

User Management REST API

This document details the RESTful API methods available to manage users for Praesidium clients.

Changelog

2024-08-18	<ul style="list-style-type: none">• Added note about ignoring duplicate requests
2023-10-30	<ul style="list-style-type: none">• Addition of content removal capabilities to the API<ul style="list-style-type: none">◦ New Endpoint for removal of content from User• Addition of content re-enrollment capabilities to the API<ul style="list-style-type: none">◦ New Endpoint for re-enrollment content in User• Updated the "Create a new user" notes to explain changes to the functionality of the HTTP 400 errors that may occur.• For "Create a new user", added a note about Academy requiring unique emails
2023-01-18	<ul style="list-style-type: none">• Added examples with and without content specification• Updated general object structure to specify some fields as optional/ignored• Noted lack of content return from GET• Noted which UUIDs will be provided by Praesidium and the general format in which they will be provided
2022-10-18	<ul style="list-style-type: none">• Added content enrollment to API<ul style="list-style-type: none">◦ User Creation & Update impacted◦ New endpoint for enrollment of User in content◦ added content into structure of User object• Clarification of JWT expiration timeframe
2021-05-18	<ul style="list-style-type: none">• Added instructions on how to use refresh token• changed field name "program-type" in user object to "program_type" for consistency

2021-02-26

- Added structure of User Object
- Endpoint URLs updated
- Sample requests updated to match structure and to use properly formatted UUIDs
- Clarified return value of creation endpoint
- Fixed typo in authentication success response
- Role details and descriptions added

Overview

Authorization will be handled by OAuth2 using JWT. There are methods provided for only one resource: Users. All URLs will be specified without hostname, as only the hostname need change to identify the test environment versus production.

Methods are available to perform the following operations:

1. Retrieve a User Record (GET)
2. Create a new User Record (POST)
3. Update a User Record (PUT)
4. Deactivate a user record (PUT) - remove system access
5. Add content to a user record (PUT)

Unless otherwise specified, all services require the JWT token to be passed in the Authorization: Bearer header

In addition to all other potential error messages listed below, any endpoint may return HTTP 429 Too Many Requests. If this code is returned, a Retry-After header will be attached to the message to indicate how long to wait before retrying the request.

Any duplicate requests to change data (e.g. Create, Update, Deactivate, Enroll) within 30 seconds will be ignored.

Authentication

JWT expiration is 900 seconds (15 minutes)

URL	/portal/authenticate/login_api
Method	POST

Data Format

```
{
  "grant_type": "client_credentials",
  "client_id": "<Praesidium provided client ID>",
  "client_secret": "<Praesidium provided client secret>",
  "scope": [ "<scope1>", "<scope2>", ..., "<scopeN>" ]
}
```

The only valid scope value is: "prae.client.api.user"

Success Response

HTTP Status Code: 200

```
{
  "access_token": "<JWT>",
  "token_type": "bearer",
  "expires_in": 900,
  "refresh_token": "<JWT>"
}
```

The Refresh token can be used to generate a new access token (if it has not expired) by passing it in the Authorization: Bearer header and using an HTTP GET to retrieve /portal/session/refresh_jwt

Failure Response

Authentication Error

HTTP Status Code: 401 Authorization Required

```
{
  "error": "invalid_client",
  "error_description": "[string]",
  "expires_uri": "[reserved for future use]"
}
```

Input Validation Error

HTTP Status Code: 400 Bad Request

```
{
  "error": "invalid_request",
  "error_description": "[string]",
  "expires_uri": "[reserved for future use]"
}
```

Invalid Scope Error

HTTP Status Code: 400 Bad Request

```
{
  "error": "invalid_client",
  "error_description": "[string]",
  "expires_uri": "[reserved for future use]"
}
```

Example

```
// JQuery
$.ajax({
  url: "/portal/authenticate/login_api",
  dataType: "json",
  data: {
    "grant_type": "client_credentials",
    "client_id": "xyz_client",
    "client_secret": "itsasecret",
    "scope": [
      "prae.client.api.user"
    ]
  },
  type: "POST",
  success: function( r ) {
    console.log( r );
  }
});
```

Resource: User

Structure of the User Object

All data fields in the user object will fall into one of five categories:

1. Principal – this category is for personal information that is primarily used to identify the user
2. System – this category is for system information, such as roles, status (active or inactive)
3. Person – this category is for other personal information that is not used to identify the user
4. Context – this category is for information describing the contexts that apply to the user
5. Attributes – this category is for information describing the user that can be used to determine which course assignments are applied, and for reporting purposes

So, in general a user object (for the purpose of this API is defined by the following format:

```
{
  "principal": {
```

```

    "UUID": "<string>",
    "first_name": "<string>",
    "last_name": "<string>",
    "email": "<email>",
    "client_external_id": "<string>"
  },
  "system": {
    "status": "<(active | inactive)>",
    "role": "<role-value>"
  },
  "person": {
  },
  "context": {
    "client": "<UUID>",
    "locations": [
      "<UUID1>",
      "<UUID2>",
      ...,
      "<UUIDN>"
    ],
    "content": [
      {
        "uuid": "<UUID>",
        "type": "<content_type-value>"
      },
      ...
    ]
  },
  "attributes": {
    "program_type": "<program_type-value>",
    "position": "<position-value>"
  }
}

```

Currently, the only valid content_type-values are "course" and "learning path"

The only valid role-values are:

- **Learner:** The user in question can only take courses, and has no administrative access
- **Administrator:** The user in question can edit user info for non-administrative user

- **Administrator - View Only:** The user in question has administrative access, but cannot edit any information

General Notes about the fields included in the user object relevant to individual REST operations

When using a POST to create a user, the UUID field at the top level is optional.

The "content" object under the "context" object is optional when passing a user object to POST for creation or PUT for general update

In addition, the "content" object will never be specified in a return value from an API call for performance reasons

The UUIDs under "client", "locations" and "content" will be provided by Praesidium (csv file will contain UUID, name for client and locations, and will additionally contain content_type and SKU for courses/learning paths)

Retrieve a User Record

URL	/portal/lms_api/v2/user/{UUID}
Method	GET
Path Parameters	<u>Required</u> UUID - UUID for the desired user
Success Response	HTTP Status Code: 200 User object (structure as described above)

Failure Response

Input Validation Error

HTTP Status Code: 400 Bad Request

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Not Found Error

HTTP Status Code: 404 Not Found

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Internal Server Error

HTTP Status Code: 500 Internal Server Error

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

status - integer HTTP status

code - internal error code (if available, otherwise HTTP Status)

message - Human readable error messages, separated by new lines

technicalMessage - Human readable error message, separated by new lines, containing technical detail

Example

```
// JQuery
$.ajax({
  url: "/portal/lms_api/v2/user/aaaaaaaa-
bbbb-cccc-dddd-5ce2287baaa",
  type: "GET",
  success: function( r ) {
    console.log( r );
  }
});
```

Create a New User

Upon creation, the user will be automatically enrolled in any content included in the payload

Email addresses must be unique across all Academy.

If an attempt is made to create a user with an email that already exists in Academy, the technicalMessage property of the HTTP 400 error will now contain the id of the user with the existing email IF that user falls under the requesting client's hierarchy.

URL	/portal/lms_api/v2/user
Method	POST
Data Parameters	<u>Required</u> User object (structured as above). Any value in the UUID field will be ignored.
Success Response	HTTP Status Code: 201 User object (structure as described above). The UUID field will contain the UUID of the newly created user

Failure Response

Input Validation Error
HTTP Status Code: 400 Bad Request

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Internal Server Error
HTTP Status Code: 500 Internal Server Error

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

status - integer HTTP status

code - internal error code (if available, otherwise HTTP Status)

message - Human readable error messages, separated by new lines

technicalMessage - Human readable error message, separated by new lines, containing technical detail

Example

```
// JQuery (without content)
$.ajax({
  url: "/portal/lms_api/v2/user",
  type: "POST",
  dataType: "json",
  data: {
    "principal": {
      "first_name": "John",
      "last_name": "Smith",
      "email": "jsmith@example.com",
      "client_external_id": "123456958"
    },
    "system": {
      "status": "active",
      "role": "Administrator - View Only"
    },
    "person": {},
    "context": {
      "client": "aaaaaaaa-bbbb-cccc-dddd-5ce2287baaa",
      "locations": [
        "aaaaaaaa-bbbb-cccc-dddd-5ce1207baaa"
      ]
    },
    "attributes": {
      "position": "director (camp)",
      "program_type": "aquatics"
    }
  },
  success: function( r ) {
    console.log( r );
  }
});

// JQuery (with content)
$.ajax({
  url: "/portal/lms_api/v2/user",
  type: "POST",
  dataType: "json",
  data: {
    "principal": {
      "first_name": "John",
      "last_name": "Smith",
```

Update an Existing User

Note: upon completion of other updates, the user will be automatically enrolled in any content included in the payload

URL	/portal/lms_api/v2/user/{UUID}
Method	PUT
Path Parameters	<u>Required</u> UUID UUID for the user to be updated
Data Parameters	<u>Required</u> User object (structured as above).
Success Response	HTTP Status Code: 200 User object (structure as described above).

Failure Response

Authorization Error

HTTP Status Code: 403

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Input Validation Error

HTTP Status Code: 400 Bad Request

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Not Found Error

HTTP Status Code: 404 Not Found

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Internal Server Error

HTTP Status Code: 500 Internal Server Error

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

status - integer HTTP status

code - internal error code (if available, otherwise HTTP Status)

message - Human readable error messages, separated by new lines

technicalMessage - Human readable error message, separated by new lines, containing technical detail

Example

```
// JQuery (without content)
$.ajax({
  url: "/portal/lms_api/v2/user/aaaaaaaa-
bbbb-cccc-dddd-5ce9942baaa",
  type: "PUT",
  dataType: "json",
  data: {
    "principal": {
      "first_name": "John",
      "last_name": "Smith",
      "email": "jsmith@example.com",
      "client_external_id": "123456958"
    },
    "system": {
      "status": "active",
      "role": "Administrator - View Only"
    },
    "person": {},
    "context": {
      "client": "aaaaaaaa-bbbb-cccc-dddd-
5ce2287baaa",
      "locations": [
        "aaaaaaaa-bbbb-cccc-dddd-
5ce1207baaa"
      ]
    },
    "attributes": {
      "position": "director (camp)",
      "program_type": "aquatics"
    }
  },
  success: function( r ) {
    console.log( r );
  }
});

// JQuery (with content)
$.ajax({
  url: "/portal/lms_api/v2/user/aaaaaaaa-
bbbb-cccc-dddd-5ce9942baaa",
  type: "PUT",
  dataType: "json",
  data: {
    "principal": {
```

Deactivate an Existing User (Remove system access)

URL	/portal/lms_api/v2/user/{UUID}/deactivate
Method	PUT
Path Parameters	<u>Required</u> UUID - UUID for the desired user
Success Response	HTTP Status Code: 200 User object (structure as described above)

Failure Response

Authorization Error

HTTP Status Code: 403

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Input Validation Error

HTTP Status Code: 400 Bad Request

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Not Found Error

HTTP Status Code: 404 Not Found

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Internal Server Error

HTTP Status Code: 500 Internal Server Error

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

status - integer HTTP status
code - internal error code (if available, otherwise HTTP Status)
message - Human readable error messages, separated by new lines
technicalMessage - Human readable error message, separated by new lines, containing technical detail

Example	<pre>// JQuery \$.ajax({ url: "/portal/lms_api/v2/user/aaaaaaaa- bbbb-cccc-dddd-5ce2287baaa/deactivate", type: "PUT", success: function(r) { console.log(r); } });</pre>
----------------	--

Add Content to an Existing User

URL	/portal/lms_api/v2/user/{UUID}/enrollContent
Method	PUT
Path Parameters	<u>Required</u> UUID UUID for the user to be updated
Data Parameters	<u>Required</u> Content Listing (structured as below). <div> <pre>{ "content": [{ "uuid": "[string]", "type": "[string]" }, ...] }</pre> </div> <p>uuid - course UUID (Praesidium provided) type - only the values "course" and "learning path" are available</p>
Success Response	HTTP Status Code: 200 User object (structure as described above).

Failure Response

Authorization Error

HTTP Status Code: 403

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Input Validation Error

HTTP Status Code: 400 Bad Request

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Not Found Error

HTTP Status Code: 404 Not Found

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Internal Server Error

HTTP Status Code: 500 Internal Server Error

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

status - integer HTTP status

code - internal error code (if available, otherwise HTTP Status)

message - Human readable error messages, separated by new lines

technicalMessage - Human readable error message, separated by new lines, containing technical detail

Example

```
// JQuery
$.ajax({
  url: "/portal/lms_api/v2/user/aaaaaaaa-
bbbb-cccc-dddd-5ce9942baaa/enrollContent",
  type: "PUT",
  dataType: "json",
  data: {
    "content": [
      {
        "uuid": "aaaaaaaa-bbbb-cccc-dddd-
5ce1248baaa",
        "type": "course"
      }
    ]
  },
  success: function( r ) {
    console.log( r );
  }
});
```

Remove Content from an Existing User

WARNING: Removing a Learning Path from a user will **NOT** remove the corresponding courses

URL	/portal/lms_api/v2/user/{UUID}/removeContent
Method	PUT
Path Parameters	<u>Required</u> UUID UUID for the user to be updated

<p>Data Parameters</p>	<p><u>Required</u> Content Listing (structured as below).</p> <div data-bbox="813 215 1485 674"><pre>{ "content": [{ "uuid": "[string]", "type": "[string]" }, ...,] }</pre></div> <p>uuid - course UUID (Praesidium provided) type - only the values "course" and "learning path" are available</p>
<p>Success Response</p>	<p>HTTP Status Code: 200 User object (structure as described above).</p>

Failure Response

Authorization Error

HTTP Status Code: 403

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Input Validation Error

HTTP Status Code: 400 Bad Request

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Not Found Error

HTTP Status Code: 404 Not Found

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Internal Server Error

HTTP Status Code: 500 Internal Server Error

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

status - integer HTTP status
code - internal error code (if available, otherwise HTTP Status)
message - Human readable error messages, separated by new lines
technicalMessage - Human readable error message, separated by new lines, containing technical detail

Example

```
// JQuery
$.ajax({
  url: "/portal/lms_api/v2/user/aaaaaaaa-
bbbb-cccc-dddd-5ce9942baaa/removeContent",
  type: "PUT",
  dataType: "json",
  data: {
    "content": [
      {
        "uuid": "aaaaaaaa-bbbb-cccc-dddd-
5ce1248baaa",
        "type": "course"
      }
    ]
  },
  success: function( r ) {
    console.log( r );
  }
});
```

Re-Enroll Content in an Existing User

Note: User's content that is re-enrolled will be set to a not-started status in Academy. Any prior completion certificates will be unchanged.

WARNING: Re-Enrolling a user in a Learning Path will **NOT** re-enroll that user in the corresponding courses

URL	/portal/lms_api/v2/user/{UUID}/reEnrollContent
Method	PUT
Path Parameters	<u>Required</u> UUID UUID for the user to be updated

<p>Data Parameters</p>	<p><u>Required</u> Content Listing (structured as below).</p> <div data-bbox="813 215 1485 674"><pre>{ "content": [{ "uuid": "[string]", "type": "[string]" }, ...,] }</pre></div> <p>uuid - course UUID (Praesidium provided) type - only the values "course" and "learning path" are available</p>
<p>Success Response</p>	<p>HTTP Status Code: 200 User object (structure as described above).</p>

Failure Response

Authorization Error

HTTP Status Code: 403

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Input Validation Error

HTTP Status Code: 400 Bad Request

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Not Found Error

HTTP Status Code: 404 Not Found

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

Internal Server Error

HTTP Status Code: 500 Internal Server Error

```
{
  "status": [integer],
  "code": "[string]",
  "message": "[string]",
  "technicalMessage": "[string]",
  "infoURI": "[reserved for future use]"
}
```

status - integer HTTP status
code - internal error code (if available, otherwise HTTP Status)
message - Human readable error messages, separated by new lines
technicalMessage - Human readable error message, separated by new lines, containing technical detail

Example

```
// JQuery
$.ajax({
  url: "/portal/lms_api/v2/user/aaaaaaaa-
bbbb-cccc-dddd-
5ce9942baaa/reEnrollContent",
  type: "PUT",
  dataType: "json",
  data: {
    "content": [
      {
        "uuid": "aaaaaaaa-bbbb-cccc-dddd-
5ce1248baaa",
        "type": "course"
      }
    ]
  },
  success: function( r ) {
    console.log( r );
  }
});
```

Revision #10

Created 14 December 2022 22:27:15 by Scott Fulbright

Updated 19 August 2024 04:57:06 by Scott Fulbright