



API Guidance

Pinnacle Series Authorization and Reporting APIs

Prepared by
Eagle Point Development Team
Last Revised: July 28, 2022

© 2022 Eagle Point Software.
All rights reserved



Eagle Point
Software

Contents

- ProdGen REST API QuickStart 4
 - Introduction 4
- Authorization Summary: 7
- Authorization Step 1: Get apiToken (apiKey) 8
 - Sample call to get the apiToken(apiKey): 8
 - Sample response from the apiToken call: 8
 - javaScript Example: 8
- Authorization Step 2: Get userOrgInfo 10
 - Sample call to get the userOrgInfo: 10
 - javaScript Example: 11
- Authorization Step 3: Post userToken (userAccessKey) 12
 - Sample call to get the userToken (userAccessKey): 12
 - Sample response from the userToken call: 13
 - javaScript Example: 14
- Reports – All User Progress 15
 - Sample call to get the allUserProgress: 15
 - Sample response from the allUserProgress API call: 15
 - javaScript Example: 16
- Reports – Completed Courses 17
 - Sample call to get the completedCourses API: 17
 - Sample response from the completedCourses API call: 18
 - javaScript Example: 19
- Reports – Current Enrollments 20
 - Sample call to get the currentEnrollments: 20
 - Sample response from the currentEnrollments API call: 20
 - javaScript Example: 22
- Reports – Learning Path Progress 23
 - Sample call to get the LPPProgress : 23
 - Sample response from the LPPProgress API call: 24
 - javaScript Example: 25
- Reports – Top Searches 26
 - Sample call to get the topSearches : 26
 - Sample response from the topSearches API call: 27

javaScript Example:.....	28
Reports – Unique User Sign-Ins.....	29
Sample call to get the uniqueUsers:	29
Sample response from the uniqueUsers API call:	30
javaScript Example:.....	31

ProdGen REST API QuickStart

Introduction

Pinnacle Series 2018 is built on top of a broad content platform called Productivity Generator. Pinnacle Series combines the platform features of Productivity Generator with pre-loaded Architectural and Engineering content and support services. Our APIs are documented in a tool called Swagger, and you can view more information about what APIs are available and how to call them.

To understand more about what an API is and learn more about API basic concepts, please see the ['Pinnacle Series API Overview'](#) knowledge base article

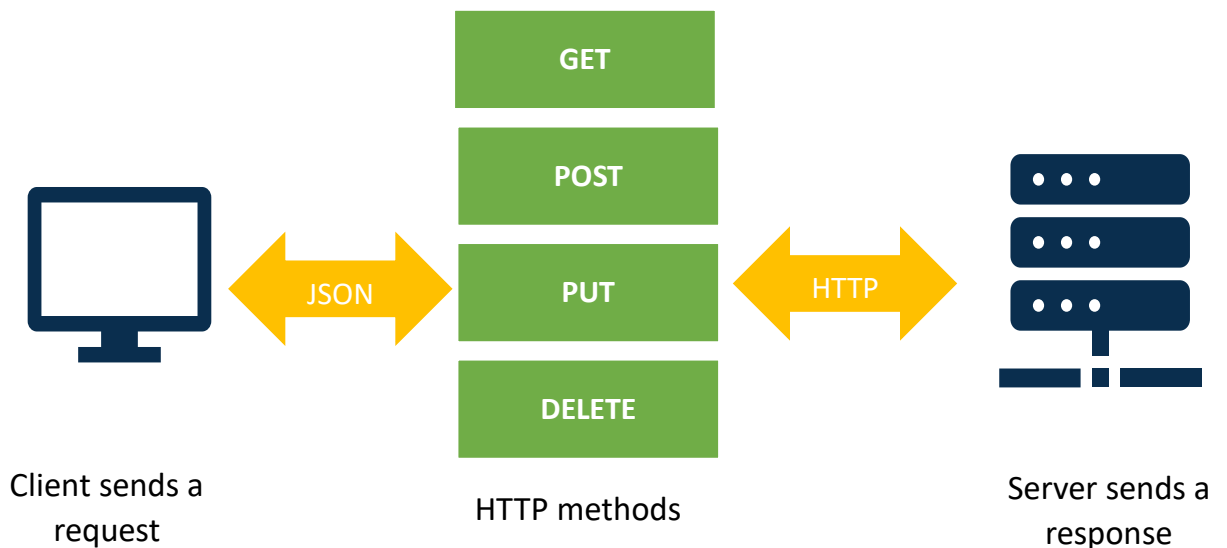
The information in this document will focus on the v1 API's for how to get authorized to use the APIs in order to call other endpoints like the "Reports – Completed Courses" API. We provide examples in this document of calling the API's using JavaScript, but you could also use applications like Postman for testing. The output from the APIs in the JavaScript examples is writing to the console of the browser. But you could also pull the data and reformat it into an output file to be pulled into an application like Excel.

The ProdGen REST API provides developers with a mechanism to access, create and update data stored in Productivity Generator platform. This includes, but is not limited to, content such as Workflows, Cheat Sheets, and Videos. Also, Learning Paths/Courses, enrollments, etc.

As mentioned, the API is built using REST architecture methodologies. Methods are accessed by specifying unique URLs along with an action type (GET, POST, PUT, DELETE, etc.). If you are not familiar with RESTful APIs, see the following for further explanation:

https://en.wikipedia.org/wiki/Representational_state_transfer

<https://www.restapitutorial.com/>



A complete look at the current APIs available at the following site:

We strongly advise using the BETA site below first to identify possible programming bugs which may cause a looping process that can have negative ramifications for production users. Once you are confident that everything is working as expected in BETA, you could switch over to production to verify that you are pulling the data as expected.

To pull reporting data from Pinnacle Series, please use V1 at this time.

Beta Testing

<https://prodgenapibeta.azurewebsites.net/swagger/ui/index>

Production Testing

<https://prodgenapi.azurewebsites.net/swagger/ui/index>

The above URL will default to V2 of our API and you will need to change to V1 by editing the address in the Swagger navigation bar.

v1:



A sample of how this site displays our API information is listed in the screenshots below. Clicking on the name of the API allows you to drill into more detail about it.

ProdGenAPI			
AssetLibrary	Show/Hide	List Operations	Expand Operations
Authorization	Show/Hide	List Operations	Expand Operations
Category	Show/Hide	List Operations	Expand Operations
Chat	Show/Hide	List Operations	Expand Operations
Content	Show/Hide	List Operations	Expand Operations
Enrollments	Show/Hide	List Operations	Expand Operations

This image shows a sampling of the main API controller area for Enrollments. Each API offers a summary of what the API is used for.

Enrollments

Show/Hide | List Operations | Expand Operations

GET	/api/enrollments	Retrieves all enrollments for the current user
DELETE	/api/enrollments/{enrollmentId}	Remove the enrollment
GET	/api/enrollments/{enrollmentId}	Retrieves the enrollment
PUT	/api/enrollments/{enrollmentId}	Complete the enrollment
PUT	/api/enrollments/{enrollmentId}/dueDate	Complete the enrollment
GET	/api/enrollments/{enrollmentId}/certificate	Retrieves the completion certificate for the enrollment

Likewise, clicking on an API will expand it to show a more detailed description of the operation's intent, parameters, and responses. See the following screen example of the enrollments API expanded to show more detail.

GET /api/enrollments/{enrollmentId}
Retrieves the enrollm

Implementation Notes

Use this method to return a specific enrollment. You must supply the enrollmentId parameter.

Response Class (Status 200)

OK

Model | Example Value

```
{
  "enrollmentId": "00000000-0000-0000-0000-000000000000",
  "courseId": "00000000-0000-0000-0000-000000000000",
  "imageUrl": "string",
  "courseName": "string",
  "learningPathName": "string",
  "userId": "00000000-0000-0000-0000-000000000000",
  "percentComplete": 0,
  "enrollmentStatus": "assigned",
  "statusDate": "2018-05-01T14:47:26.571Z",
  "hasDueDate": true
}
```

Response Content Type

Parameters





Parameter	Value	Description	Parameter Type	Data Type
enrollmentId	<input type="text" value="(required)"/>	The unique identifier of the enrollment item.	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	BadRequest		
401	Unauthorized		

Authorization Summary:

To call any of the API's, you will first need to get authorization to verify that you have access to call the API's. This can be done in 2 or 3 steps, as outlined below.

-  To use the API's, you will first need to get your organization's ApplicationID which gives you access to utilize the API's.
-  **You can obtain your ApplicationID by contacting Eagle Point Software support.**
-  This ID will be unique to you or your organization and allows you to access the API.
-  This ApplicationID should be passed in as a parameter to the apiToken (apiKey) operation to retrieve a session-based API token. That API token (as mentioned above) will be passed into each subsequent call to the API.

To call many of the operations included in the API, you need to supply two essential things in the HTTP headers. The first is an API Token, and the second is a User Token. In general terms, the API token is what allows you access to make the calls to the API, and the User Token is used to determine which operations and data you can operate with based on your specific user permissions.

Once a valid API token is received, the next step is to retrieve a list of tenants (platform databases) that a user would have access to login to. A single user can be listed in multiple tenants for an organization, even though typically an organization will only be associated with a single tenant. To retrieve the list of available tenants, the following operation should be called, which will return a list of tenants and, more importantly, the unique tenant ID associated to that tenant.

Note: If your organization only has a single tenant, you can contact Eagle Point to retrieve your tenant id directly and bypass the necessity of this API call.

Both the **userToken** and **apiToken** should be stored in memory for the session. The user token must be added to the HTTP header of each call to the API with a header name of "**userAccessKey**" and the **apiToken** must also be added as a header with the name of '**apiKey**'. See the samples below for more information.

General Steps to authorize for v1 API's:

1. Do an API call to: **apiToken**. This will retrieve an **apiToken (apiKey)** for the session.
2. Do an API call to: **userOrgInfo**. This will retrieve tenant and organization information for the user.
3. Do an API call to: **userToken**. This will retrieve a user access token for the session that is needed in future API calls.

Authorization Step 1: Get apiToken (apiKey)

To call many of the operations included in the API, you need to supply two essential things in the HTTP headers. The first is an API Token, and the second is a User Token. In general terms, the API token is the thing that allows you access to make the calls to the API, and the User Token is used to determine which operations and data you can operate with based on your specific user permissions.

Sample call to get the apiToken(apiKey):

GET /api/authorization/apiToken Retrieve an API Token for this session

Implementation Notes
This method will retrieve an API token that will be used to access the ProdGen API for the current session. This token must be passed into all future API calls in the HTTP header with a key value of 'apiKey'.
The applicationId parameter is a unique string supplied to you by Eagle Point in order to request access to the API. An applicationId specifically ties your organization to it's ability to call the ProdGen API.

Response Class (Status 200)
string
Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
applicationId	(required)	Unique ID supplied by Eagle Point for API access.	query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	BadRequest		
403	Forbidden		

[Try it out!](#)

Sample response from the apiToken call:

The value in the ResponseBody is the apiToken (apiKey).

Response Body

```
"irr+og5hy2PEvH8M0ejEX3t0JzD59CI9DJ2j50vDezCwqMOBS5XJ8SnAyPp+wMb0iDtC1XYuZu0E/utgGQoTgsraV4PyQuHRiWgq2QNMFjrjeeUcz+hqL3Fg5JorihA"
```

Response Code

```
200
```


Javascript Example

Some global variables are used in all the JavaScript examples in this document. We have also included a simple HTML document with JavaScript for testing. This HTML has buttons on the page that you can call the `apiToken`, `userOrgInfo`, `userToken` and the report API's. To test with this application, you will need to edit the .html document to add your `applicationId`, `emailId`, `passwordText`, and `tenantId`. The other values for `ApiKey` and `UserBearerToken` will get populated by some of the authorization API calls.

```
let applicationId = 'NEED VALUE HERE'; //This is your application ID as assigned by Eagle Point Software.
let ApiKey = ''; //This value is set in the GetApiToken function.
let emailId = "test_email@company.com"; //Your email associated with the tenant
let passwordText = "PASSWORD"; //Your password associated with the email and tenant
let myTenantId = 'TENANT VALUE'; //Hardcoded if you only have 1 tenant, or get value from userOrgInfo
let myLanguage = 'en'; //This value is the default for language - english
let myTimeZone = 'CST'; //This value is the default for timezone
let UserBearerToken = ''; //This value is set in the GetUserAccessToken endpoint

//BETA URL:
let m_BaseURL = "https://prodgenapibeta.azurewebsites.net/api/";
//PRODUCTION URL
//let m_BaseURL = "https://prodgenapi.azurewebsites.net/api/";
```

```
function GetApiToken() {
//This function call is the first one needed in the v1 authorization. It will take the applicationId provided to you from Eagle
//Point Software which gives you access to the endpoints.
var xhr = new XMLHttpRequest();
var localURL = m_BaseURL + "authorization/apiToken/?applicationId=" + applicationId;
xhr.open("GET", localURL, true);
xhr.onload = function (e) {
    if (xhr.readyState === 4) {
        if (xhr.status === 200) {
            ApiKey = xhr.responseText;
        } else {
            console.log("*Error Found");
            console.error(xhr.statusText);
        }
    }
};
xhr.onerror = function (e) {
    console.error(xhr.statusText);
    console.error(e);
};
xhr.send("");
}
```

Authorization Step 2: Get userOrgInfo

This API call will get you more information about tenants associated with your ApplicationID. The call requires an email address and the ApiKey (apiToken) that was generated from the apiToken call.

Sample call to get the userOrgInfo:

GET /api/authorization/userOrgInfo Retrieve general information and valid tenants for a given user

Implementation Notes
This method will retrieve some general information about a user based on their email address. Of the information returned, the most important is the list of valid tenants (databases) the user has permission to log into.
The tenantId of one of these tenants must be specified in order to receive a valid userToken.

Response Class (Status 200)
OK

Model | **Example Value**

```
{
  "organizationId": 0,
  "partnerId": "00000000-0000-0000-0000-000000000000",
  "partnerName": "string",
  "loginImageUrl": "string",
  "productName": "string",
  "supportEmail": "string",
  "tenantList": [
    {
      "tenantId": "00000000-0000-0000-0000-000000000000",
    }
  ]
}
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
email	(required)	User's email address	query	string
ApiKey	(required)	Your API token. To get an API token, do a GET call to /api/authorization/apiToken with your applicationId.	header	string

Sample response from the userOrgInfo call:

The response from this API will be the organization and tenant information. The tenantId is needed for some future API calls.

```
{
  "organizationId": 0,
  "partnerId": "00000000-0000-0000-0000-000000000000",
  "partnerName": "string",
  "loginImageUrl": "string",
  "productName": "string",
  "supportEmail": "string",
  "tenantList": [
    {
      "tenantId": "00000000-0000-0000-0000-000000000000",
    }
  ]
}
```

JavaScript Example:

```
function GetUserOrgInfo() {
//This function call is only needed if your organization has multiple tenants.
//If you only have 1 tenant, this call would only be needed 1 time to get the tenant ID
//or Eagle Point can provide you with your tenant ID.
    var xhr = new XMLHttpRequest();
    var localURL = m_BaseURL + "authorization/userOrgInfo/?email=" + emailId;
    xhr.open("GET", localURL, true);
    xhr.setRequestHeader("Content-Type", "application/json");
    xhr.setRequestHeader("apiKey", ApiKey);
    xhr.onload = function (e) {
        if (xhr.readyState === 4) {
            if (xhr.status === 200) {
                let orgs = JSON.parse(xhr.responseText);
                var findValue = 'organizationId';
                for(var k in orgs) {
                    if (k.match(/organizationId/)) {
                        console.log("** Main organizationId was found: " + orgs[k]);

                        if (k.match(/tenantList/)) {
                            for(var k1_1 in orgs[k]) {
                                for(var k1_2 in orgs[k][k1_1]) {
                                    if (k1_2.match(/tenantId|name/)) {
                                        console.log("      * " + k1_2 + " value is: " + orgs[k][k1_1][k1_2]);
                                        //Display the tenant ID and name to the debug console
                                    }
                                }
                            }
                        }
                    }
                }
            }
            else {
                console.error(xhr.statusText);
            }
        }
    };
    xhr.onerror = function (e) {
        console.error(xhr.statusText);
        console.error(e);
    };
    xhr.send("");
}
```

Authorization Step 3: Post userToken (userAccessKey)

As noted in the description of this operation, a valid tenantId from the tenantList must be specified to get a userToken (userAccessKey). Retrieving a valid userToken (userAccessKey) is conceptually the same as logging into the platform as a user. The tenantId specifies which database the user is logging into. With a valid tenantId, the following operation should now be called to retrieve the session-based userToken (userAccessKey):

Sample call to get the userToken (userAccessKey):

POST /api/authorization/userToken Retrieve a User Access Token for this session

Implementation Notes

This method will retrieve a userToken that provides access to specific content and areas of the system. The email address, password, and a valid tenantId must be supplied in order to receive a valid userToken. This is the equivalent of logging into the system as a user and is valid only for the current session.

The returned userToken must be passed into future calls to the API as an identification of the user. This must be done by setting a 'userAccessKey' key in the HTTP header to the access token returned from this call.

Response Class (Status 200)
string

Response Content Type:

Parameters

Parameter	Value	Description	Parameter Type	Data Type
authBody	(required) <input type="text"/>		body	Model <pre>{ "email": "string", "password": "string", "tenantId": "string", "language": "string", "timeoffset": 0, "timezoneName": "string", "returnLongLivedToken": true }</pre>
ApiKey	(required) <input type="text"/>	Your API token. To get an API token, do a GET call to /api/authorization/apiToken with your applicationId.	header	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
403	Forbidden		

Sample response from the userToken call:



The screenshot displays a REST client interface with two sections: "Response Body" and "Response Code".

Response Body: A JSON object containing the following fields:

- "userAccessKey": "xIquAG8Z9q/n+RPsZSv/QgNAYbYX5ZuA5TnMMZQ29KTJJrRrjRCjZTGxJopkXtSTlcjYshZ+p9KqDqYvJfwa1jAcZ3c1Lfw016zXvVgK8Mb"
- "apiV2AccessKey": {
 - "bearerToken": null,
 - "expiration": null,
 - "orgID": 0,
 - "tenantid": "00000000-0000-0000-0000-000000000000",
 - "userId": "00000000-0000-0000-0000-000000000000"}
- "longLivedToken": "SONIF7DCKxbqXkQdi6RzgkuwiI5FGsjP31IuR6sp0dX8LLB01m82FIbxzFZzCEQdJ4htxpTTMpJW4/ZcnPe3V1vuwMDcRoc4zU3VIhH8hi"
- "expirationDate": "2023-02-25T12:06:58.2502295-06:00",
- "isExternalUser": false,
- "email": [REDACTED]
- "tenantid": [REDACTED]

Response Code: 200

JavaScript Example:

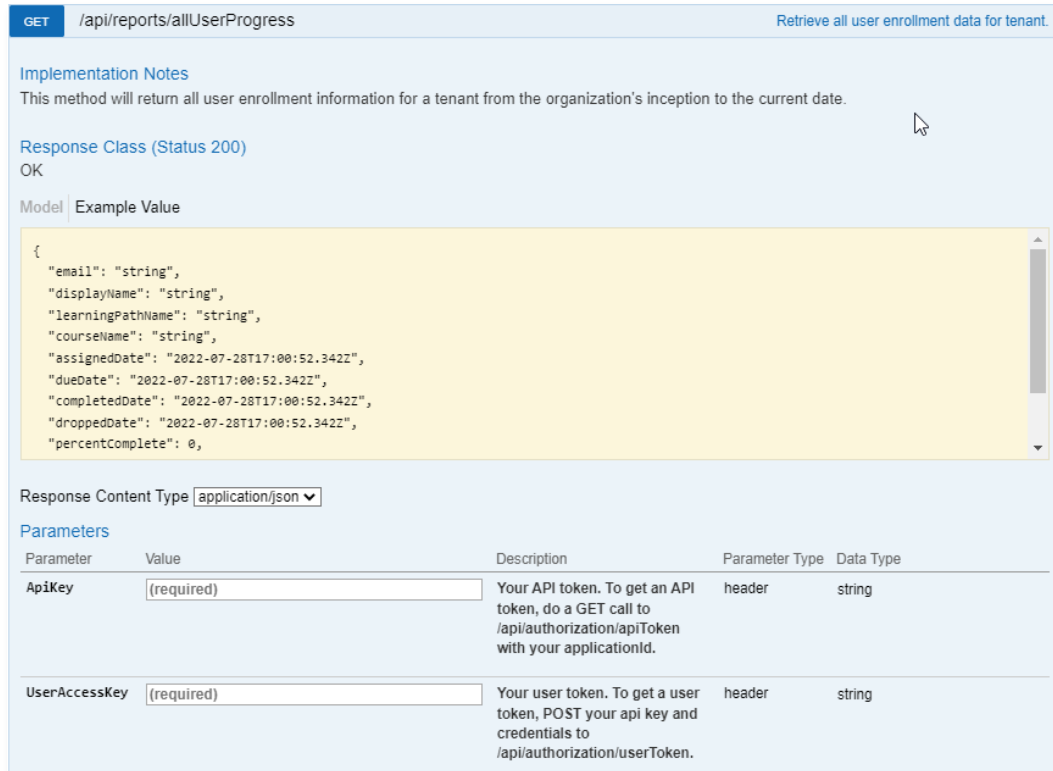
```
function GetUserAccessKey() {
    //This function call is last one needed in the authorization process. It will take the ApiKey from the call to ApiToken along
    with the authBody which consists
    //of the email, password, tenantId, language, timeOffset, timezoneName and returnLongLivedToken.
    //Here is an example of the body object:
    // "email": "test_email@companyname.com",
    // "password": "Pinnacle1",
    // "tenantId": "e4428eee-ac8c-4b94-9042-6ce6a8b37e0c",           //This is your tenantId from the userOrgInfo endpoint
    // "language": "en",                                           //This indicate "English"
    // "timeOffset": 0,
    // "timezoneName": "CST",                                       //This indicates "Central Standard Time"
    // "returnLongLivedToken": true
    var xhr = new XMLHttpRequest();
    var localURL = m_BaseURL + "authorization/userToken/";
    xhr.open("POST", localURL, true);
    xhr.setRequestHeader("Content-Type", "application/json");
    xhr.setRequestHeader("apiKey", ApiKey);
    xhr.onload = function (e) {
        if (xhr.readyState === 4) {
            if (xhr.status === 200) {
                let orgObj = JSON.parse(xhr.responseText);
                for(var k in orgObj) {
                    if (k.match(/userAccessKey/)) {
                        UserBearerToken = orgObj[k];
                    }
                }
            } else {
                console.error(xhr.statusText);
            }
        }
    };
    xhr.onerror = function (e) {
        console.error(xhr.statusText);
        console.error(e);
    };
    var body = "{ \"email\": \"" + emailId + "\", \"password\": \"" + passwordText + "\", \"tenantId\": \"" +
    myTenantId + "\", \"language\": \"" + myLanguage + "\", \"timeOffset\": \"0\", \"timezoneName\": \"" + myTimeZone +
    "\", \"returnLongLivedToken\": \"true\"}";
    xhr.send(body);
}
```

Reports – All User Progress

This API will return a listing of all user enrollment data for a tenant from the organization's inception to the current date.

A call to the allUserProgress API will return the list of enrollment data for your organization. This API requires the ApiKey from the apiToken API and the UserAccessKey from the userToken API.

Sample call to get the allUserProgress:



The screenshot shows the API documentation for the endpoint `GET /api/reports/allUserProgress`. The page includes implementation notes, response class information, and a table of parameters.

Implementation Notes
This method will return all user enrollment information for a tenant from the organization's inception to the current date.

Response Class (Status 200)
OK

Model Example Value

```
{
  "email": "string",
  "displayName": "string",
  "learningPathName": "string",
  "courseName": "string",
  "assignedDate": "2022-07-28T17:00:52.342Z",
  "dueDate": "2022-07-28T17:00:52.342Z",
  "completedDate": "2022-07-28T17:00:52.342Z",
  "droppedDate": "2022-07-28T17:00:52.342Z",
  "percentComplete": 0,
}
```

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
ApiKey	<input type="text" value="(required)"/>	Your API token. To get an API token, do a GET call to /api/authorization/apiToken with your applicationId.	header	string
UserAccessKey	<input type="text" value="(required)"/>	Your user token. To get a user token, POST your api key and credentials to /api/authorization/userToken.	header	string

Sample response from the allUserProgress API call:



The screenshot shows the 'Response Class (Status 200)' section of the API documentation, displaying the 'Example Value' in a JSON format.

Response Class (Status 200)
OK

Model Example Value

```
{
  "email": "string",
  "displayName": "string",
  "learningPathName": "string",
  "courseName": "string",
  "assignedDate": "2022-07-28T17:00:52.342Z",
  "dueDate": "2022-07-28T17:00:52.342Z",
  "completedDate": "2022-07-28T17:00:52.342Z",
  "droppedDate": "2022-07-28T17:00:52.342Z",
  "percentComplete": 0,
}
```

JavaScript Example:

```
function ReportAllUserProgress() {
    //Debug console.log("Start - ReportAllUserProgress");
    var searchCount = 0;
    var xhr = new XMLHttpRequest();
    var localURL = m_BaseURL + "reports/allUserProgress"
    xhr.open("GET", localURL, true);
    xhr.setRequestHeader("Content-Type", "text/json");
    xhr.setRequestHeader("Accept", "application/json");
    xhr.setRequestHeader("apiKey", ApiKey);
    xhr.setRequestHeader("userAccessKey", UserBearerToken);
    xhr.onload = function (e) {
        if (xhr.readyState === 4) {
            if (xhr.status === 200) {
                let userProgress = JSON.parse(xhr.responseText);
                for(var progress in userProgress) {
                    for(var up in userProgress[progress]) {
                        //Display results to console
                        if (up.match(/email|displayName|learningPathName|courseName|assignedDate|dueDate|completedDate/)) {
                            if (up.match(/email/)) {
                                searchCount=searchCount+1;
                                console.log( "***Search " + up + " value is: " + userProgress[progress][up]);
                            }
                            else {
                                console.log( " " + up + " value is: " + userProgress[progress][up]);
                            }
                        }
                    }
                }
                console.log("***Total Search Records returned: " + searchCount);
            } else {
                console.log("**Error Found");
                console.error(xhr.statusText);
            }
        }
    };
    xhr.onerror = function (e) {
        console.log("**Error Found");
        console.error(xhr.statusText);
        console.error(e);
    };
    xhr.send("");
}
```


Reports – Completed Courses

A call to the 'completedCourses API' will return the list of completed courses for all users in your organization.

This API requires the ApiKey from the apiToken API and the UserAccessKey from the userToken API. You can also enter date criteria with this API to narrow down the results to specific timeframes.

Sample call to get the completedCourses API:

GET /api/reports/completedCourses Retrieve a list of completed courses for all users.

Implementation Notes
This method will return a list of all completed courses for the organization based on the date parameters.

Date parameter examples:

- If you want to pull data starting with date="2022-06-01", you would enter startDate="2022-06-01".
- If you want to pull data that were completed prior to "2022-04-01", you would enter an endDate="2022-04-01".
- If you want to pull data for a date range like "2021-12-01" to "2022-03-15", you would enter a startDate="2021-12-01" and an endDate="2022-03-15".
- If you want to pull data for a specific date like "2021-06-10", you would enter a startDate="2021-06-10" and endDate="2021-06-11".

Response Class (Status 200)
OK

Model | Example Value

```
[
  {
    "userName": "string",
    "userEmail": "string",
    "userId": "00000000-0000-0000-0000-000000000000",
    "employeeID": "string",
    "learningPathName": "string",
    "courseName": "string",
    "enrollmentDueDate": "2022-07-28T14:56:41.018Z",
    "enrollmentCompletedDate": "2022-07-28T14:56:41.018Z",
  }
]
```

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
startDate	<input type="text"/>	Optional. The date to get data from in YYYY-MM-DD format.	query	string
endDate	<input type="text"/>	Optional. The date to get data to in YYYY-MM-DD format.	query	string

ApiKey	<input type="text" value="(required)"/>	Your API token. To get an API token, do a GET call to /api/authorization/apiToken with your applicationId.	header	string
UserAccessKey	<input type="text" value="(required)"/>	Your user token. To get a user token, POST your api key and credentials to /api/authorization/userToken.	header	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		

Sample response from the completedCourses API call:



The screenshot shows a web-based interface for viewing an API response. The top section is titled "Response Body" and contains a JSON object. The bottom section is titled "Response Code" and shows the status code "200".

```
[
  {
    "userName": "REDACTED",
    "userEmail": "REDACTED",
    "userId": "538c406f-REDACTED",
    "employeeID": "",
    "learningPathName": "Advance Steel Fundamentals",
    "courseName": "Working with Advance Steel",
    "enrollmentDueDate": "2022-02-15T08:11:16",
    "enrollmentCompletedDate": "2022-02-15T08:11:16",
    "enrollmentProgress": 100,
    "viewingMinutes": 0.25,
    "quizComposite": 0.75,
    "quizScoreStatus": "",
    "courseProducts": "Advance Steel 2021",
    "courseid": "c1ed66b1-3764-4a56-99ce-83c68dac09c0",
    "enrollid": "80335f52-15ac-4dee-b41a-e3f7e9c0916e",
    "username": ""
  },
  {
    ...
  }
]
```

Response Code

200

JavaScript Example:

```
function ReportCompletedCourses() {
//Debug console.log("Start - ReportCompletedCourses");
//This function call is an example of calling one of the v1 Reporting API's with the apiKey and userAccessKey.
    var searchCount = 0;
    var xhr = new XMLHttpRequest();
    //testing start/end date use -> var localURL = m_BaseURL +
"reports/completedCourses?startDate=6%2F1%2F2021&endDate=3%2F25%2F2022"
    var localURL = m_BaseURL + "reports/completedCourses"
    xhr.open("GET", localURL, true);
    xhr.setRequestHeader("Content-Type", "text/json");
    xhr.setRequestHeader("Accept", "application/json");
        xhr.setRequestHeader("apiKey", ApiKey);
    xhr.setRequestHeader("userAccessKey", UserBearerToken);

    xhr.onload = function (e) {
    if (xhr.readyState === 4) {
        if (xhr.status === 200) {
            let courses = JSON.parse(xhr.responseText);
            for(var course in courses) {
                for(var courseInfo in courses[course]) {
                    if (courseInfo.match(/userName|userEmail|learningPathName|courseName/)) {
                        if (courseInfo.match(/userName/)) {
                            searchCount=searchCount+1;
                            console.log("***Search " + courseInfo + " value is: " + courses[course][courseInfo]);
                        }
                        else {
                            console.log(" " + courseInfo + " value is: " + courses[course][courseInfo]);
                        }
                    }
                }
            }
            console.log("***Total Search Records returned: " + searchCount);
        } else {
            console.log("***Error Found");
            console.error(xhr.statusText);
        }
    }
    };
    xhr.onerror = function (e) {
        console.log("***Error Found");
        console.error(xhr.statusText);
        console.error(e);
    };
    xhr.send("");
}}

```

Reports – Current Enrollments

A call to the 'currentEnrollments API' will return the list of currently active enrollments for all users in your organization. Enter date criteria with this API to narrow down the results to specific timeframes.

This API requires the ApiKey from the apiToken API and the UserAccessKey from the userToken API.

Sample call to get the currentEnrollments:

GET /api/reports/currentEnrollments Retrieve a list of currently active enrollments for all users.

Implementation Notes
This method will return a list of all currently active enrollments for the organization based on the date parameters.

Date parameter examples:

- If you want to pull data starting with date="2022-06-01", you would enter startDate="2022-06-01".
- If you want to pull data that were completed prior to "2022-04-01", you would enter an endDate="2022-04-01".
- If you want to pull data for a date range like "2021-12-01" to "2022-03-15", you would enter a startDate="2021-12-01" and an endDate="2022-03-15".
- If you want to pull data for a specific date like "2021-06-10", you would enter a startDate="2021-06-10" and endDate="2021-06-11".

Response Class (Status 200)
OK

Model | Example Value

```
[
  {
    "learningPathName": "string",
    "courseName": "string",
    "userId": "00000000-0000-0000-0000-000000000000",
    "employeeID": "string",
    "userName": "string",
    "userEmail": "string",
    "enrollmentStatus": "string",
    "enrollmentProgress": 0,
  }
]
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
startDate	<input type="text"/>	Optional. The date to get data from in YYYY-MM-DD format.	query	string
endDate	<input type="text"/>	Optional. The date to get data to in YYYY-MM-DD format.	query	string
ApiKey	(required) <input type="text"/>	Your API token. To get an API token, do a GET call to /api/authorization/apiToken with your applicationId.	header	string
UserAccessKey	(required) <input type="text"/>	Your user token. To get a user token, POST your api key and credentials to /api/authorization/userToken.	header	string

Sample response from the currentEnrollments API call:

Response Body

```
[
  {
    "learningPathName": "001 (mixed units) - Update for AutoCAD 2020 & 2021 Users",
    "courseName": "Collaboration Tools",
    "userId": "████████████████████",
    "employeeID": "",
    "userName": "████████████████████",
    "userEmail": "████████████████████",
    "enrollmentStatus": "Not Started",
    "enrollmentProgress": 0,
    "enrollmentDueDate": "0001-01-01T00:00:00",
    "assignedByUser": "████████████████████"
  },
  {
    "learningPathName": "001 (mixed units) - Update for AutoCAD 2020 & 2021 Users",
    "courseName": "Editing Commands",
    "userId": "538c406f-1037-43b5-99eb-c1f7cdfaf114",
    "employeeID": "",
    "userName": "Chris Ryan (Production)"
  }
]
```

Response Code

200

JavaScript Example:

```
function ReportCurrentEnrollments() {
    //Debug console.log("Start - ReportCurrentEnrollments");
    var searchCount = 0;
    var xhr = new XMLHttpRequest();
    var localURL = m_BaseURL + "reports/currentEnrollments"
    xhr.open("GET", localURL, true);
    xhr.setRequestHeader("Content-Type", "text/json");
    xhr.setRequestHeader("Accept", "application/json");
    xhr.setRequestHeader("apiKey", ApiKey);
    xhr.setRequestHeader("userAccessKey", UserBearerToken);

    xhr.onload = function (e) {
        if (xhr.readyState === 4) {
            if (xhr.status === 200) {
                let enrollments = JSON.parse(xhr.responseText);
                for(var enrollment in enrollments) {
                    for(var values in enrollments[enrollment]) {
                        if (values.match(/learningPathName|courseName|userId|userName|userEmail/)) {
                            if (values.match(/learningPathName/)) {
                                searchCount=searchCount+1;
                                console.log( "***Search " + values + " value is: " + enrollments[enrollment][values]);
                            }
                            else {
                                console.log( "    " + values + " value is: " + enrollments[enrollment][values]);
                            }
                        }
                    }
                }
            }
            console.log("***Total Search Records returned: " + searchCount);
        } else {
            console.log("***Error Found");
            console.error(xhr.statusText);
        }
    };
    xhr.onerror = function (e) {
        console.log("***Error Found");
        console.error(xhr.statusText);
        console.error(e);
    };
    xhr.send("");
}
```

Reports – Learning Path Progress

Listing of learning path progress for all users.

A call to the 'LPPProgress API' will return the list of learning path progress for all users in your organization. Enter date criteria with this API to narrow down the results to specific timeframes.

This API requires the ApiKey from the apiToken API and the UserAccessKey from the userToken API.

Sample call to get the LPPProgress :

GET /api/reports/LPPProgress Retrieve Learning Path progress information for all users.

Implementation Notes
This method will return a list of all learning path progress for the organization based on the date parameters.

Date parameter examples:

- If you want to pull data starting with date="2022-06-01", you would enter startDate="2022-06-01".
- If you want to pull data that were completed prior to "2022-04-01", you would enter an endDate="2022-04-01".
- If you want to pull data for a date range like "2021-12-01" to "2022-03-15", you would enter a startDate="2021-12-01" and an endDate="2022-03-15".
- If you want to pull data for a specific date like "2021-06-10", you would enter a startDate="2021-06-10" and endDate="2021-06-11".

Response Class (Status 200)
OK

Model | Example Value

```
[
  {
    "userName": "string",
    "learningPathName": "string",
    "email": "string",
    "courseName": "string",
    "courseVersion": "string",
    "userId": "00000000-0000-0000-0000-000000000000",
    "status": "string",
    "statusDate": "2022-07-28T15:29:18.651Z",
  }
]
```

Response Content Type:

Parameters

Parameter	Value	Description	Parameter Type	Data Type
startDate	<input type="text"/>	Optional. The date to get data from in YYYY-MM-DD format.	query	string
endDate	<input type="text"/>	Optional. The date to get data to in YYYY-MM-DD format.	query	string
ApiKey	<input type="text" value="(required)"/>	Your API token. To get an API token, do a GET call to /api/authorization/apiToken with your applicationId.	header	string
UserAccessKey	<input type="text" value="(required)"/>	Your user token. To get a user token, POST your api key and credentials to /api/authorization/userToken.	header	string

Sample response from the LPPProgress API call:

The screenshot displays the response body and code for an API call. The response body is a JSON array containing one object with the following fields:

- userName: [REDACTED]
- learningPathName: "ACCO",
- email: [REDACTED]
- courseName: "Question Bank test v2 SCORM 1.2",
- courseVersion: "1.0",
- userId: [REDACTED]
- status: "Enrolled (Not Started)",
- statusDate: "2022-07-28T10:39:59",
- enrollmentDate: "2022-07-28T10:39:59",
- assignedBy: [REDACTED]
- dueDate: "2022-08-27T00:00:00",
- contentReviewedPercent: 0,
- viewingMinutes: 0,
- videoWatchPercent: 0,
- quizComposite: 0,
- courseProducts: "",
- employeeId: "",
- enrollmentId: "1404-0d3-1496-441-83-d-3e970204cc4d"

The response code is 200.

JavaScript Example:

```
function ReportLpProgress() {
    //Debug console.log("Start - ReportLpProgress");
    var searchCount = 0;
    var xhr = new XMLHttpRequest();
    var localURL = m_BaseURL + "reports/LPPProgress"
    xhr.open("GET", localURL, true);
    xhr.setRequestHeader("Content-Type", "text/json");
    xhr.setRequestHeader("Accept", "application/json");
    xhr.setRequestHeader("apiKey", ApiKey);
    xhr.setRequestHeader("userAccessKey", UserBearerToken);

    xhr.onload = function (e) {
        if (xhr.readyState === 4) {
            if (xhr.status === 200) {
                let lps = JSON.parse(xhr.responseText);
                for(var lp in lps) {
                    for(var values in lps[lp]) {
                        //Display results to console
                        if (values.match(/email|displayName|learningPathName|courseName|assignedDate|dueDate|completedDate/)) {
                            if (values.match(/learningPathName/)) {
                                searchCount=searchCount+1;
                                console.log( "***Search " + values + " value is: " + lps[lp][values]);
                            }
                            else {
                                console.log( " " + values + " value is: " + lps[lp][values]);
                            }
                        }
                    }
                }
                console.log("***Total Search Records returned: " + searchCount);
            } else {
                console.log("**Error Found");
                console.error(xhr.statusText);
            }
        }
    };
    xhr.onerror = function (e) {
        console.log("**Error Found");
        console.error(xhr.statusText);
        console.error(e);
    };
    xhr.send("");
}
```

Reports – Top Searches

This is a Usage Report for top searches done into the Pinnacle Series application.

A call to the topSearches API will return a list of search terms and the number of times each unique search term was searched for. Enter date criteria with this API to narrow down the results to specific timeframes.

This API requires the ApiKey from the apiToken API and the UserAccessKey from the userToken API.

Sample call to get the topSearches :

GET /api/reports/topSearches Retrieve a list of the top search terms used in the Pinnacle Series application.

Implementation Notes
This method will return a list of unique search terms used in the Pinnacle Series application and how many times that search term was used. The results are based on the data parameters if used.

Date parameter examples:

- If you want to pull data starting with date="2022-06-01", you would enter startDate="2022-06-01".
- If you want to pull data that were completed prior to "2022-04-01", you would enter an endDate="2022-04-01".
- If you want to pull data for a date range like "2021-12-01" to "2022-03-15", you would enter a startDate="2021-12-01" and an endDate="2022-03-15".
- If you want to pull data for a specific date like "2021-06-10", you would enter a startDate="2021-06-10" and endDate="2021-06-11".

Response Class (Status 200)
OK

Model | Example Value

```
{
  "searchData": [
    {
      "searchTerm": "string",
      "numberOfSearches": 0
    }
  ],
  "totalSearches": 0
}
```

Response Content Type:

Parameters

Parameter	Value	Description	Parameter Type	Data Type
startDate	<input type="text"/>	Optional. The date to get data from in YYYY-MM-DD format.	query	string
endDate	<input type="text"/>	Optional. The date to get data to in YYYY-MM-DD format.	query	string
ApiKey	<input type="text" value="(required)"/>	Your API token. To get an API token, do a GET call to /api/authorization/apiToken with your applicationId.	header	string
UserAccessKey	<input type="text" value="(required)"/>	Your user token. To get a user token, POST your api key and credentials to /api/authorization/userToken.	header	string

Sample response from the topSearches API call:



The screenshot displays a web-based interface for viewing an API response. It is divided into two main sections: 'Response Body' and 'Response Code'. The 'Response Body' section contains a JSON array of search data. The 'Response Code' section shows the status code '200'.

```
Response Body
```

```
{
  "searchData": [
    {
      "searchTerm": "the",
      "numberOfSearches": 1
    },
    {
      "searchTerm": "health insurance",
      "numberOfSearches": 1
    },
    {
      "searchTerm": "cch",
      "numberOfSearches": 1
    },
    {
      "searchTerm": "ace",
      "numberOfSearches": 1
    }
  ],
  "totalSearches": 4
}
```

Response Code

```
200
```

JavaScript Example:

```
function ReportTopSearches() {
    //Debug console.log("Start - ReportTopSearches");
    var searchCount = 0;
    var xhr = new XMLHttpRequest();
    var localURL = m_BaseURL + "reports/topSearches"
    xhr.open("GET", localURL, true);
    xhr.setRequestHeader("Content-Type", "text/json");
    xhr.setRequestHeader("Accept", "application/json");
    xhr.setRequestHeader("apiKey", ApiKey);
    xhr.setRequestHeader("userAccessKey", UserBearerToken);

    xhr.onload = function (e) {
        if (xhr.readyState === 4) {
            if (xhr.status === 200) {
                let searches = JSON.parse(xhr.responseText);
                for(var search in searches) {
                    if (search.match(/totalSearches/)) {
                        console.log("* Total Unique Searches: " + searches[search]);
                    }
                    if (search.match(/searchData/)) {
                        for(var terms in searches[search]) {
                            for(var result in searches[search][terms]) {
                                if (result.match(/searchTerm|numberOfSearches/)) {
                                    console.log(result + " value is: " + searches[search][terms][result]);
                                }
                            }
                        }
                    }
                }
            }
            else {
                console.error(xhr.statusText);
            }
        }
    };
    xhr.onerror = function (e) {
        console.log("*Error Found");
        console.error(xhr.statusText);
        console.error(e);
    };
    xhr.send("");
}
```

Reports – Unique User Sign-Ins

This Usage Report will return a listing of unique user sign-ins to the Pinnacle Series application.

A call to this API will return statistics by date of the number of users that have signed into the Pinnacle Series application during the last 7 days. The date range used is based on the current date of when you ran this report. The end date is the current date, and the start date is the current date, less 7 days. Results are returned in ascending order by the sign-in date.

This API requires the ApiKey from the apiToken API and the UserAccessKey from the userToken API.

Sample call to get the uniqueUsers:

GET /api/reports/uniqueUsers Retrieve a summary of unique user sign-ins to the Pinnacle Series application.

Implementation Notes
This method will return a summary by date of unique user sign-ins to the Pinnacle Series application during the last 7 days. It takes the current date less 7 days as the start date range and uses the current date as the end date range.

Response Class (Status 200)
OK

Model | Example Value

```
{
  "uniqueUsers": [
    {
      "signInDate": "string",
      "uniqueLogins": 0
    }
  ],
  "totalUniqueLogins": 0
}
```

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
ApiKey	<input type="text" value="(required)"/>	Your API token. To get an API token, do a GET call to /api/authorization/apiToken with your applicationId.	header	string
UserAccessKey	<input type="text" value="(required)"/>	Your user token. To get a user token, POST your api key and credentials to /api/authorization/userToken.	header	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		

Sample response from the uniqueUsers API call:

The screenshot displays the response body and code for an API call. The response body is a JSON object with a key "uniqueUsers" containing an array of five objects. Each object represents a date and the number of unique logins for that date. The response code is 200.

```
Response Body
```

```
{
  "uniqueUsers": [
    {
      "signInDate": "7/22/2022",
      "uniqueLogins": 0
    },
    {
      "signInDate": "7/23/2022",
      "uniqueLogins": 1
    },
    {
      "signInDate": "7/24/2022",
      "uniqueLogins": 0
    },
    {
      "signInDate": "7/25/2022",
      "uniqueLogins": 1
    },
    {
      "signInDate": "7/26/2022"
    }
  ]
}
```

Response Code

```
200
```

JavaScript Example:

```
function ReportUniqueUsers() {
    //DEBUG console.log("Start - ReportUniqueUsers");
    var searchCount = 0;
    var xhr = new XMLHttpRequest();
    var localURL = m_BaseURL + "reports/uniqueUsers"
    xhr.open("GET", localURL, true);
    xhr.setRequestHeader("Content-Type", "application/json");
    xhr.setRequestHeader("Accept", "application/json");
    xhr.setRequestHeader("ApiKey", ApiKey);
    xhr.setRequestHeader("UserAccessKey", UserBearerToken);
    xhr.onload = function (e) {
    if (xhr.readyState === 4) {
        if (xhr.status === 200) {
            let users = JSON.parse(xhr.responseText);
            for(var k in users) {
                if (k.match(/totalUniqueLogins/)) {
                    console.log("* Total Unique Logins: " + users[k]);
                }
                if (k.match(/uniqueUsers/)) {
                    for(var k1_1 in users[k]) {
                        for(var k1_2 in users[k][k1_1]) {
                            if (k1_2.match(/signInDate|uniqueLogins/)) {
                                console.log(k1_2 + " value is: " + users[k][k1_1][k1_2]);
                            }
                        }
                    }
                }
            }
        }
        else {
            console.error(xhr.statusText);
        }
    }
    };
    xhr.onerror = function (e) {
        console.log("*Error Found");
        console.error(xhr.statusText);
        console.error(e);
    };
    xhr.send("");
}
```