

REST Documentation Client

Last Modified on 08/01/2024 11:14 am CDT

[Key Concepts](#) | [OneRoster Walkthrough](#)

This article applies to connections using OAuth 1.0, which is no longer supported by 1EdTech. Connections using OAuth 1.0 cannot be enabled.

The REST Doc Client provides consumers of the API the means of discovering available resources and implement methods. The client allows users to verify and test endpoints exposed through the API.

The Campus OneRoster API allows districts to integrate 3rd party applications with Campus by retrieving information from Campus such as roster, student, and teacher data. OneRoster is a set of specifications established by the [IMS Global Learning Consortium](#). The REST Documentation Client allows users to test and validate this API by issuing HTTP requests and reviewing the responses.

The Rest Documentation Client can be used to modify data through PUT and DELETE methods. Campus recommends testing OneRoster API calls using those methods via a district's sandbox environment and not in production.

Key Concepts

- **Resources** are objects with associated data and a set of methods to operate on it.
- **Endpoints** are unique URLs that represents an object or collection of objects. The endpoint is used by the HTTP client to interact with data resources.
- **Models** represent data objects. Aside from the model name, the model contains a description, along with properties that define the contained elements and data types.
- **Path** includes the URL for accessing the resource endpoints.
- **Method** is an operation or action that can be performed on a resource.

The REST Doc Client handles URL encoding by replacing unsafe ASCII characters with a "%" followed by two hexadecimal digits. For more information on URL encoding visit http://www.w3schools.com/tags/ref_urlencode.asp

OneRoster Walkthrough

1. Log Into the REST Doc Client
 1. The REST Doc client exposes sensitive SIS data and therefore requires authentication.
 2. The user is granted full access to all resources based on the available actions.
 3. Granular calendar or tool rights are not supported.
 4. This account is not linked with any Campus SIS accounts.
 5. The credentials are based on the Consumer Key and Secret configured on that specific district edition.

2. Browse or Search for Resources
 1. Before any requests can be submitted, the user must first identify what resource they would like to access.
 2. The resources are based on OneRoster object naming conventions

3. Review the associated model
 1. Access the resource models to familiarize yourself with the mapped data
 2. Close the model when you are done reviewing it.

Resources

Show Advanced Filter Options

Filter by name/path/description X

AcademicSessionSubResource
(/v1/academicSessions)

Provides read access to academic sessions

Models

Enter one or more keywords to filter

Path

/v1/classes

/v1/classes/{classSourcedId}

Models

Primary Model

ClassModel

Description: Represents a class. Back to previous

Properties:

```

_baseCode (String)
_id (String)
_model (String)
classCode (String)
classType (String)
course (CourseHypermediaModel)
dateLastModified (CampusDateTime)
location (String)
school (SchoolHypermediaModel)
sourcedId (String)
status (String)
subjects (List<String>)
terms (List<TermHypermediaModel>)
title (String)

```

*property is required

Related Models

4. Select a request method (action)

- Determine which method you would like to perform on the resource based on the provided description

Models	Endpoints	
Enter one or more keywords to filter endpoints		
Path	Method	Description
/v1/classes	GET	Retrieves a list of all classes
/v1/classes/{classSourcedId}	GET	Retrieves a class by sourcedId

- Some paths require additional parameters be provided in order to issue a request. (In the example below I would need to enter a Section.sectionID)

Query Form: GET

Retrieves a class by sourcedId

Main Parameters

Parameter	Value	Parameter Type	Data Type
classSourcedId	{required}	path	GetService

- PUT Methods require input content. (In the example below I am posting a Result (GradingScore record) with the required date, score (percent) and custom letter grade (score) fields:

Main Parameters

Parameter	Value	Parameter Type	Data Type
lineitemSourcedId	246639_793_36	path	String
studentSourcedId	s28545	path	String
sourcedId	31154CF8-CA14-4B81-8B2	path	UpdateService

```

{
  "result": {
    "date": "2016-05-12T21:49:00.000Z",
    "score": "90.0",
    "metadata": {
      "ext_infiniteCampus_scoreGrade": "A"
    }
  }
}

```

Input Content Type:

- The REST Doc client supports pagination
 - Set the following values to override the defaults: Limit = Number of Records per Page, Offset = Starting Record

Query Form: GET

Retrieves a list of all enrollments

Filters Remove All Filters

Order Priority	Field	Filter	Value	Options
No filters added				

To add a new filter, select a filter type from the dropdown list

<table style="width: 100%; border: 1px solid #ccc;"> <tr> <td style="width: 100px; border: 1px solid #ccc;">Filter</td> <td style="border: 1px solid #ccc;"> <input type="text"/> </td> <td style="width: 20px; border: 1px solid #ccc; text-align: center;">+</td> </tr> </table>	Filter	<input type="text"/>	+
Filter	<input type="text"/>	+	

Page Limit: <input style="width: 40px;" type="text" value="1000"/>	Page Offset: <input style="width: 40px;" type="text" value="1000"/>
--	---

Response Content Type:

- Additional **custom** filters can be added for GET Methods

- The filter needs to be added to the request, otherwise the value will be ignored.
 - The filter syntax always starts with the string prefix of "filter="+fieldName+filterType+" "+value+" "
 - Values are enclosed in single quotes, regardless of dataType (i.e. not just for text but for IDs and Booleans too)
 - Note: Field names are case sensitive

2. Below is an example of the syntax for filtering based off a single field value.

Here is an example of a contains filter:
`filter=usernamecontains'Bob.Marley'`

3. Once added the filters will appear above, and can be edited or removed before submitting the request.

Order	Priority	Field	Filter	Value	Options
1			custom	filter=title='530 21 science 5'	

[Remove All Filters](#)

5. Submit the request in order to retrieve a response

6. Review the response

1. The response is formatted as JSON, which is similar to XML.
 1. It displays data organized by a hierarchy of tags; starting with the object, embedded elements and values, and followed by any linked objects and their values.
2. The status code reveals whether or not the request was successfully satisfied.
 1. <http://www.imglobal.org/lis/imsonerosterv1p0/imsOneRoster-v1p0.html#toc-33>
 2. [Click here to expand...](#)

Error Code	Description
200	OK - It was possible to read the collection / resource
201	OK - New resource has been created
204	OK - The resource was deleted successfully
400	Bad Request - the Request was invalid and cannot be served.
401	Unauthorized - the Request requires authorization
404	Not Found - there is no resource behind the URI.
422	Entity cannot be processed - used where the server cannot validate an incoming entity.
500	Internal Server Error

Query Response
Back To Top

Request URL

<https://oneroster2.infinitecampus.org/oneroster/learningdata/v1/classes/247936>

Response

```

{
  "class": {
    "sourcedId": "247936",
    "status": "active",
    "dateLastModified": "2016-05-24T21:27:08.680Z",
    "title": "530 21 Science 5",
    "classType": "scheduled",
    "classCode": "",
    "course": {
      "sourcedId": "22804",
      "href": "https://oneroster2.infinitecampus.org/oneroster/learni",
      "type": "course"
    },
    "school": {

```

Status

200