on ensuring that assessment results can be made available in a timely manner and that the results are transmitted in a meaningful manner.*	X	X	X
Assessment for SIF 3 *Assessment rehauled to primarily support adaptive assessments. In order to maximize backwards compatibility, the classic Assessment objects where included in Unity.*	X		
Authentications *Authentication allows a system that stores usernames and/or passwords to share them with other applications through SIF.*	X	X	X
Common Elements *Complex elements designed for reuse that are expressed in the XML Schema as having their own roots. These have been converted to Common Types in newer releases.*	X		
Data Warehouse *Data Warehouse encompasses the detailed data that schools, and parents need to improve student achievement and organizational effectiveness.*	X	X	X
Facilities and Energy Management *Energy Management reports energy usage and associated environmental information; and conveys the results of the analysis of that information in standardized reports.*	X	X	X
Food Services *Food Services allows computer applications for the food service sector of the education industry to effectively exchange information with all SIF applications.*	X	X	X

Feature / Description	SIF Data Model Specification (North America)		
	2.7M	2.8	Unity
Grade Book *Grade Book supports the entire SIF structure, allowing for easy, bi-directional transfer of data integral to grading.*	X	X	X
Human Resources and Financials *Human Resources & Financials defines the SIF specifications for human resources and financial application software.*	X	X	X
Identity Management *Identity Management empowers sharing and using digital identities within an educational enterprise.*	X	X	X
Infrastructure Payloads *The Global SIF Infrastructure (now defined separately) leverages pre-existing standards to define secure, efficient, and increasingly private interoperable sessions.*	X		
Instructional Services *Instructional Services covers information needed in instruction and educational service delivery in the classroom and instructional settings.*	X	X	X
Library Automation *Library Automation allows for reporting on the overall status of library including checkouts, overdues, fines, and refunds.*	X	X	X
Pluggable Code Sets *Both a relaxing of validation around most fields that hold codes and corresponding new attributes throughout the data model, allow for the localization and documentation (URL) of the code sets used.*			X
Professional Development *Professional Develop includes training and assessment of staff particularly as it relates to qualifications and requirements over time.*	X	X	X
Special Programs *Special Programs enables effective data exchange regarding special education and other individualized programs.*	X	X	X
State Reporting *State Reporting adds the fundamental concept of a State Education Agency and other needed fields to the SIF (NA) Data Model for state reporting.*	X	X	X

Feature / Description	*SIF Data Model Specification (North America)*		
	2.7M	*2.8*	*Unity*
Student Information Systems *Student Information Systems data includes student, teacher, school, and enrolment and is core to most educational agencies and processes.*	X	X	X
Student Record Exchange *Student Record Exchange is designed to streamline the enrolment process when moving a student between schools.*	X	X	X
Transportation and Geographic Information *Transportation and Geographic Information provides bus planning and location information from the transportation department to the rest of the school community.*	X	X	X
Vertical Reporting (& StudentLocator) *Vertical Reporting comprise alternative objects and methods for moving, matching, and verifying information up to a reporting authority.*	X		
xPress IEP *Information to support Special Education students transferring into a new educational institution.*			X
xPress Roster *A simpler set of core data objects that can be used by itself or in conjunction with the rest of the rich enterprise data.*			X
Zone (Functional) Services *Functional Services define a beginning, middle, and end to a multistep process. Similar functionality is empowered by the Global SIF Infrastructure, but details must be defined anew.*	X		

Before You Write Code

The next two sections of this document cover how you *might* go about building a SIF Adaptor for your software. This is done by introducing two open frameworks that directly support the (global) SIF Infrastructure and load the data model in a modular way. However, you should not assume that this is the best path for your organization. You might prefer the benefits of working with another company that specializes in SIF solutions.

Need something fast?

There are commercial products out there that can tap into popular databases to have a quality SIF Adaptor up and running very quickly. How quickly depends on the database software and data schema that back your software. However, configuring one of these products is usually a quick and easy way to meet your Unity support needs.

Further reading:

- Ecosystem Empowerment Guide
- Unity: Marketplace Providers Readiness Scoresheet
- SIF RFP Language
- Questions for your Integrators

All the above documentation can be found here: https://data.A4L.org/implementation-readiness-planning/

Prefer support from SIF experts?

If you don't consider SIF a core component of your organization, you *may not* just want software written by those that do, but access to those people. We highly recommend that those who utilize a 3rd party solution, partner with a marketplace provider for ongoing support.

Want to share development and maintenance costs?

Let's face it, writing software isn't cheap and maintaining it is a very real, yet often unexpected, liability. This is a great way to realize the benefits of an ongoing partnership with those who specialize in these matters. However, this is also a great way to save some costs by sharing them with the other customers of the commercial SIF solution you have chosen.

Concerned about the future?

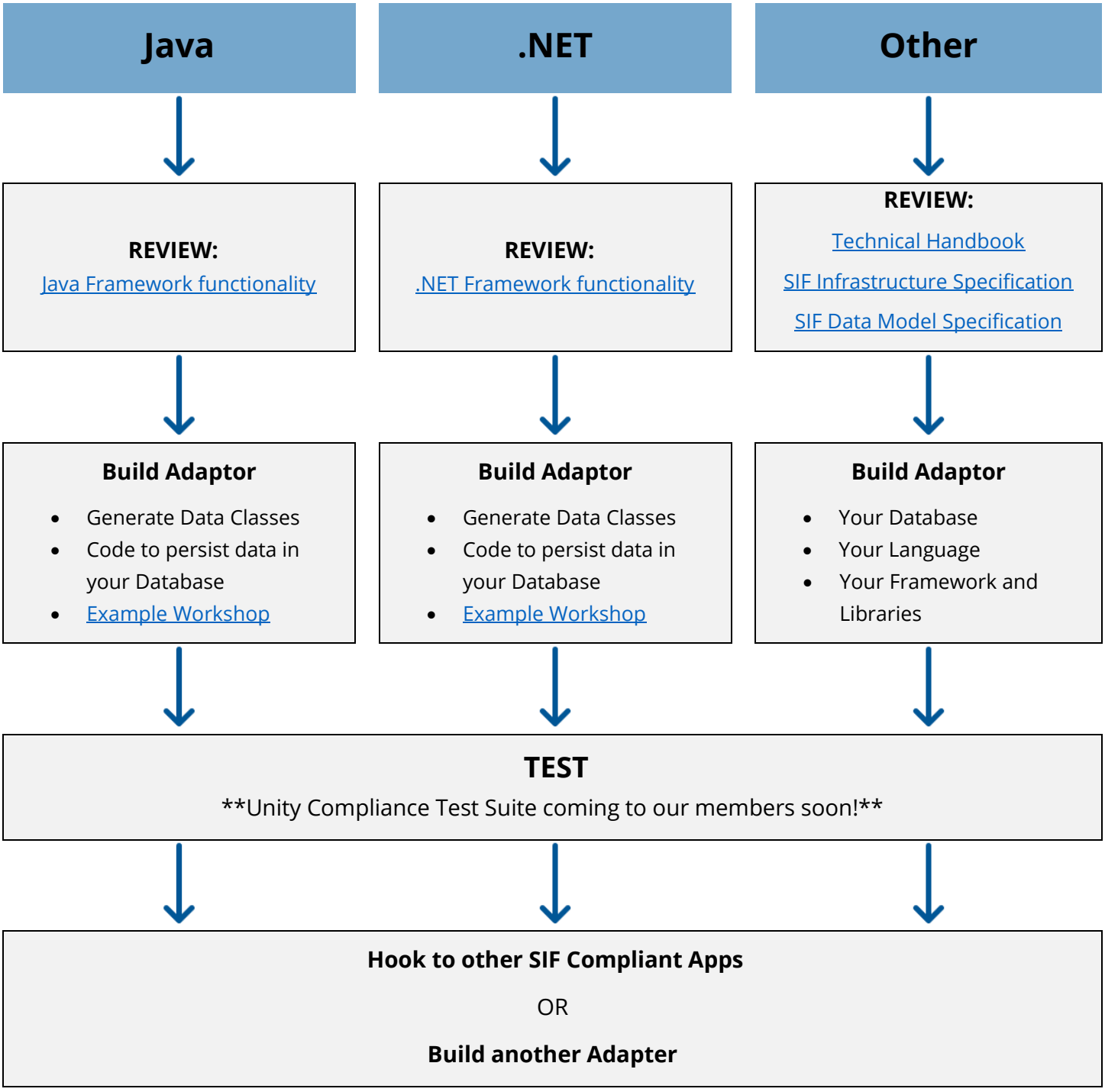
SIF is a set of standards that improves over time and adopters often want to take advantage of new features. If you are concerned about keeping up, this is another opportunity where a partner can make all the difference. Be sure you understand how each of these points is covered by any agreement you sign.

Still want to build?

No problem! Take control of your own destiny by continuing to read this section and the two that follow.

SERVICE PROVIDER AND/OR CONSUMER ADAPTER*

Start by choosing your preferred programming language:



* This flowchart assumes you are seeking to create your own SIF Adaptor.

Those considering building a SIF Agent using one of the open frameworks should be aware that they are available under a Commercial Friendly License.

Java Open Framework

Introduction

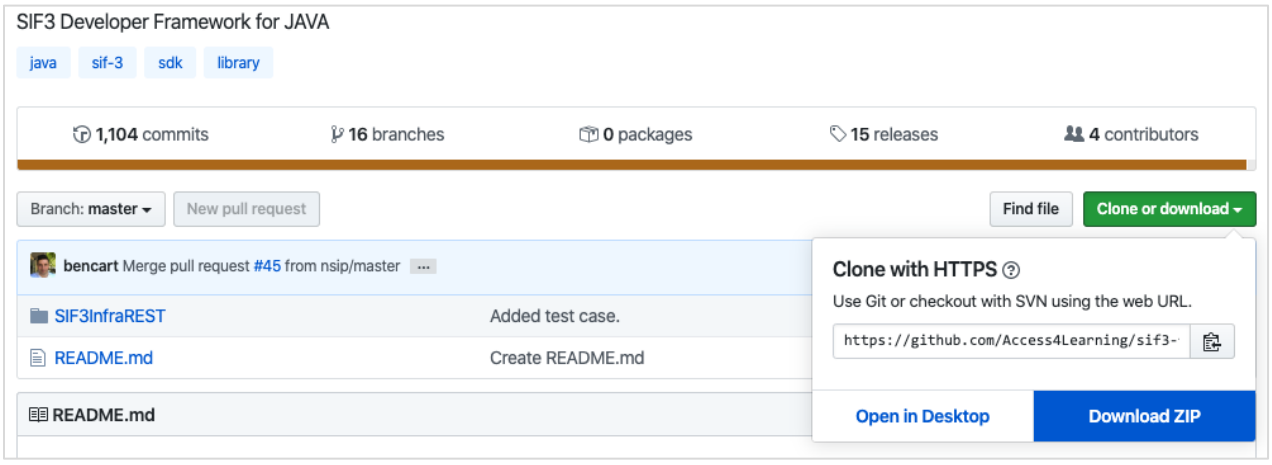
A4L's developer community has produced SDKs for developing SIF3.x Adapters in Java and C# and have made these available as frameworks that can be used openly for building Services/Adapters for Unity and other data models around the world by utilizing the SIF Infrastructure. This chapter will comprise my experience in getting started with this framework utilizing my usual technology stack. These tools are different from those used to develop and test the framework, so this will be both a good test and an opportunity to add to the official documentation.

To see what features are currently included please visit: http://data.a4l.org/technical-support/

Downloading the Framework

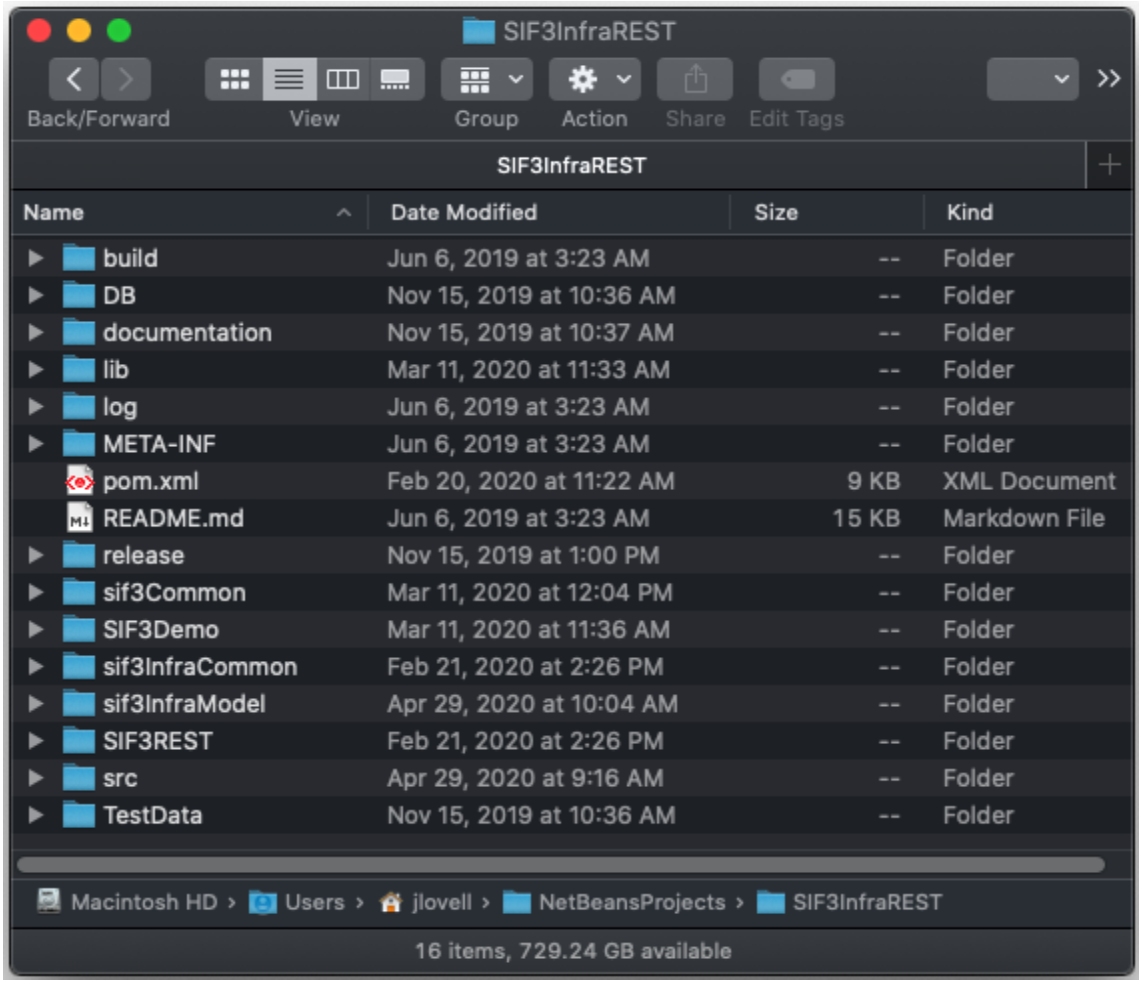
For the purposes of this example the global repository of the Java Open Framework will be used. Just be aware that this repository only receives stable features that have been developed and tested elsewhere. For this reason, it lags behind when it comes to incorporating new features.

Visit the global repository: https://github.com/Access4Learning/sif3-framework-java[9]



[9] *To see the collection of Open Source tools made available by the Access 4 Learning Community please visit: http://data.a4l.org/technical-support/*

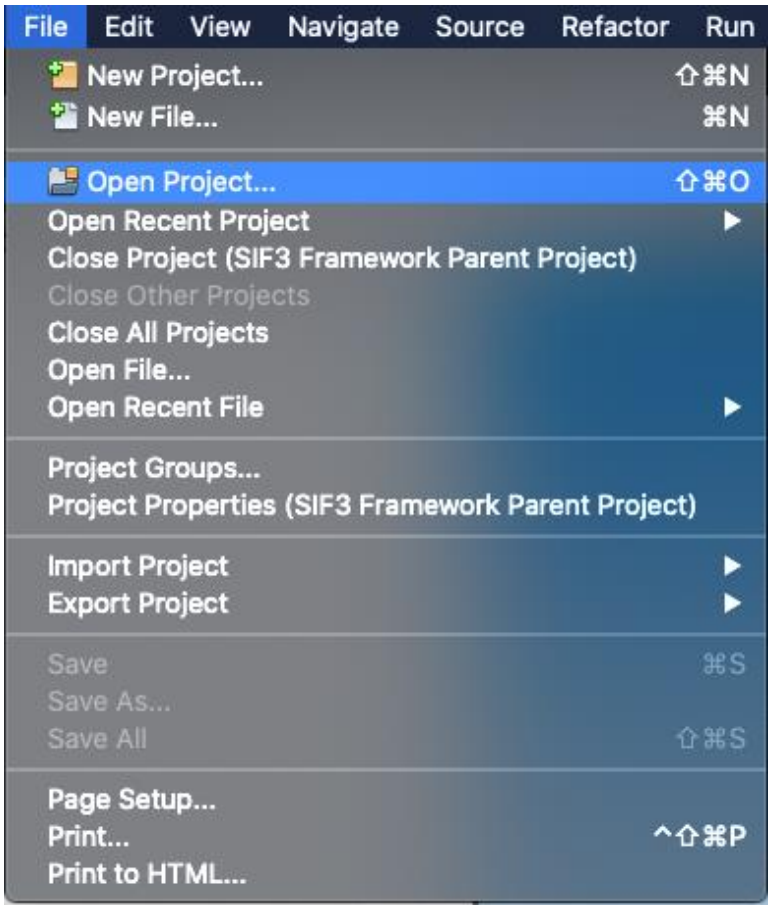
I used the “Clone or download” button and the “Download ZIP” function. However, those familiar with Git may want to clone the repository in order to more easily keep their code up to date.



I then expanded the .zip file and my copy looked like the above. I then placed it alongside my other Java projects.

Opening the Framework Workspace

Because this project has been migrated to Maven[10] it should work with a variety of integrated development environments (IDEs). The official documentation (see further reading) covers Eclipse and IntelliJ; however, I usually develop in NetBeans so that is what I demonstrated here. I also did this work on a UNIX® Certified Product[11], so the paths in this document reflect the choice of operating system. The primary author of the Java Open Framework developed it on Windows and the paths in the downloaded files serve as a solid example for others using that operating system. Other operating system differences are handled by the Java software platform.

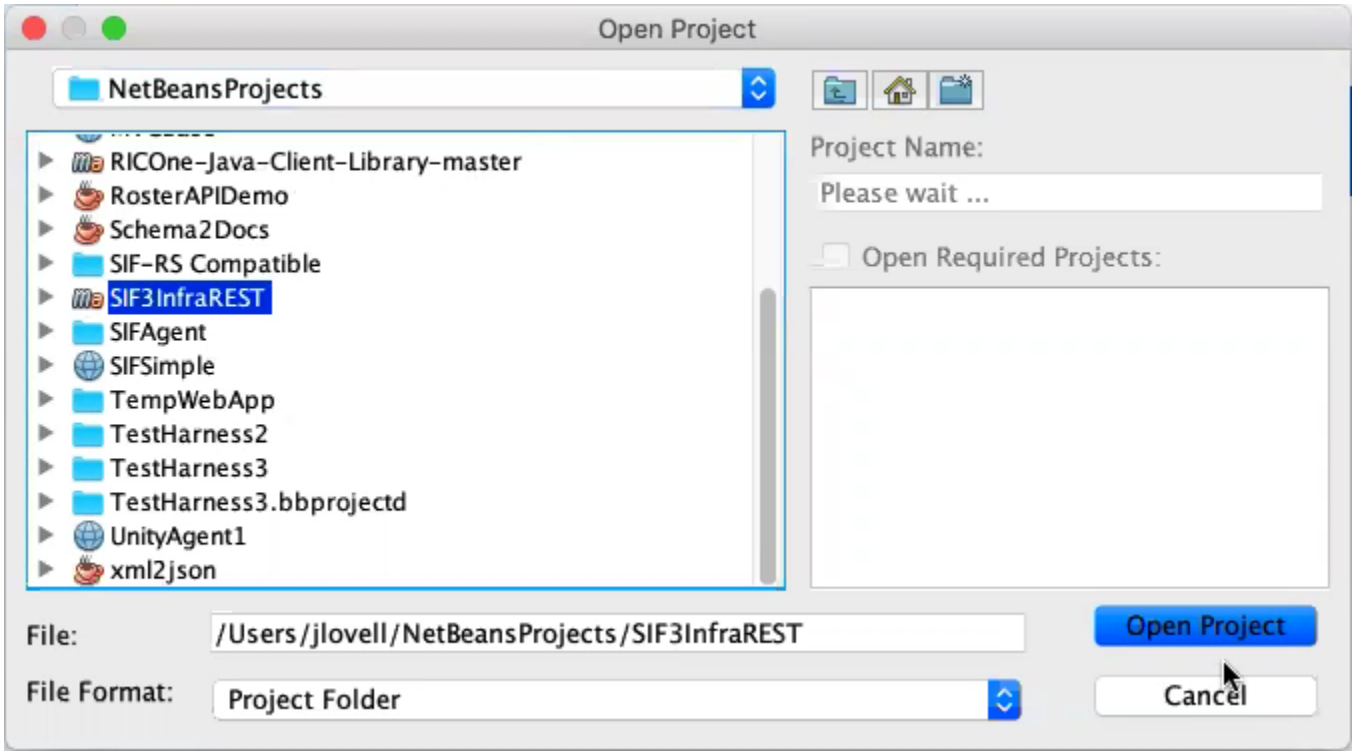


I then opened the project that I had downloaded.

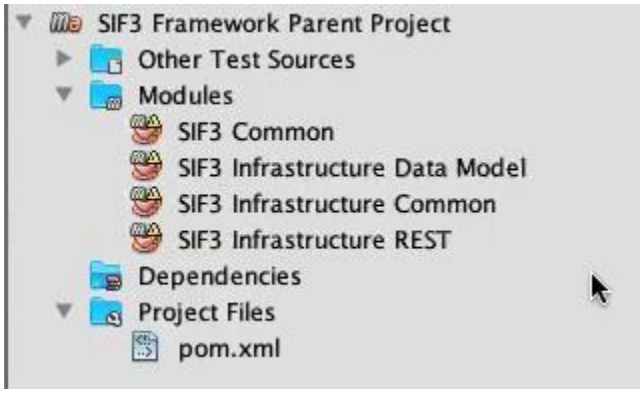
---

[10] *If you need further information about Maven or a newer version (at least 3.3) you can learn more here: http://maven.apache.org/*

[11] *UNIX® Certified Product https://www.opengroup.org/openbrand/register/*



I navigated to the SIF3 Developer Framework for JAVA and opened it.



Building the Framework

The official documentation (see further reading) lays out six important building instructions, included here as they are addressed.

1. The SIF3 Framework libraries are not yet in a global Maven repository. This means you need to build the SIF3 Framework yourself to get it into your personal or organization's Maven repository or you need to manually add the libraries in the release directory to your Maven repository.

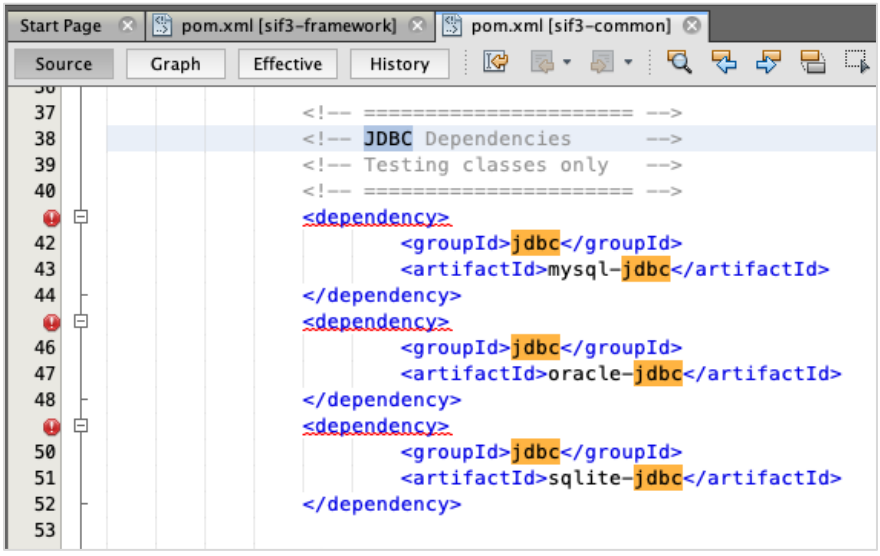
By opening the project and working to build our own copy I have already addressed the first.

2. As part of the Maven “clean install” goal of the SIF3 Framework build, a few JDBC Driver libraries will be installed in your Maven repository. They are only required for tests of the SIF3 Framework. You can remove that in the parent pom.xml of the SIF3 Framework if you do not want to run any tests.

As for the second it involves removing JDBC Driver libraries. These drivers are used by the Open Source community to test the framework.

```
<!--
<execution>
        <id>install-external-mysql-jdbc</id>
        <phase>clean</phase>
        <configuration>
                <file>${project.lib.dir}/jdbc/mysql.jar</file>
                <repositoryLayout>default</repositoryLayout>
                <groupId>jdbc</groupId>
                <artifactId>mysql-jdbc</artifactId>
                <version>4.0</version>
                <packaging>jar</packaging>
                <generatePom>true</generatePom>
        </configuration>
        <goals>
                <goal>install-file</goal>
        </goals>
</execution>
<execution>
        <id>install-external-oracle-jdbc</id>
        <phase>clean</phase>
```

I am going to trust that the community has released a stable framework and start by removing (commenting out) these dependencies (in multiple places).



It is critical to also remove similar entries from the projects under “Modules” before continuing.

3. As with the above there is also a data model library installed (i.e. SIF AU 3.4) in your local Maven repository. Again, it is only used for tests of the Framework and you can remove that install from the parent pom.xml.

The third is related to the second, as it is another dependency used for testing. In this case it is an Australian data model that I will not be using with my agent. So, I will remove it from the Parent Project POM as well.

```
54      <dependencyManagement>
55          <dependencies>
56
57              <!-- ==================== -->
58              <!-- Local Dependencies  -->
59              <!-- Non-Maven Libraries -->
60              <!-- ==================== -->
61              <!--
62              <dependency>
63                  <groupId>sifau</groupId>
64                  <artifactId>sif3-au-datamodel</artifactId>
65                  <version>3.4.4</version>
66                  <scope>test</scope>
67              </dependency>
68              -->
```

```
<!--
<execution>
        <id>install-external-au-datamodel</id>
        <phase>clean</phase>
        <configuration>
                <file>${project.lib.dir}/datamodel/sifDataModel_au3.4.4.jar</file>
                <repositoryLayout>default</repositoryLayout>
                <groupId>sifau</groupId>
                <artifactId>sif3-au-datamodel</artifactId>
                <version>3.4.4</version>
                <packaging>jar</packaging>
                <generatePom>true</generatePom>
        </configuration>
        <goals>
                <goal>install-file</goal>
        </goals>
</execution>
```

I also had to remove the AU data model from the sif3-infra-common and sif3-infra-rest modules.

```
32              <!-- ====================== -->
33              <!-- Local Dependencies     -->
34              <!-- Part of parent Project -->
35              <!-- ====================== -->
36              <!--
37              <dependency>
38                  <groupId>sifau</groupId>
39                  <artifactId>sif3-au-datamodel</artifactId>
40              </dependency>
41              -->
```

Note: Once I had removed the AU data model, I was going to have to add the Unity data model to my project (Important Build Instruction 5).

4. If you build the SIF3 Framework you must first modify the parent pom.xml (in the root directory of the framework). There is a property defined called “project.lib.dir”. Make sure that it points to the framework’s “lib” directory where you have installed the SIF3 Framework. It is the location where it finds the JDBC drivers and data model mentioned under the previous two points.

The fourth instruction is both critical (the only one in bold) and specific to the software configuration of the system being used to build the framework. It is to point the framework at the framework’s “lib” directory so it can include libraries not available through Maven directly.

```
<!-- Hardcoded Variable for the non-maven library location -->
<!--project.lib.dir>C:/DEV/workspaces/jbds10.1.0/sif3-framework/lib</project.lib.dir-->
<!--<project.lib.dir>C:/Development/GitHubRepositories/SIF3InfraRest/SIF3InfraREST/lib</project.lib.dir>-->
<project.lib.dir>/Users/jlovell/NetBeansProjects/SIF3InfraREST/lib</project.lib.dir>
```

Note: The needed line above will vary from system to system. See the three examples (two commented out) for a better idea of what you need.

5. In your very own SIF3 Adapter project (not the SIF3 Framework) you need to link a data model library to it. It is assumed that this library is installed somewhere in your or your organization’s Maven repository. An example can be found in the SIF3Demo Project of this framework.

Now that I have removed the Data Model dependencies from the framework, it is not a surprise that the tests that depend on them are now failing. Specifically, our SIF3 Infrastructure Common and SIF3 Infrastructure REST modules cannot complete the build process.

```
91      <build>
92          <plugins>
93              <!-- ============================================================
94              <!--  Test Plugin configuration. Required to turn on/off test
95              <!-- ============================================================
97                  <plugin>
98                      <groupId>org.apache.maven.plugins</groupId>
99                      <artifactId>maven-compiler-plugin</artifactId>
100                     <executions>
101                         <execution>
102                             <id>default-testCompile</id>
103                             <phase>test-compile</phase>
104                             <goals>
105                                 <goal>testCompile</goal>
106                             </goals>
107                             <configuration>
108                                 <skip>true</skip>
109                             </configuration>
110                         </execution>
111                     </executions>
112                 </plugin>
113                 <plugin>
114                     <groupId>org.apache.maven.plugins</groupId>
115                     <artifactId>maven-surefire-plugin</artifactId>
116                 </plugin>
117         </plugins>
    </build>
```

Turning off these tests in the related POM files, allows us to build the framework.

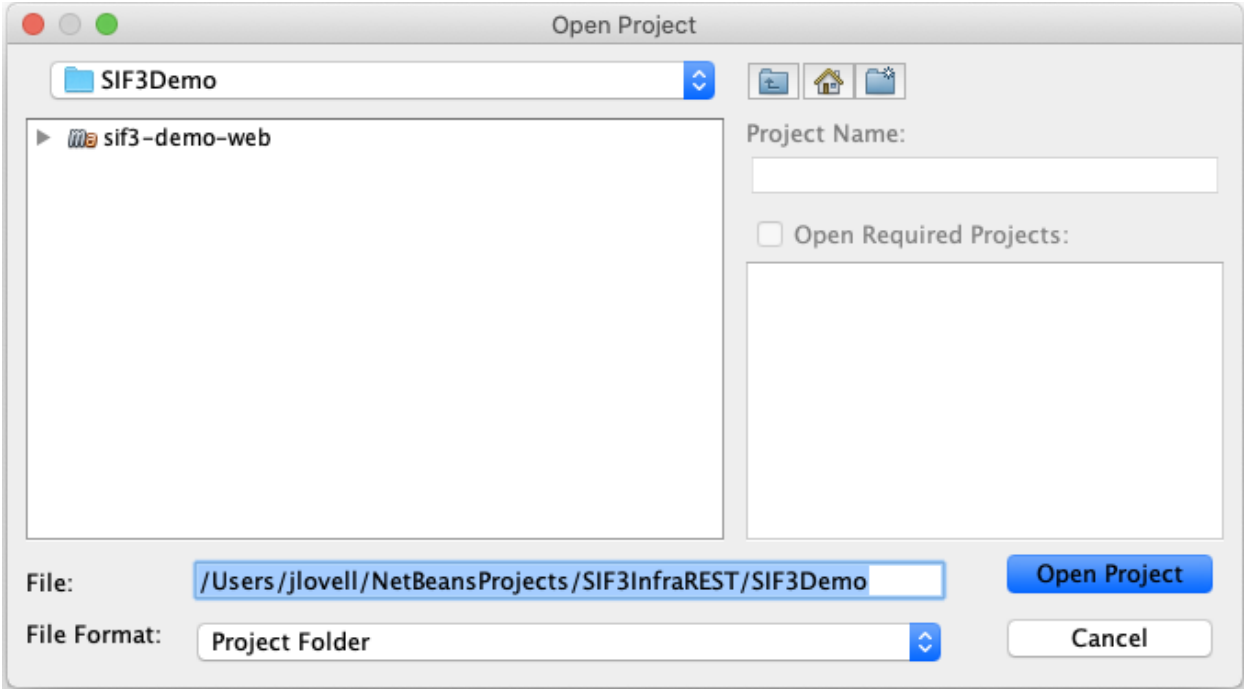
```
------------------------------------------------------------------------
Reactor Summary:

SIF3 Framework Parent Project ..................... SUCCESS [  0.210 s]
SIF3 Common ....................................... SUCCESS [  1.996 s]
SIF3 Infrastructure Data Model .................... SUCCESS [  0.365 s]
SIF3 Infrastructure Common ........................ SUCCESS [  0.216 s]
SIF3 Infrastructure REST .......................... SUCCESS [  0.784 s]
------------------------------------------------------------------------
BUILD SUCCESS
```

Because the SIF3 Framework uses JAX-RS, the build only provides the API. In your very own adapter you may need to add an appropriate dependency to an actual JAX-RS implementation such as Jersey, RESTEasy etc. For details please refer to section 6.5. An example can be found in the SIF3Demo Project of this framework.

The final build instruction is to include a JAX-RS implementation with your final adaptor. JAX-RS is a RESTful Web Services specification for Java that is both popular and has more than one implementation. The demo project I will be using includes a JAX-RS implementation, so I will not have to address this directly.

Demo Project

Opening the demo projects is similar to what I did for the framework.



The official documentation says, “Simply call the Maven clean & install goal in the sif3-demo-web module.” Of course, this assumes I have included the AU Data Model that I disabled when building the framework.

Generate the Unity Data Model

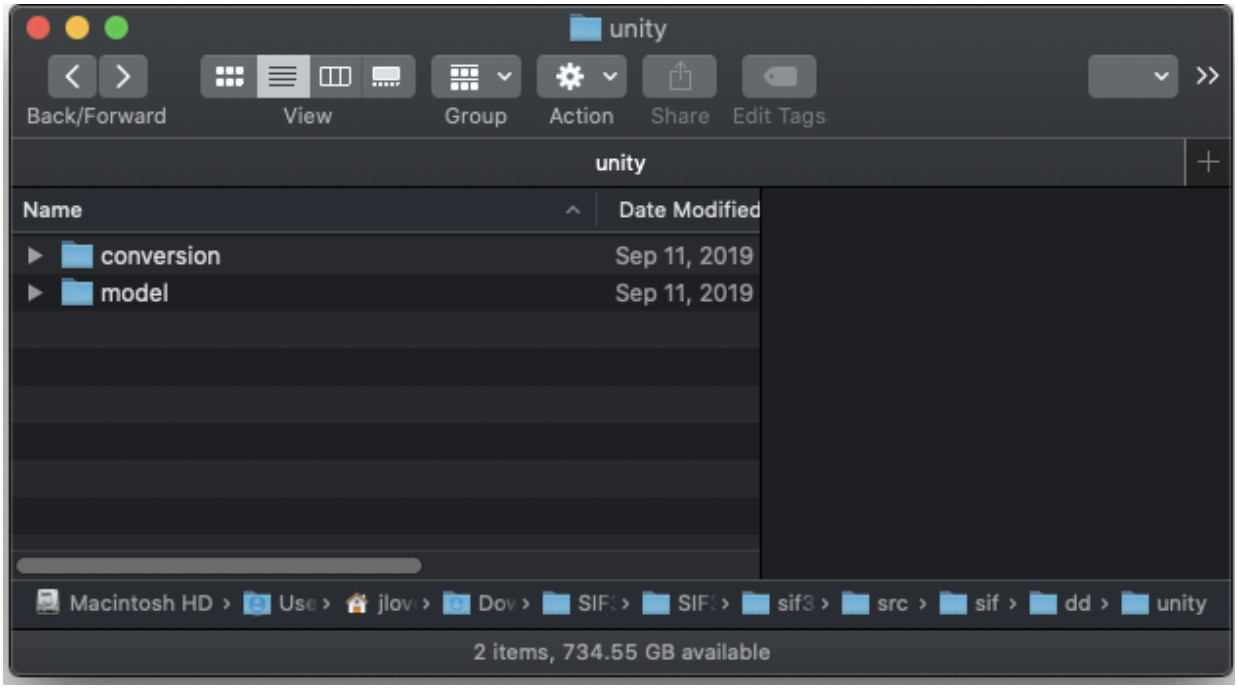
Unlike the framework I need to have a data model and database to wind up with a working example. So I needed to add these components to meet our needs. The data model was expected to come from the framework, so I needed to look back to figure out how to best go forward.

The first thing I needed to do was get my hands on a copy of the Unity Data Model. To best show how to do this with any future release, I generated one myself from the XML Schemas shipped with the specification. However, I also needed to add back in the factory marshaller and unmarshaller provided with the project, a detail I made sure not to miss.

I downloaded the global repository: https://github.com/Access4Learning/SIF3DMGenerator-Java/

Then I navigated to the Unity Data Model and found the data model and marshalling libraries.

Relative Path: /SIF3DMGenerator-Java-master/SIF3DMGenerator/sif3Datamodel/src/sif/dd/unity



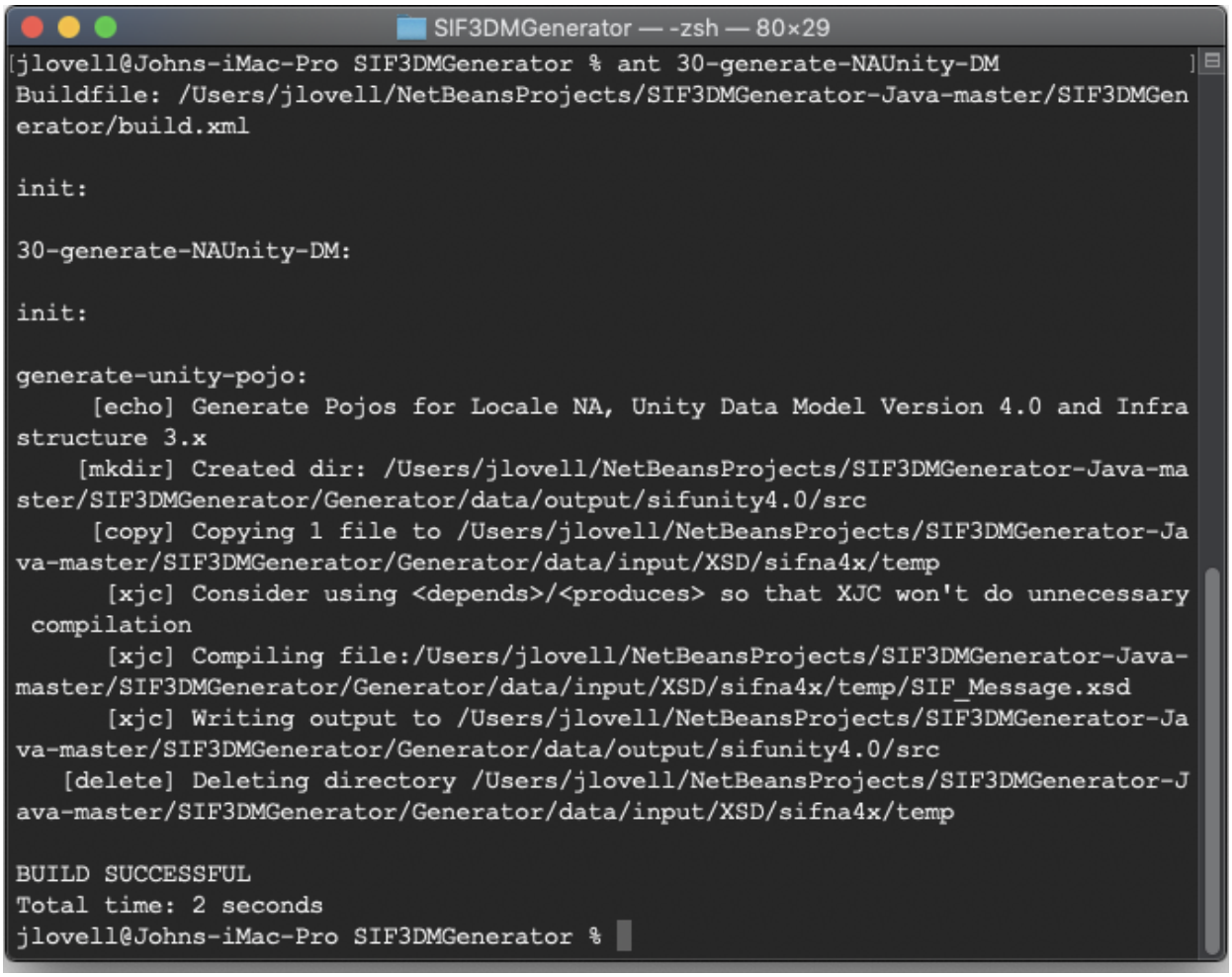
While the Unity Data model was already present, I am recreating it. The steps here will not recreate the factories, so I had to make a copy of the "conversion" folder to add back in later. Also, I deleted the contents of the "dd" folder. This allowed me to confirm both the outcome of the next process and that the project only contained the Unity Data Model when I was done.

It turns out this project is still set up as an Ant project with multiple targets. NetBeans doesn't support this from the IDE so I needed to build my data model from the command line.

First I updated the ant.properties file so it understood that I was trying to build the Unity Data Model.

```
#Local for which the DM shall be generated...

sif.local=na

#The Data Model version using the dot-notation...

sif.dm.version=4.0

#The Data Model version as above but without the dots...

sif.dm.version.nodot=40
```

Then I generated the POJOs by running: ant 30-generate-NAUnity-DM



Next, I copied the conversion folder back into the “unity” directory.

Then I got ready to create the Java Archive (JAR).

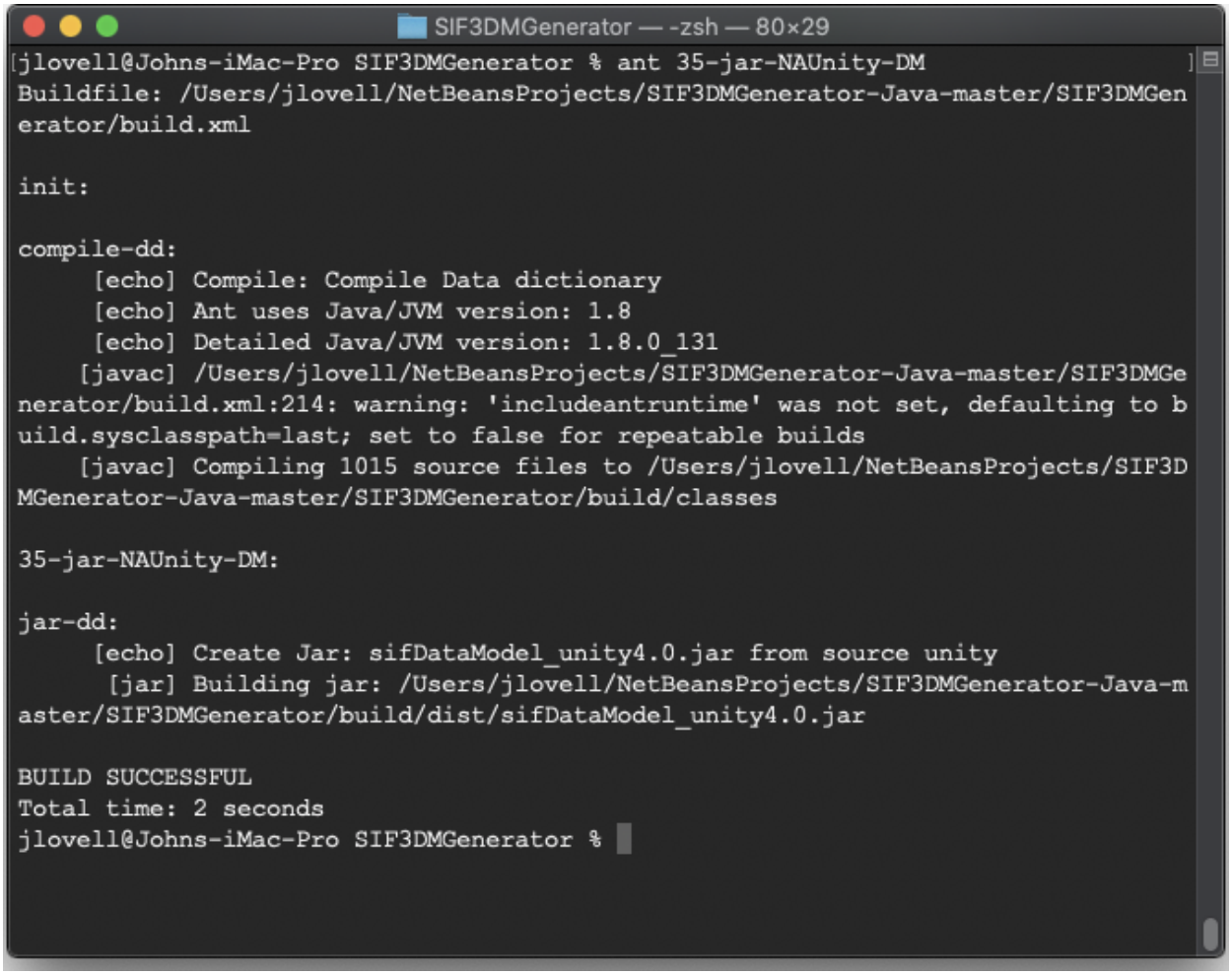
Copy:

Note: The destination directory needs to be empty before copying.

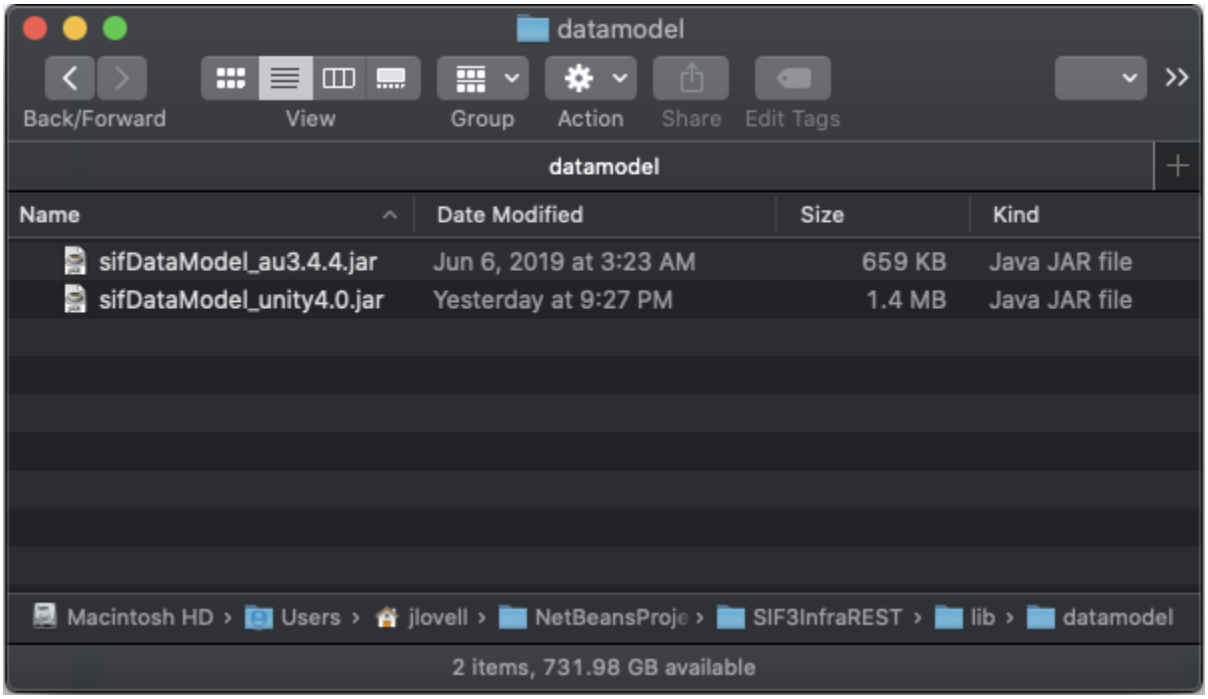
From: Generator/data/output/sifunity4.0/src

To: sif3Datamodel/

Create the JAR by running: ant 35-jar-NAUnity-DM



I found the library (sifDataModel_unity4.0.jar) waiting in the build/dist directory.



Next, I simply placed our results in the /SIF3InfraREST/lib/datamodel directory.

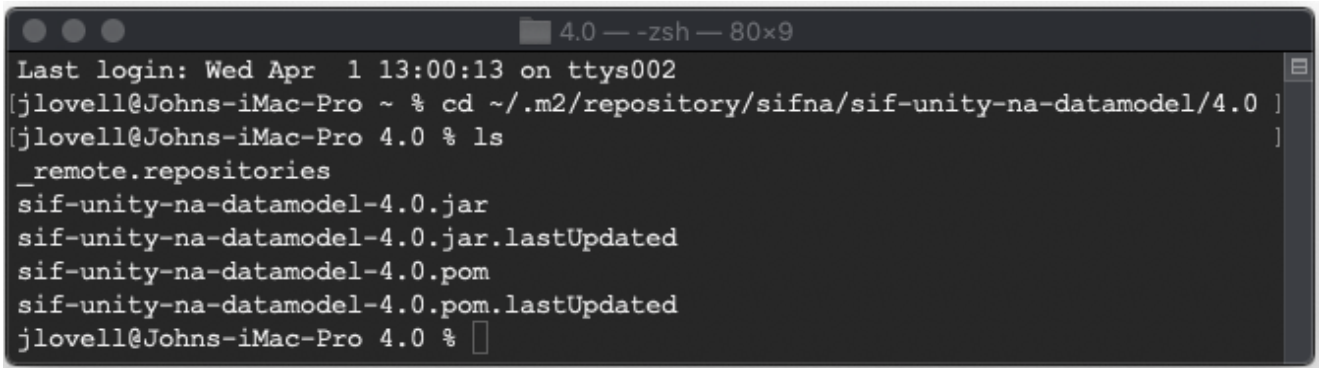
Adjusting Maven for the Unity Release

I included the data model inside my Maven POM.

```
50          <!-- ====================== -->
51          <!-- Local Dependencies     -->
52          <!-- Datamodel              -->
53          <!-- ====================== -->
54          <dependency>
55              <groupId>sifna</groupId>
56              <artifactId>sif-unity-na-datamodel</artifactId>
57              <version>4.0</version>
58          </dependency>
```

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-install-plugin</artifactId>
        <version>2.5.2</version>
        <executions>
                <execution>
                        <id>install-external-na-datamodel</id>
                        <phase>clean</phase>
                        <configuration>
                                <file>${project.lib.dir}/datamodel/sifDataModel_unity4.0.jar</file>
                                <repositoryLayout>default</repositoryLayout>
                                <groupId>sifna</groupId>
                                <artifactId>sif-unity-na-datamodel</artifactId>
                                <version>4.0</version>
                                <packaging>jar</packaging>
                                <generatePom>true</generatePom>
                        </configuration>
                        <goals>
                                <goal>install-file</goal>
                        </goals>
                </execution>
```

Note: Try and try as I might, on macOS I couldn't get the above to work. In the end I determined that the .pom file for the dependency was not being generated. What fixed this was to visit the Maven dependency cache (~/.m2/repository/sifna/sif-unity-na-datamodel/4.0) and try again. For some reason that resolved the issue. I included an example of successful cache creation below. If you are running into trouble, then you can take a look.



Including a Database

Just for testing things out I decided to try SQLite. I was fully aware that this would not perform as well as a regular database and I should use a different solution for any production work I may do in the future.

```
111             <dependency>
112                 <groupId>jdbc</groupId>
113                 <artifactId>sqlite-jdbc</artifactId>
114                 <version>3.7.2</version>
115             </dependency>
```

```
            <execution>
                <id>install-external-sqlite-jdbc</id>
                <phase>clean</phase>
                <configuration>
                    <file>${project.lib.dir}/jdbc/sqlite-jdbc-3.7.2.jar</file>
                    <repositoryLayout>default</repositoryLayout>
                    <groupId>jdbc</groupId>
                    <artifactId>sqlite-jdbc</artifactId>
                    <version>3.7.2</version>
                    <packaging>jar</packaging>
                    <generatePom>true</generatePom>
                </configuration>
                <goals>
                    <goal>install-file</goal>
                </goals>
            </execution>
        </executions>
</plugin>
```

Playing with the Demo

When I built the demo I got a list of errors I needed to directly address.

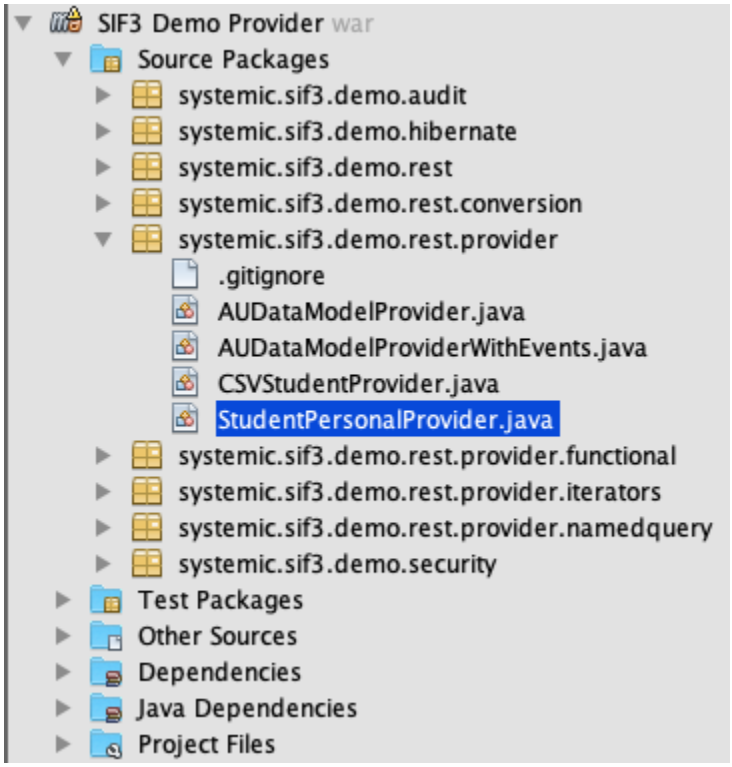
```
--- maven-compiler-plugin:3.1:compile (default-compile) @ sif3-demo-web ---
Changes detected - recompiling the module!
Compiling 25 source files to /Users/jlovell/NetBeansProjects/SIF3InfraREST/SIF3Demo/sif3-demo-web/target/classes
-------------------------------------------------------------
COMPILATION ERROR :
-------------------------------------------------------------
systemic/sif3/demo/rest/provider/SchoolInfoProvider.java:[27,25] package sif.dd.au30.model does not exist
systemic/sif3/demo/rest/provider/SchoolInfoProvider.java:[28,25] package sif.dd.au30.model does not exist
systemic/sif3/demo/rest/provider/SchoolInfoProvider.java:[29,25] package sif.dd.au30.model does not exist
systemic/sif3/demo/rest/provider/AUDataModelProvider.java:[24,30] package sif.dd.au30.conversion does not exist
systemic/sif3/demo/rest/provider/AUDataModelProvider.java:[25,30] package sif.dd.au30.conversion does not exist
systemic/sif3/demo/rest/provider/SchoolInfoProvider.java:[54,40] cannot find symbol
  symbol:   class SchoolInfoType
  location: class systemic.sif3.demo.rest.provider.SchoolInfoProvider
```

...

This was truly a sign of progress because each and every error existed because it could not find the AU Data Model. Yet Maven only tries to find these components after loading all the dependencies, so I knew that my Unity Data Model library was just waiting to be utilized.

At this point, I decided the biggest strong suit of the framework was reducing work of those building Service Providers. So, I set off to cut the demo down to just provide StudentPersonal and demonstrate it with a third party tool.

First, I removed all the source files I didn't want for this scope (Service Consumer and SchoolInfo related).



Then I decided I was going to try to convert the source but not worry about the existing names. For instance files may still be called "AU something" but the important thing was to rewire the demo for the Unity Data Model. See Further Reading (below) for where to get a developer's guide with much more guidance on project creation and source requirements. The idea was to get a sense of the code and what it looks like to have it work!

Note: The AU StudentPersonal and NA StudentPersonal actually share a common lineage, so this will be much more straightforward than one might think.

```
//import sif.dd.au30.model.ObjectFactory;
import sif.dd.unity.model.ObjectFactory;
```

I made changes like the above (sif.dd.au30 to sif.dd.unity) throughout the remaining code to load the Unity Data Model components instead of the AU ones.

```
//import sif.dd.unity.model.StudentListType;
```

I then went and commented out all the components (and the code that uses them) that were still causing trouble. This was necessary because the AU StudentPersonal and NA StudentPersonal objects have differences.

I then removed the Service Consumer tests, like I had for the framework.

```
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 4.328 s
Finished at: 2020-04-03T19:22:47-07:00
Final Memory: 28M/586M
------------------------------------------------------------------------
```

In order to run the demo Service Provider I first needed to configure it. This was done by copying (and modifying as necessary) the included configuration files into the classpath. These were found at /SIF3InfraREST/src/test/resources/config and in its sub directories. I found I had to copy them into the web server's classpath (discoverable by running: ./catalina.sh version or version.bat) instead of adding them to the web application itself. Here is a file-by-file description of how I modified them.

log4j.properties: Just copied.

environment.properties: Converted to relevant Unix style paths.

sif3infra.hibernate.properties: Renamed to hibernate.properties, disabled MySQL and configured SQLLite.

StudentProvider.properties: Updated URIs to reflect our server and converted to relevant Unix style paths.

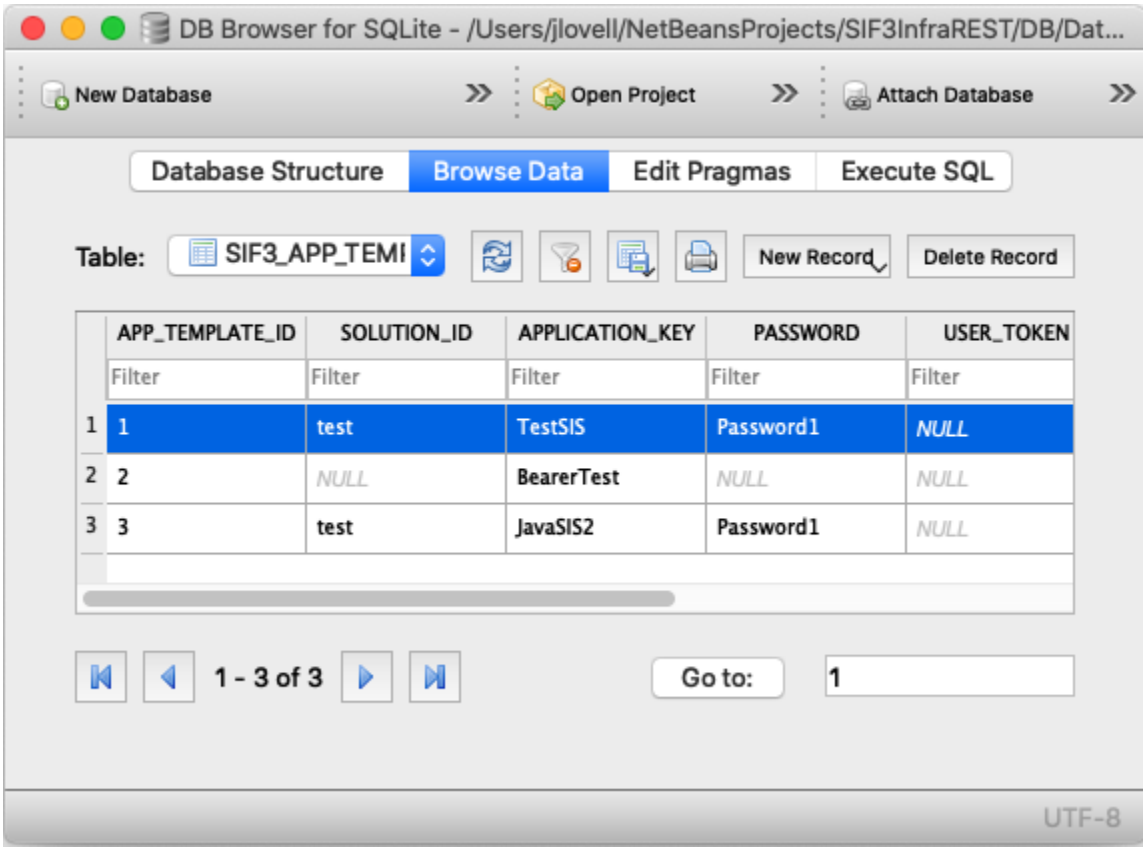
I also had to change the following parameter in the demo projects web.xml file in order to properly load hibernate.properties. Ideally I would have used a more specific filename (like sif3infra.hibernate.properties above) to help prevent conflicts. Whatever I chose to do, I needed to omit the ".properties" file extension for this setting to work properly.

```
<context-param>
    <param-name>SERVICE_PROPERTY_FILE</param-name>
    <param-value>hibernate</param-value>
</context-param>
```

I built and deployed the resulting application to a webserver and can now access the services I have set up.

The first service I accessed was the environment service. Specifically I created an environment. While this isn't necessarily required, it will allow me to discover, and/or confirm the services my software has Authorization to access. This is a great way to manage configurations and detect problems before they are encountered while trying to exchange data. Software can even attempt to overcome configuration issues by making subsequent Infrastructure Service requests.

The first thing I needed in order to create an environment was credentials with which to do so. Strictly for testing purposes, I used my API testing tool's support for Basic Authentication to set the Authorization Header with the necessary Authorization Token. For simplicity, I used one of the preprovisioned environments (TestSIS) for the demo in the example database (SIF3Infra.sqliteDB) for my credentials.



Note: In order to access the SQLite database I used the Open Source tool DB Browser for SQLite (https://sqlitebrowser.org/), however there are many applications you can choose from.

The next piece of information I needed for creating an environment was where (the URL) to direct my request to go. The pieces came from multiple sources:

1. Where the webserver is running

   Tomcat's default example: http://127.0.0.1:8080

2. How the web server disambiguates web applications it is running

   Tomcat use the web application's (.war) filename

   Resulting example from building with the demo web.xml: SIF3InfraREST

3. Where the web application places its RESTful services

   Example from the demo web.xml: /sif3/*

4. The relative path of the environment service

   Known from the (global) SIF Infrastructure: environments/environment

These would result in: http://127.0.0.1/SIF3InfraREST/sif3/environments/environment

Note: Because of the multiple sources of this information and multiple potential technologies between the Service Provider and Service Consumers, the Service Provider must be configured with this information so it can both use and share it. This configuration is done through fields such as env.connector.url in the StudentProvider.properties file.

The last and probably the biggest part of creating an environment is putting together the body. Fortunately, the framework ships with some example templates you can use to more easily arrive at the proper result. For my purposes I started with the contents of /SIF3InfraREST/src/test/resources/config/environments/consumer/template/auTestSolution.xml and modified them to meet my goals.

POST /SIF3InfraREST/sif3/environments/environment

Response time: 95 ms

Request Headers & Body

```
Accept: */*
Accept-Encoding: gzip, deflate
Content-Type: text/plain
Authorization: Basic VGVzdFNJUzpQYXNzd29yZDE=
Accept-Language: en-us

<environment xmlns="http://www.sifassociation.org/infrastructure/3.2.1">
    <solutionId>test</solutionId>
    <authenticationMethod>Basic</authenticationMethod>
    <consumerName>TestSIS</consumerName>
    <applicationInfo>
        <applicationKey>TestSIS</applicationKey>
        <supportedInfrastructureVersion>3.2.1</supportedInfrastructureVersion>
        <dataModelNamespace>http://www.sifassociation.org/datamodel/na/4.x</dataModelNamespace>
        <transport>REST</transport>
        <applicationProduct>
            <vendorName>A4L Community</vendorName>
            <productName>Test Driver</productName>
            <productVersion>0.1alpha</productVersion>
        </applicationProduct>
    </applicationInfo>
</environment>
```

Response Headers

```
HTTP/1.1 201 Created

Content-Type: application/xml
fingerprint: bbbc1573-5e78-4c5e-9bb9-98b2a19fb343
providerid: StudentProvider
Keep-Alive: timeout=20
relativeservicepath: environments/environment
Content-Encoding: gzip
timestamp: 2020-05-01T19:38:58Z
messagetype: RESPONSE
Transfer-Encoding: Identity
Server: Apache-Coyote/1.1
Date: Fri, 01 May 2020 19:38:58 GMT
responseaction: CREATE
Connection: keep-alive
environmenturi: http://localhost:8084/SIF3InfraREST/sif3/environments/d3e9e457-c3ba-4261-bf2f-e3adec969c16
Vary: Accept-Encoding
X-Compressed-By: com.github.ziplet.filter.compression.CompressingFilter/1.7.1
```

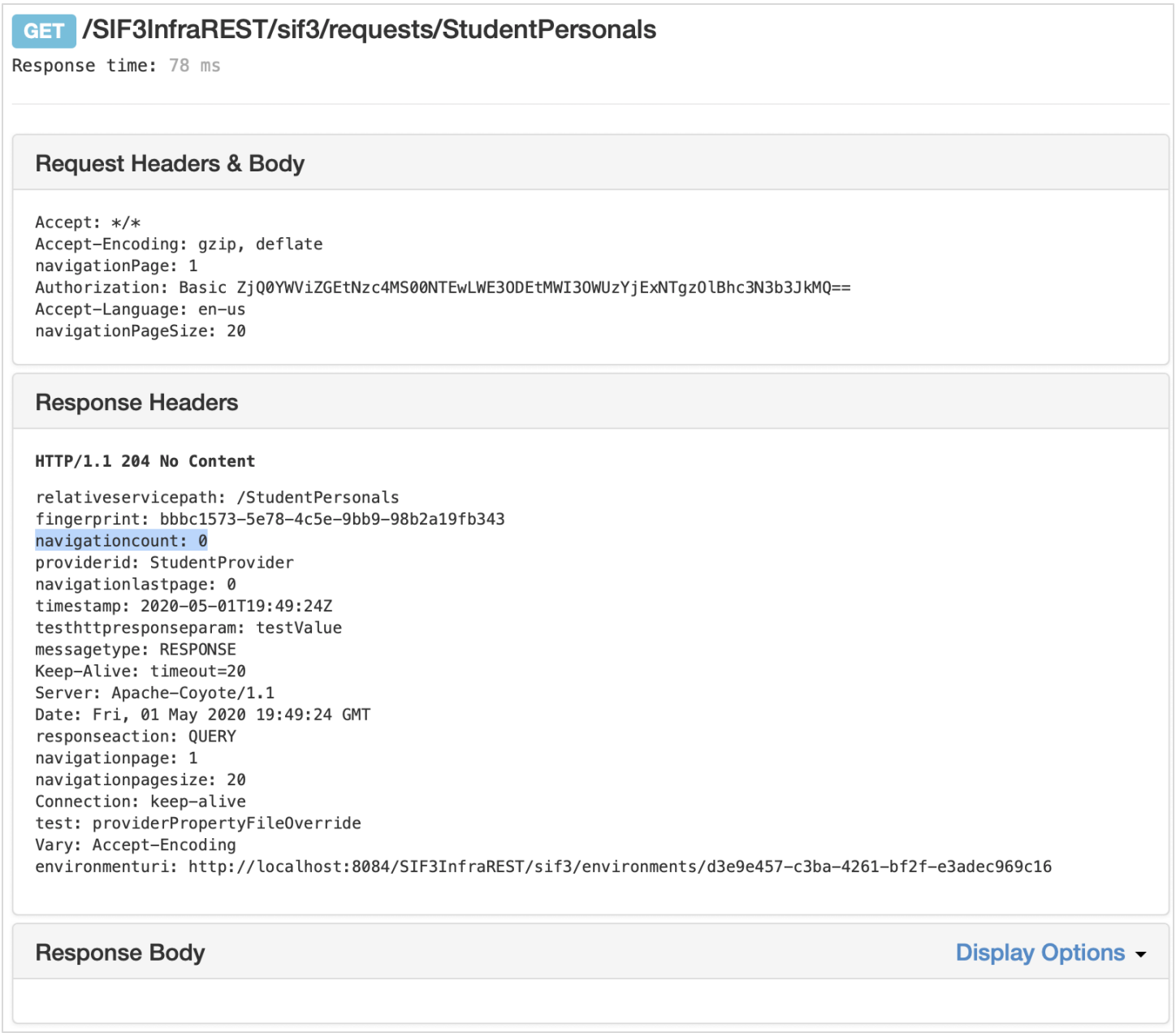
Response Body

Display Options

```
<environment type="DIRECT" id="d3e9e457-c3ba-4261-bf2f-e3adec969c16"
    xmlns="http://www.sifassociation.org/infrastructure/3.2.1">
    <fingerprint>bbbc1573-5e78-4c5e-9bb9-98b2a19fb343</fingerprint>
    <sessionToken>f44aebda-7781-4510-a781-1b79e3b11583</sessionToken>
    <solutionId>test</solutionId>
```

...

To try out object services, I moved the sessionToken to the username portion of the authentication, changed the HTTP method to GET, updated the URL to the path of the collection my adapter is set up to provide, and added the navigation headers to request the first page.



The results accurately reflected that there were no StudentPersonal objects loaded into the Service Provider. It was clear evidence that my SIF Adaptor was up and running properly. When I hadn't swapped in the sessionToken for the username I received a 401 Unauthorized with the corresponding SIF error.

```
<error id="ebcf49a7-40ff-4e31-89e6-6b09760f6473"
```

```
    xmlns="http://www.sifassociation.org/infrastructure/3.2.1">
    <code>401</code>
    <message>Not Authorized.</message>
    <description>No environment exits for the given session token =
UnityConsumer. Ensure that environment is created first.</description>
</error>
```

Reading the details of this error, it seemed to be disconnected from the problem. I thought to myself, isn't the problem a bad username? Then I realized, the session is usually determined from information embedded in the Authorization Token. So, at this point, even my errors are set up and ready to go. I'm ready to build what I need atop the foundation of the Java Open framework and start to enjoy the many supported features[12] of the (global) SIF Infrastructure with a minimum of effort.

Including in Your Workspace

From here you can create code to provide data in a variety of ways based on what is in your application's data store. You may create an Adaptor that runs separately from your (target) application, usually by working with the applications database independently (like the demo). This generally has the advantages of simplicity and modularity. However, the potential to provide feedback to the user is much greater, if you bundle your integration code with an application you control. Different solutions are appropriate for different situations, so choose your path wisely.

Further Reading

The current SIF 3 Developers Guide (Java) can be found in: SIF3InfraREST/documentation/UserGuide

Details on the Data Model build is here: https://github.com/Access4Learning/SIF3DMGenerator-Java/tree/master/SIF3DMGenerator

[12] *A table of supported features can be found here: http://data.a4l.org/technical-support/*

.NET Open Framework

Introduction

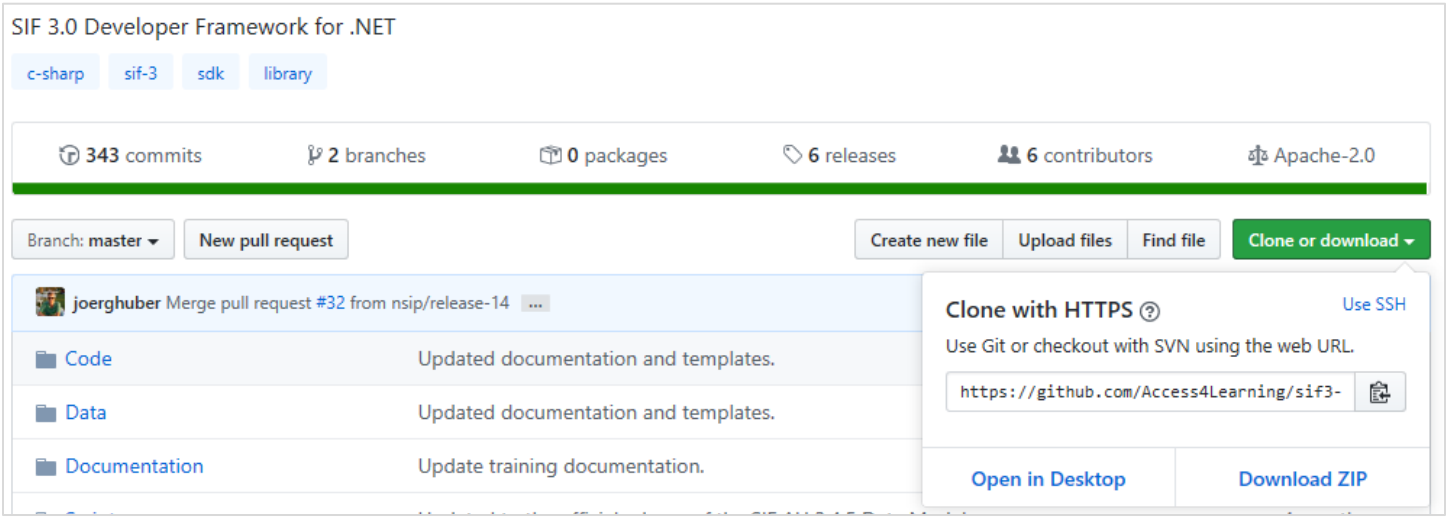
The .NET Developer Framework, like the Java Framework, is designed to help developers build Services/Adapters for Unity and other data models around the world by utilizing the SIF Infrastructure. This chapter will comprise of the authors experience getting started with this framework utilizing the Microsoft suite of tools including Visual Studio and IIS Express.

To see what features are currently included please visit: http://data.a4l.org/technical-support/

Downloading the Framework

For the purposes of this example we will be using the global repository of the SIF 3.0 Developer Framework. Just be aware that this repository only receives stable features that have been developed and tested elsewhere. For this reason, it lags behind when it comes to incorporating new features.

Visit the global repository: https://github.com/Access4Learning/sif3-framework-dotnet

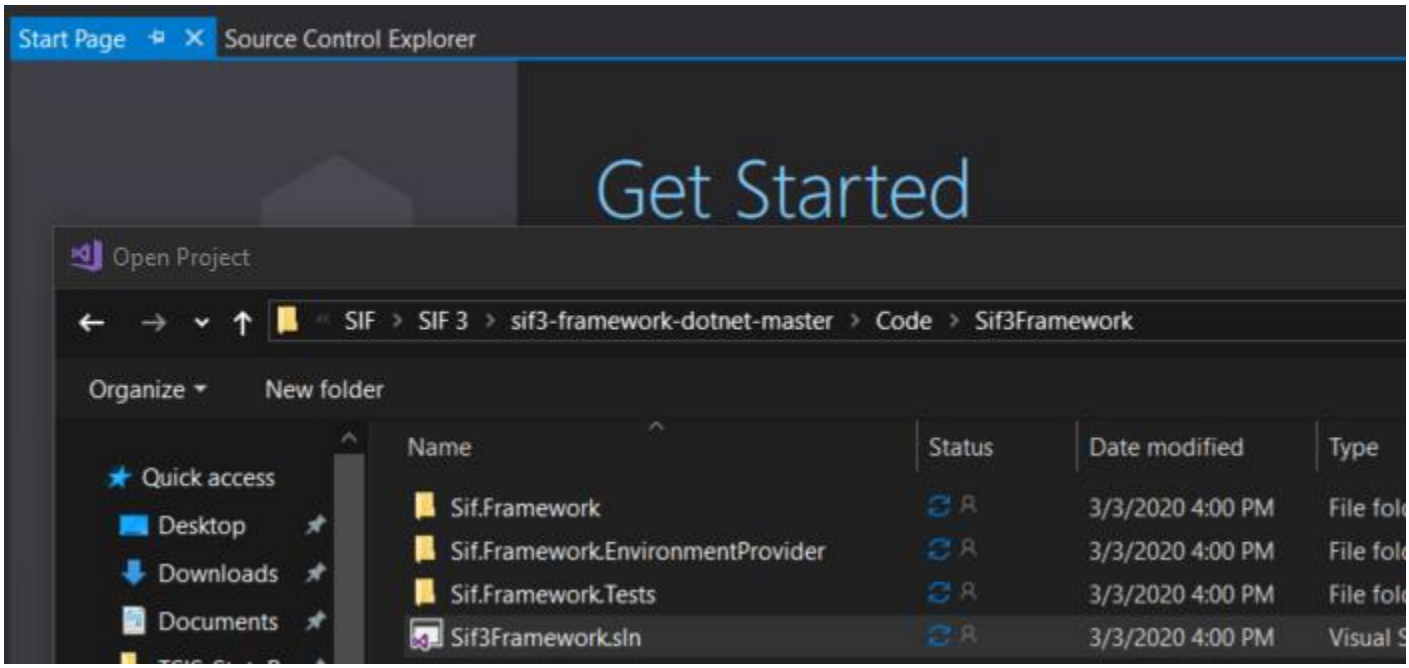


Either download and expand the current ZIP archive or clone the Git repository, to more easily fetch updates. As we will be building the framework, be sure to place it alongside your other Visual Studio projects.

Be sure to expand and retain the folder structure of the entire "sif3-framework-dotnet-master" folder. The projects in the "Code" folder reference dll files in the "SharedLibs" folder, the sample SQLite DB is in the "Data" folder and the scripts in the "Scripts" folder are necessary as well.

Opening the Framework Solution

The documentation states that the project was developed in Visual Studio 2015 using the Microsoft .NET Framework 4.5. That seems to be outdated. VS 2015 presented us with errors related to the version of MS Build and the XML namespace. The solutions successfully opened in VS 2017, but it needs to be version 15.6.6 at a minimum. Earlier versions of 2017 generated errors related to the NuGet packages and .NET compatibility. Opening the solutions in VS 2019 worked just fine. This exercise was completed using VS 2017.



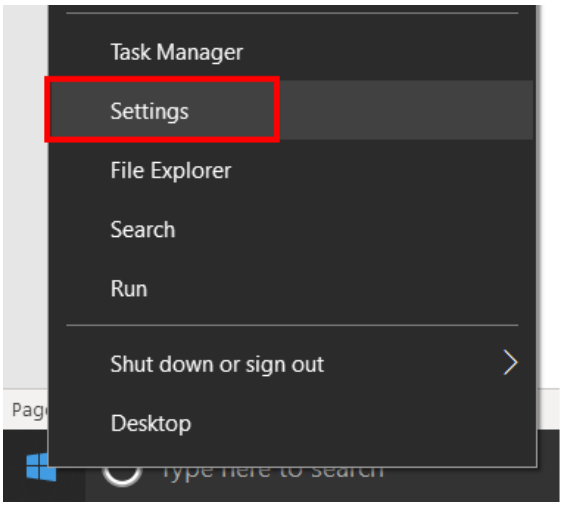
Once the solutions are open, there are several things to do before we are ready to use it to back our own Unity project. The first thing to note is that there are multiple solutions with multiple projects. Each must

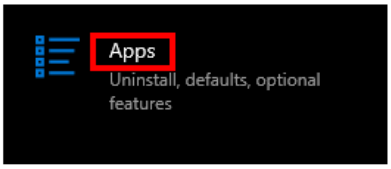
build successfully in order to have a working framework. In the next section the author will build his own copy of the framework for use in his projects.

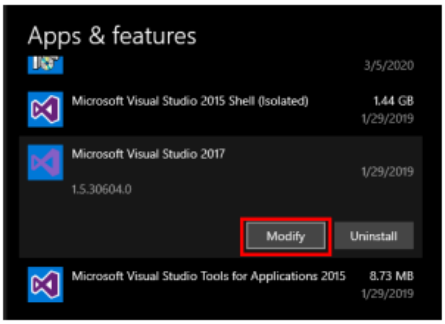
Building the Framework

At this point building the project resulted in errors related to missing assemblies. It was necessary to add the .NET 4.7 components to the VS 2017 install.

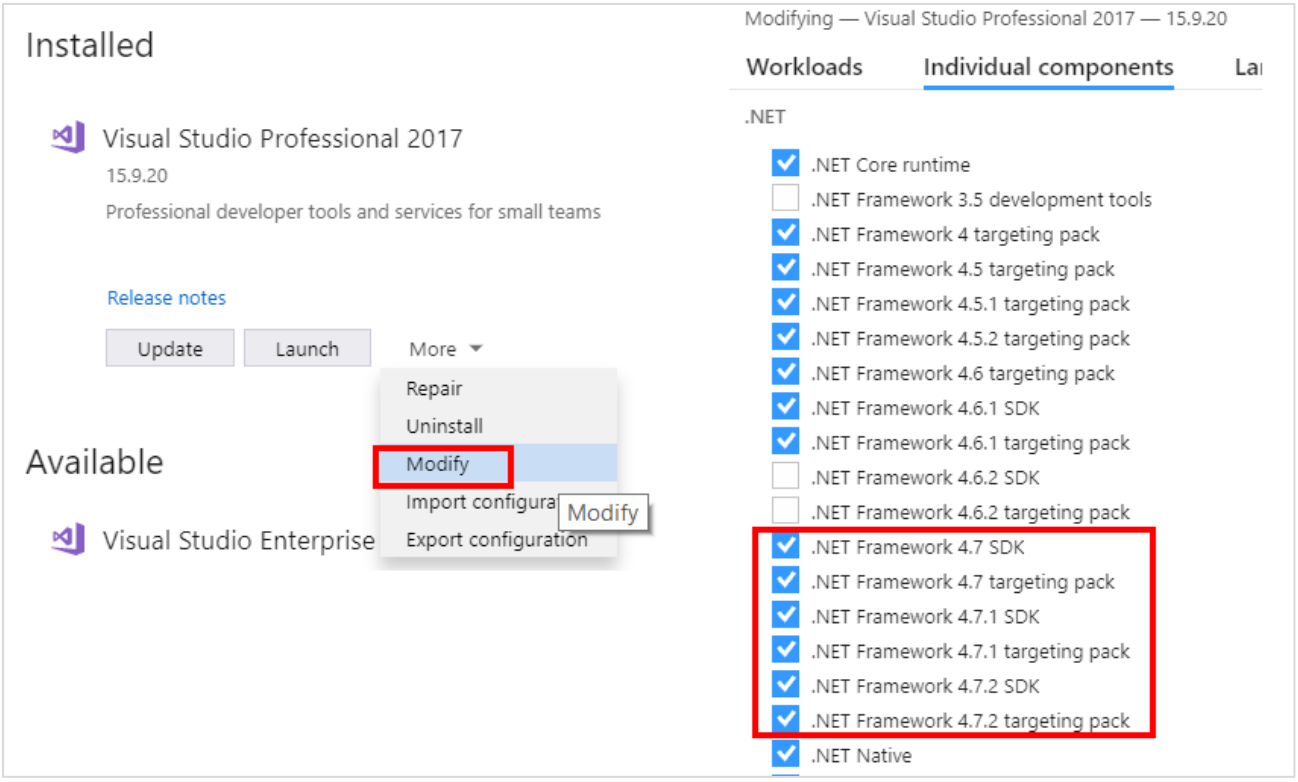
Open the Visual Studio Installer to modify the current installation:





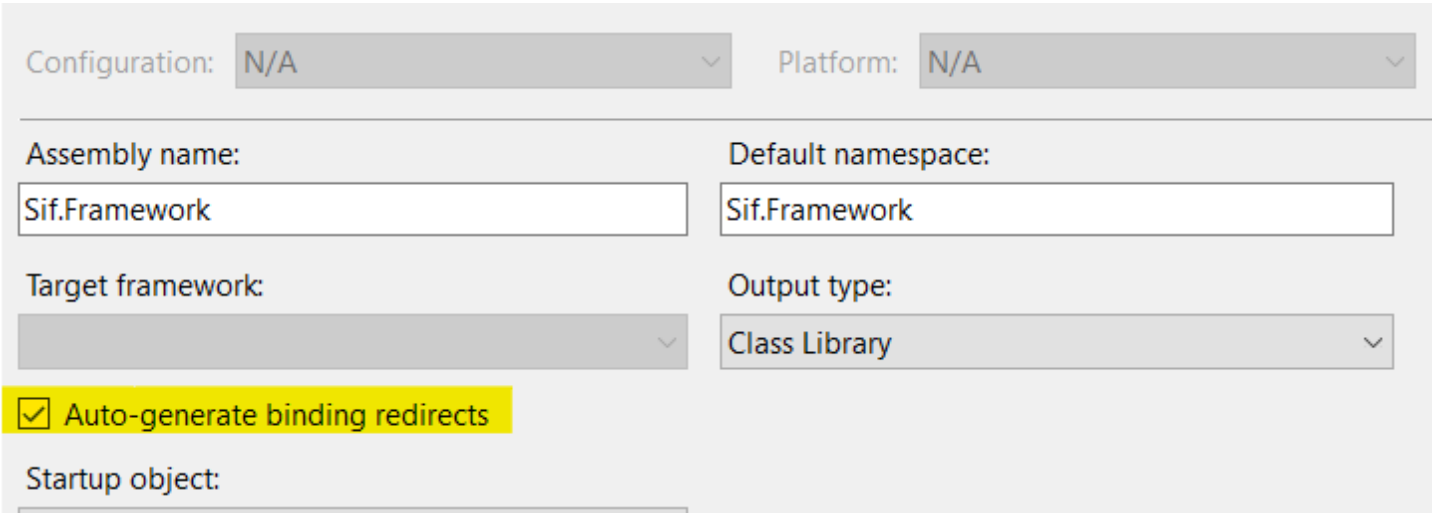


Modify the install and add the necessary SDK and target pack components. The items noted below were missing from this specific VS 2017 install. Your experience may vary.

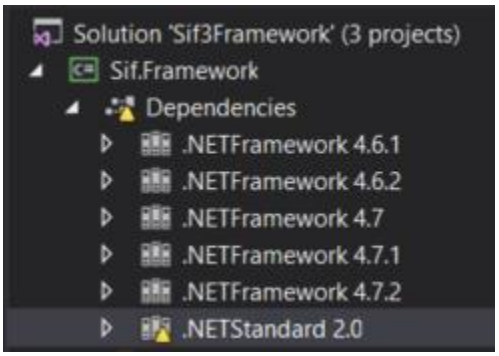


Interesting note, the development workstation in use also had the .NET 4.6.2 SDK and targeting pack installed but it must have been done outside of VS as they are not showing checked in the VS Installer.

Not sure if this next step is necessary but it seems to help the compiler. Check the option to "Auto-generate binding redirects" in the properties of the Sif.Framework project.



After that the build was successful. There is still a warning on the .NET 2.0 dependency… I'll circle back around to this later



Specification Project

Opening the specification solution is similar to what we did for the .NET framework. The author was able to open and build this solution and its projects without issue.

Each project in this solution builds a different version of the data model or infrastructure DLL files referenced by the Service Consumer and Service Provider.

This step is important to build the specification DLL files that are stored and referenced in the SharedLibs folder.

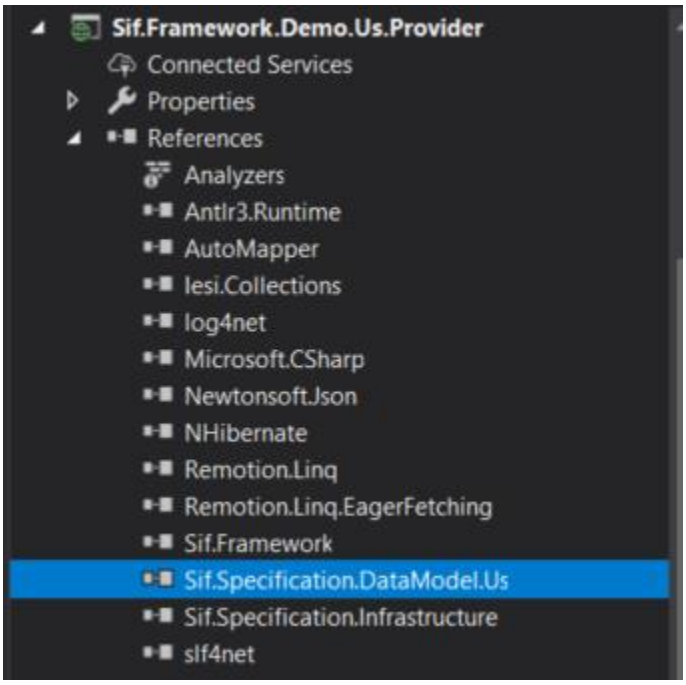
Demo Project

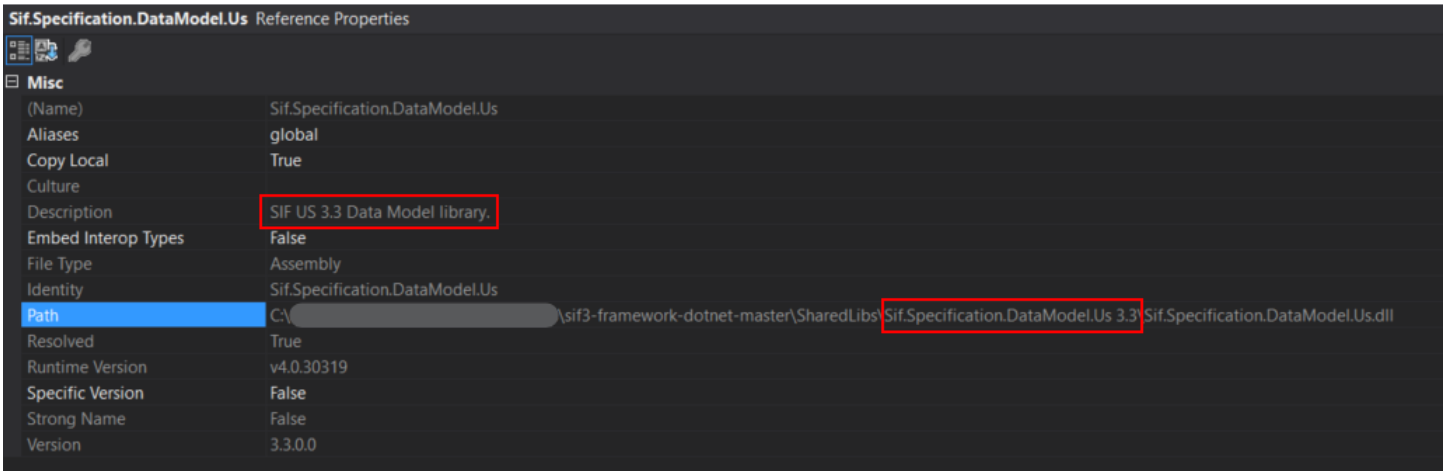
Opening the demo solution is similar to what we did for the .NET framework.

Also, similar to what we did for the Framework, check the option to "Auto-generate binding redirects" in the properties of the Sif.FrameworkDemo.Hits.Consumer, Sif.FrameworkDemo.Uk.Consumer and Sif.FrameworkDemo.Us.Consumer projects.

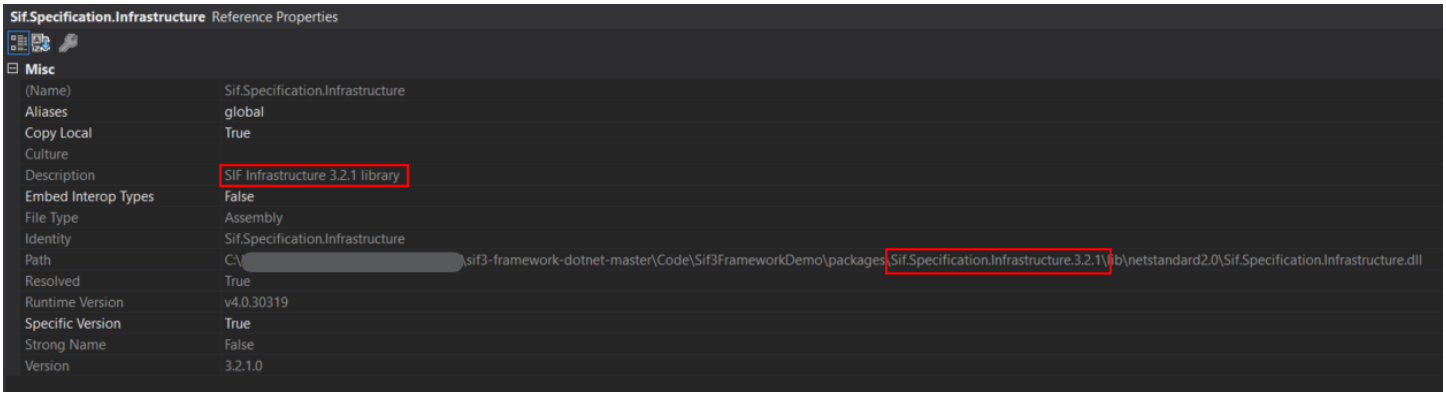
This is a good time to check the version of the SIF Data Model being used. Right click on the Sif.Specification.DataModel.Us in the References section of the Sif.Framework.Demo.Us.Provider project

and view its Properties. The path of that reference will indicate which version of the data model DLL is being used in the SharedLibs folder.



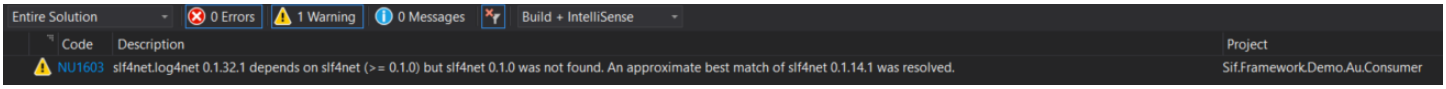


Similarly check the SIF Infrastructure reference.



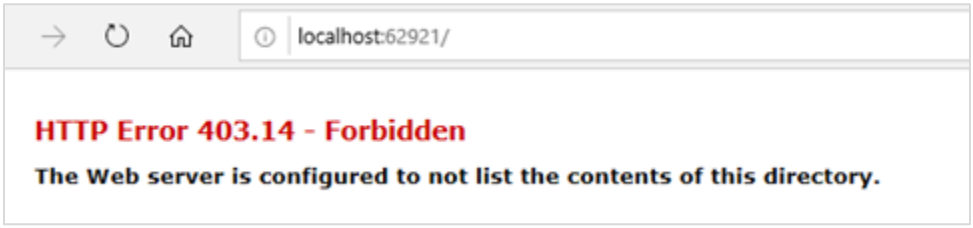
Do the same for the Sif.Framework.Demo.Us.Consumer project.

Build the Demo project: Below is the only error generated and I don't think it is critical so I'm ignoring it for now.

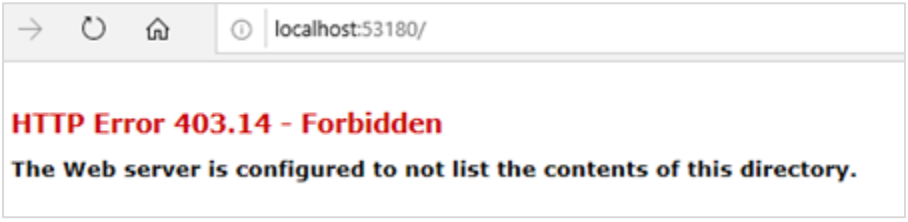


Build the Environment: Service Providers and Service Consumers need an environment to interact. Creation of an initial Environment for this demo is performed by running the *Scripts\BAT\Demo execution\DemoUsSetup.bat* script.

Start the Environment Provider: In the Sif3Framework Solution in Visual Studio, ensure that Sif.Framework.EnvironmentProvider is set as the StartUp project and run it by clicking on the Internet Explorer/IIS Express button. Ignore the "HTTP Error 403.14 – Forbidden" message that appears



Start the demo Student Provider: In the Sif3FrameworkDemo Solution in Visual Studio, ensure that Sif.Framework.Demo.Us.Provider is set as the StartUp project and run it by clicking on the Internet Explorer/IIS Express button. ignore the "HTTP Error 403.14 – Forbidden" message that appears



Run the demo Student Consumer: Do this by running the *Scripts\BAT\Demo execution\DemoUsConsumer.bat* script. The Service Consumer should connect to the Service Provider and output results of CRUD operations to a console window.

That's it! The Service Consumer and Service Provider successfully communicated (directly) through the Environment Provider. The Service Provider and Environment Provider projects can be stopped now.

Further Reading

The current SIF 3 Developers Guide (.NET) can be found in: https://github.com/Access4Learning/sif3-framework-dotnet

Next Steps

Once you have a Unity Adaptor, how do people find you?

Access 4 Learning membership is a community of both marketplace providers and end user organizations. Both benefit from knowing about the companies that support Unity and provide compliant software.

All SIF Certified products are listed on the SIF Certification Registry[13].

[13] *SIF Certification Registry: http://data.a4l.org/sif-certification-registry/*