Access 4 Learning Community

Powered by: SIF

Simple. Secure. Scalable. Standard.

SIF Standards: Technical Handbook

www.A4L.org

Version 1.2, June 2019

Contents

Introduction

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[1] and featuring the North American xPress Roster objects[2].

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 professions are building APIs enabling access to the data and impacting learning through real world products, services, integrations, and solutions.

It is best to use the xPress Roster objects for these examples because they are the simplest, most well understood, with the most production use in JSON formats. They also address the most common use case developers face.

Why use roster as an example?

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.

[1] *SIF Infrastructure Implementation Specification: https://www.a4l.org/page/Infrastructure3-3*

[2] *xPress objects can be found in the Unity Specification (also known as the SIF Data Model Implementation Specification (North America) 4.0): http://specification.sifassociation.org/Implementation/NA/4.0/*

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/

Component 1: Authentication

To access our xPress Roster objects we will use OAuth 2.0 for ease of use and the ability to integrate with many user-provisioning systems. While the necessary technical interactions are demonstrated here, the A4L Community highly recommends using Client and/or Server packages based on your development environment to simplify your use of OAuth 2.0.

Getting a Token

In order to ensure Interoperability and fit with the user stories we seek to support, xPress Roster solutions **must** support the creation of tokens through Password Grant. Tokens allow applications to submit credentials for synchronizing data. Tokens require a responsible user's permission, which gives them the ability to give the indicated user one or more shortcuts to the requested data.

Tokens are similar to barcodes. When you look at a barcode, it's not understandable, but when you have the right access and format, you can read it. A token requires two things: a key and a format. The key is what is passed first when you authenticate. This authentication is passed in HTTP calls using the sample formats below and a call to the REST interface.

Example Submission of Credentials

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=password
&username=jsmith
&password=awesome
&client_id=s6BhdRkqt3
&client_secret=7Fjfp0ZBr1KtDRbnfVdmIw
```

The results of that call are similarly returned for processing as HTTP strings.

Below is an example of the JSON string returned. You should use OAuth libraries as you need for the systems you are building.

Example Return

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
     "access_token":"2YotnFZFEjr1zCsicMWpAA",
     "token_type":"bearer",
     "expires_in":3600,
     "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
     "example_parameter":"example_value"
}
```

Using the Token

When authentication is successful, and you have obtained your OAuth 2.0 token you **may** use it with the SIF 3 Infrastructure in either of the standard ways: Bearer token in the HTTP Authorization header or access_token Query Parameter in your request. When servicing a request, it does not matter how your software received the access token, it is up to the software handling the request to verify the tokens validity and access level.

Example Header

```
Authorization : bearer 2YotnFZFEjr1zCsicMWpAA
```

Example Query Parameter

```
?access_token=2YotnFZFEjr1zCsicMWpAA
```

Other Information

Along with the OAuth response there may be one or more UUIDs returned that relate to the account being used. These take the form of <object name> + RefId and can be used to greatly reduce the number of interactions required to make requests. While this may be useful in some data synchronization scenarios we will take a deeper look at this in our mobile application user story.

Example Snippet from Response

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

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 utilizes 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 SIF Infrastructure Implementation Specification 3.3.

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 leverage 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 Consumer** 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 Kingdome 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 SIF Infrastructure Implementation Specification 3.3.

The Root URL

Example GET

URL:

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

Headers:

```
Accept: */*
Accept-Encoding: gzip
Authorization: bearer ZTk3MDcxZmItYmFiOS00ODYwLThiOTctNjM2OWZhYjk1Y2IxOlBhc3N3b3JkMQ
```

Parameters

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 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 it 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.
Timestamp	MC	M	M	HQ	Date / Time of Event creation (in ISO-8601 format also used as the basis of xs:dateTime) If not need 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
providerId: NERIC01
navigationLastPage: 1
timestamp: 2015-10-27T18:51:33Z
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. Maybe 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 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 Consumers. It is optional for immediate Responses.
requestId	MC	MC		H	Only required for delayed Requests. A Consumer specified "token" that uniquely identifies every delayed Request issued by the Consumer. It could be as simple as a monotonically increasing integer. 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
Timestamp	MC	M	M	HQ	Date / Time of Event creation (in ISO-8601 format also used as the basis of xs:dateTime) If not need 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 action requested by the Consumer (or in the case of publishing an Event, the Provider) contained in the 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 Consumer Request, as well as every published SIF Provider Event.
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..../student/{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 an unsupported query or an unsupported order clause
401	Unauthorized	Illegal Consumer Authorization token accompanying the request
403	Forbidden	Consumer Authorization token is legal, but 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 Consumer Requests requiring an immediate Response. This error indicates that the expected Service processing time for the Request is great enough that the 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: 2015-10-23T15:40:09Z
relativeServicePath: /xStudents.json
providerId: NERIC01
```

```
{
     "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 wonders around her classroom.

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

Minimums

In order to ensure all xPress Roster 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*.

xPress Roster – Data Guidance

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: bearer
ZTk3MDcxZmItYmFiOS00ODYwLThiOTctNjM2OWZhYjk1Y2IxOlBhc3N3b3JkMQ
```

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: 2015-11-02T17:12:41Z
relativeServicePath: /xCourses.json
providerId: NERIC01
```

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: bearer
ZTk3MDcxZmItYmFiOS00ODYwLThiOTctNjM2OWZhYjk1Y2IxOlBhc3N3b3JkMQ
```

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: 2015-11-02T17:15:45Z
relativeServicePath: /xCourses/A2258F48-7B8C-4406-A00E-462DCDCD3CE3/xRosters.json
providerId: NERIC01
```

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 a xPress Roster app that gives Jill the class list she needs, right on her tablet.

Behind the scenes, Jill's app is surprisingly similar to Jack's software. After all, both are collecting rosters of the people they serve. However Teacher Jill is an already authenticated part of the school's own system as is the app, so both Jill and her app get to take a shortcut. When Jill logs in, the OAuth response includes her refId.

Example Snippet from Response:

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

This allows her app 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: bearer
ZTk3MDcxZmItYmFiOS00ODYwLThiOTctNjM2OWZhYjk1Y2IxOlBhc3N3b3JkMQ
```

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 includes 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 that can be realized above and beyond this particular API, all of which can be found in the SIF Infrastructure Implementation Specification 3.3

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.