Resident Management API - Quick Start Guide

This quick start guide details how to access the Resident Management API and use each endpoint to manage residents within Site Manager sites. For detailed reference about how Site Manager functions and interacts with the Resident Management API, see Resident Management API - Overview.

We have created a Postman collection for your convenience. You can download the collection from the Resident Management API - Postman Collection article.

All of the examples in this guide use a service account with the staff user type and non-elevated permissions. Please see the Resident Management API - Access and Permissions guide for more information on creating a service account and the permissions that best fit your needs.

Change Log

Date	Changes Made
11/11/2021	-Initial release
06/08/2022	-Edit article for grammar and spelling errors

Base URL

All requests must use the following base URL:

```
https://api.11os.com/v2
```

Authorization

To interact with the Resident Management API, you must first obtain an access token and use it in the `Authorization` header of all requests. You can obtain your access token using the following endpoint:

```
POST /oauth/token
```

The easiest way to generate your access token is by Base64-encoding your Site Manager user email and password and putting it into the `Authorization` header.

Your Base64-encoded credentials should look something like this: `example@email.com:ElevenIsAwesome!` = `YmlsbHlib2JAZ21haWwuY29tOlBhbmNha2VzNDIwNjk=`

The `Authorization` header will look like this: `Authorization: Password YmlsbHlib2JAZ21haWwuY29tOlBhbmNha2VzNDIwNjk=`

This endpoint returns a JSON object containing the access token you must use in all subsequent requests.

Request Examples

cURL Python

Example Response

Status: 200

Successful operation

JSON

Use the `access_token` value as your bearer token for all subsequent requests to the Resident Management API.

Find a Team ID

Now that you have an access token, you can start using the Resident Management API. First, you need to obtain a Team ID. A Team ID uniquely identifies each team within Site Manager and is required for all endpoints with the `TeamId` parameter.

To find your Team ID, use the following endpoint:

```
GET /team
```

This endpoint returns all of the teams that the account used to make the request is a member of. If your account is only on one team, then the `Teams` array in the response will only contain one team. If your account is on multiple teams, ensure that you use the correct Team ID of the sites within it.

Request Examples

cURL Python

Response

Status: 200

Successful operation

Save the `TeamId` value as the `TeamId` parameter in endpoints that require it.

JSON

Find a Site ID

To start managing residents, you need the Site ID of the site you wish to manage residents. A Site ID is required for all endpoints that require the `SiteId` parameter.

To find your Site ID, use the following endpoint:

```
GET /team/{TeamId}/site
```

This endpoint returns a list of all sites on the specified team that the account has access to. If your account has access to only one site, then the `Sites` array in the response will only contain one site. If your account has access to multiple sites, ensure that you use the correct Site ID of the site you want to manage residents.

Request Examples

cURL Python

Response

Status: 200

Successful operation

JSON

Save the `SiteId` value as the `SiteId` parameter in endpoints that require it.

Get PAN Mappings

Now that you have a Team and Site ID, you may want to manage residents right away. However, before you start managing residents, you will also need to know the available PAN Mapping configuration of the site in Site Manager.

When managing an individual resident or multiple residents at once, the Building, Floor, and Unit values sent in the POST request must match available values returned from the `GET /team/{TeamId}/site/{SiteId}/mappings` endpoint.

To get a list of PAN Mappings at a site, use the following endpoint:

```
GET /team/{TeamId}/site/{SiteId}/mappings
```

Request Examples

cURL Python

Response

Status: 200

Successful operation

JSON

Use the returned values from the `Mappings` array to fill out the `Building`, `Floor`, and `Unit` values in the `ResidentObject` object for adding individual residents or the `ResidentImportObject` object when adding multiple residents. The `Passphrase`, `SSID`, and `Vlan` items are not required for adding residents.

Add Residents

Now that you have a Team ID, Site ID, and PAN Mapping values, you can start adding residents to the desired site. You can add residents individually or bulk import multiple residents.

Keep in mind the following requirements when adding residents:

- Only one resident can occupy a unit on any given date or date range.

Add Individual Residents

First, add a single resident to the site using the Team ID and Site ID you received from the previous requests.

To add residents individually, use the following endpoint, which imports a JSON object containing the one resident you want to add:

```
POST /team/{TeamId}/site/{SiteId}/resident
```

In the body of the `/team/{TeamId}/site/{SiteId}/resident` request, you must include the `ResidentObject` object.

The `ResidentObject` contains all the necessary information to create a resident. If any of this information needs to change in the future, please use the endpoint for updating a resident.

Important!
You cannot change the Email field once set.
You cannot add a resident if their email already exists in Site Manager.

Example Request Body

JSON

Request Examples

cURL Python

Response

Status: 200

Successful operation

A JSON object is returned with the new resident's status. Save the `ResidentId` value as the `ResidentId` parameter in endpoints that require it.

Add Multiple Residents

To add multiple residents at once, use the following endpoint, which imports a JSON object containing all residents you want to add:

```
POST /team/{TeamId}/site/{SiteId}/resident/import
```

Be Careful If Using Excel!
If you create the JSON object of residents from a .csv file opened in Excel, the time and date formatting may be incorrect if saved in Excel. Please ensure that you are using the correct date and time format. Or use a text editor to make adjustments to the .csv file.

Request Body Example

In the body of the `POST` request, you must include the `ResidentImportObject` request body object payload.

JSON

API and UI Field Names
When using the bulk upload feature in the Site Manager UI, the downloadable import template uses different parameter names than the Resident Management API. The Site Manager UI import template uses more user-friendly terms which are not compatible with the Resident Management API. **Table-2** below shows the differences between the Resident Management API and Site Manager UI job field names for bulk uploading:

Table-2 Resident Management API and Site Manager UI field name differences for bulk importing residents

Resident Management API	Site Manager UI

FirstName	First Name
LastName	Last Name
PhoneNumber	Phone Number
ActivationDateUTC	Access Start
DeactivationDateUTC	Access End

Request Examples

cURL Python

Response

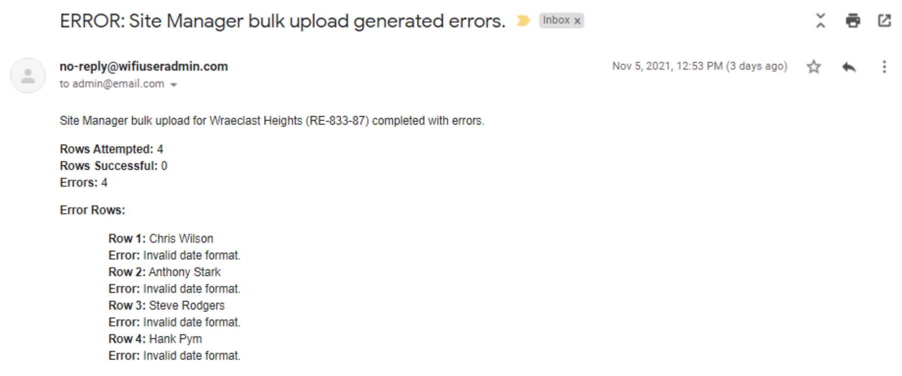
Status: 200

Successful operation

JSON

A JSON object is returned containing the Job ID of the import job. Save the `job_id` value as the `job_id`parameter in endpoints that require it. This endpoint only returns a Job ID and does not indicate whether the job has failed or succeeded. If any errors were present in the import job, the following endpoint describes how to find them.

Additionally, Site Manager sends an email to the account associated with the access token that made the call. This email provides the same breakdown available in the next endpoint. Example email:


ERROR: Site Manager bulk upload generated errors.

no-reply@wifiuseradmin.com
to admin@email.com

Nov 5, 2021, 12:53 PM (3 days ago)

Site Manager bulk upload for Wraeclast Heights (RE-833-87) completed with errors.

Rows Attempted: 4
Rows Successful: 0
Errors: 4

Error Rows:

Row 1: Chris Wilson
Error: Invalid date format.
Row 2: Anthony Stark
Error: Invalid date format.
Row 3: Steve Rodgers
Error: Invalid date format.
Row 4: Hank Pym
Error: Invalid date format.

Get Import Job Status

After importing residents and receiving the Job ID, you can check the status of the import job using the following endpoint:

```
GET /team/{TeamId}/site/{SiteId}/resident/import/{import_job_id}
```

Request Examples

cURL   Python

Response

An incomplete job will not display the `result` field because the job has not finished yet. Once a job is complete, the `result` field displays each resident that was successfully added or failed. Errors found during the import contain the value where the error occurred for each resident.

Table-3 below contains the fields and their descriptions that are returned from this endpoint:

Table-3 Import Job Status Fields

Field Name	Description

`failed`	The number of residents that failed to import.
`progress`	The percentage of job completion.
`result`	An array containing arrays of residents and their details that were successfully imported (success) or failed to import (failed). Errors are indicated in the failed array.
`status`	The status of the job (`started`, `finished`).
`success`	The number of residents successfully imported.
`total`	The sum of success and failed residents attempted to import.

Below are three example responses; an incomplete job's response, a complete job's response that succeeded in importing, and a complete job's response that failed to import.

Status: 200

Successful operation

JSON (Incomplete)   JSON (Completed - Success)   JSON (Completed - Failed)

Get All Residents

Now that you've added some residents to the site, you can now view all residents at the site.

To get all residents at a site, use the following endpoint:

```
GET /team/{TeamId}/site/{SiteId}/resident
```

Request Examples

cURL Python

Response

A JSON object is returned with all of the residents at a site. Save the `ResidentId` value for desired residents you want to manage as the `ResidentId` parameter in endpoints that require it.

Status: 200

Successful operation

JSON

Get an Individual Resident

After getting the list of all residents at a site from the previous endpoint, you can use the `ResidentId` of specific residents to see the details of each resident individually.

To get details about a specific resident at a site, use the following endpoint:

```
GET /team/{TeamId}/site/{SiteId}/resident/{ResidentId}
```

Request Examples

cURL Python

Response

Status: 200

Successful operation

```
JSON
```

Update a Resident

To update a resident, use the following endpoint:

```
PUT /team/{TeamId}/site/{SiteId}/resident/{ResidentId}
```

In the body of the `/team/{TeamId}/site/{SiteId}/resident/{ResidentId}` request, you must include the `ResidentObject` object.

The `ResidentObject` contains the editable fields for each resident.

Important!
Do not update the `email` field. A resident's email address is their unique identifier and cannot be changed. If you attempt to change their email using the `/team/{TeamId}/site/{SiteId}/resident/{ResidentId}` endpoint, you will encounter unforeseen errors, and the email will not change.

Example Request Body

Using the same resident from the example in the `GET /team/{{TeamId}}/site/{{SiteId}}/resident` response, for this example, the request body below contains a change to the resident's name in the `ResidentObject`.

```
JSON
```

Request Examples

Response

cURL Python

A successful response includes the same information as the `GET /team/{{TeamId}}/site/{{SiteId}}/resident` response but with the updated information that you sent in the `ResidentObject`.

Status: 200

Successful operation

JSON

Terminate a Resident

To terminate a resident, use the following endpoint:

```
DELETE /team/{TeamId}/site/{SiteId}/resident/{ResidentId}
```

When you terminate a resident, their account remains in Site Manager in the Expired status for a 31-day grace period before being automatically purged and completely removed. During the 31-day grace period, the resident no longer occupies the unit, and you can add new residents to this unit.

Terminating a resident changes their Deactivation Date to the day before you terminated them. For example, if you terminate a resident on Nov 10th, their Deactivation Date is set to Nov 9th.

You cannot add residents in the Expired status as they still exist in Site Manager. To reactivate service for a resident in Expired status, you must use the endpoint for updating a resident.

Request Examples

cURL Python

Response

The response returned contains a JSON object that indicates if an error occurred and the result of the termination. A `Result` value of `Terminated` means you successfully terminated the resident.

Status: 200

Successful operation

JSON

Terminate and Purge a Resident

To terminate and purge a resident, use the following endpoint:

```
DELETE /team/{TeamId}/site/{SiteId}/resident/{ResidentId}/purge
```

Terminating and purging a resident bypasses the 31-day grace period and immediately removes their account from Site Manager. Therefore, residents you terminate and purge cannot be updated because this completely removes them from Site Manager. If you accidentally terminated and purged a resident, you can add the resident as a brand new resident.

Terminating and purging residents can also be a helpful troubleshooting step if issues arise with a resident account.

Info
Residents already in the Expired status cannot be terminated and purged. Only residents that are in a non-expired status can be terminated and purged.

Request Examples

cURL   Python

Response

The response returned contains a JSON object that indicates if an error occurred and the result of the termination. A `Result` value of `Deleted` means you successfully terminated and purged the resident from Site Manager.

Status: 200

Successful operation

```
JSON
```

Resend Invitation Email or WiFi Details

To resend an invitation email to residents in Pending status or send the WiFi details to residents in non-Pending status, use the following endpoint:

```
POST /team/{TeamId}/site/{SiteId}/resident/{ResidentId}/invite
```

This endpoint has two main functions based on the status of a resident:

- **Pending Residents** - Resend the invitation email sent to residents on the Activation Date of their service. Note that subsequent invitation emails contain a new link different from the link sent in the first invitation email. However, all links sent remain valid until the resident has completed the onboarding process.
- **Non-Pending Residents** - Send the WiFi details (WiFi Network Name and Passphrase) in an email. WiFi details are only sent once a resident has completed the onboarding process.

Request Examples

```
cURL     Python
```

Response

Regardless of the

Status: 200

Successful operation

```
JSON
```

Reset a Resident's Pre-Shared Key

To reset a resident's pre-shared key, use the following endpoint:

```
POST /team/{TeamId}/site/{SiteId}/resident/{ResidentId}/pskreset
```

This endpoint sends an email to the resident containing a link redirecting them to the ElevenOS portal. On the ElevenOS portal page, the resident confirms that they want to reset their PSK, and after a few moments, the resident is sent another email containing their new PSK.

Important!
You can only use this endpoint on MDU sites with the Site Type of PPK-C and PPK-D.

Request Examples

cURL Python

Response

Status: 200

Successful operation

JSON

Suspend a Residents Status (Change Status)

To suspend or unsuspend a resident, use the following endpoint:

```
PATCH /team/{TeamId}/site/{SiteId}/resident/{ResidentId}/status
```

This endpoint can only suspend or unsuspend residents. You can only suspend residents in a

Current non-Pending status or unsuspend residents in a Current and Suspended status. If you attempt a suspend a resident that is not in Current non-Pending status, the state of that resident will remain unchanged, and the response returned will indicate no error occurred.

You must include the `ResidentStatus` object in the request body, which can only contain the following values for the `Status` field:

- `Active` - Changes a resident's status to Current non-Pending in the Site Manager UI.
- `Disabled` - Change a resident's status to Suspended and Current in the Site Manager UI.

If you attempt to send any other status in `ResidentStatus` object, this endpoint returns an error.

Example Request Body

JSON(Suspend)   JSON(Unsuspend)

Request Examples

cURL   Python

Response

Status: 200

Successful operation

JSON

Perform Bulk Action on Multiple Residents

To perform bulk actions on residents, use the following endpoint:

```
POST /team/{TeamId}/site/{SiteId}/resident/bulkaction
```

This endpoint allows you to perform the following actions in bulk by sending the `ResidentBulkAction` object.

If you attempt to send any other status in `ResidentBulkAction` object, this endpoint returns an error.

Example Request Body

In the body of the `POST` request, you must include the `ResidentBulkAction` request body object payload.

Table-4 below contains the allowed values for the `Action` field in the `ResidentBulkAction` endpoint:

Table-4 Allowed ResidentBulkAction Object Action Field Values

Action Field Value	Description
`suspend`	Suspends all residents in Current non-Pending status.
`unsuspend`	Unsuspends all residents in Current and Suspended status.
`terminate`	Terminates all residents in Current, Current and Pending, and Scheduled statuses.
`terminate-purge`	Terminates and purges all residents in Current, Current and Pending, and Scheduled statuses.

`resend-invites`	Resends invitation emails to all residents in Current and Pending statuses. OR Sends an email with WiFi details to all residents in Current and Current and Suspended statuses.
`onboard-now`	Triggers the onboarding process and sends the first invitation email to all Scheduled residents.

```
JSON
```

Request Examples

```
cURL     Python
```

Response

Status: 200

Successful operation

```
JSON
```

A JSON object is returned containing the Job ID of the bulk action job. Save the `job_id` value as the `job_id`parameter in endpoints that require it. This endpoint only returns a Job ID and does not indicate whether the job has failed or succeeded. If any errors were present in the import job, the next endpoint describes how to find them.

Additionally, Site Manager sends an email to the account associated with the access token that made the call. This email provides the same breakdown available in the next endpoint.

Get Bulk Action Job Status

After performing a bulk action on residents and receiving the Job ID, you can check the status of the bulk action job using the following endpoint:

```
GET /team/{TeamId}/site/{SiteId}/resident/bulkaction/{job_id}
```

Request Examples

cURL   Python

Response

An incomplete job will not display the `result` field because the job has not finished yet. Once a job is complete, the `result` field displays each resident that was successfully added or failed. Errors found during the import contain the value where the error occurred for each resident.

Table-5 below contains the fields and their descriptions that are returned from this endpoint:

Table-5 Bulk Action Job Status Fields

Field Name	Description
`failed`	The number of residents that failed to import.
`progress`	The percentage of job completion.
`result`	An array containing arrays of residents and their details that were successfully imported (success) or failed to import (failed). Errors are indicated in the failed array. The `result` array also includes the `action` field which contains the bulk action

	performed or the `action_invite` indicating invitation emails or WiFi details were sent to residents.
`status`	The status of the job (`started`, `finished`).
`success`	The number of residents successfully imported.
`total`	The sum of success and failed residents attempted to import.

Like the import multiple residents job, errors are presented in line with the item that caused an error.

Below are three example responses; an incomplete job's response, a complete job's response that succeeded in importing, and a complete job's response that failed to import.

Status: 200

Successful operation

JSON

Each success or failed resident in the `result` array contains the `result` field which describes the action taken indicating its success or failure.

Related Links

- Resident Management API - Overview
- Resident Management API - Access and Permissions
- Resident Management API - Postman Collection