Using REST APIs

in i-Vu® v10.0

Carrier ©2025 · Catalog No. 11-808-1222-01 · 10/28/2025

Verify that you have the most current version of this document from **www.hvacpartners.com**, the **Carrier Partner Community** website, or your local Carrier office.

Important changes are listed in **Document revision history** at the end of this document.

Contents

What are REST APIs?

REST APIs are stateless HTTP based programming interfaces to the i-Vu® system. The APIs support viewing the system structure, configuring the system structure, querying alarms, and other supporting operations. With each release of i-Vu® the set of REST APIs is expected to grow.

Authentication

All REST API interfaces require authentication and authorization. Authentication must be done over HTTPS unless the i-Vu® server is configured to “Allow SOAP applications over HTTP”. Machine to machine authentication currently has two options, Basic Authentication and API Key Authentication.

Basic Authentication

HTTP basic authentication is supported using simple user name and password. The operator account must also have the Remote Data Access privilege as well as the necessary privileges for the specific REST API. When using this method, the account should have the absolute minimum necessary privileges and care should be take to secure the password. This method is convenient for simpler tools like Excel®. Like all other forms of authentication, attempting to use an invalid password locks the account for some period of time.

API Key Authentication

An API Key, or “system key” can be created for a specific set REST operations. It can only be used with REST APIs and does not provide any other access to the i-Vu® system. An API key is not associated with an operator and does not use privilege sets. It is considered “fully privileged” but is granted access to only specific “REST end points”. This allows for an API key to be created just for a specific purpose.

API key is created by an administrator on the “API Keys” page on the config tree. Adding an API key requires the following:

- **Display name** - a label for the API key
- **Reference name** - the ID for the API key that must be passed in each request
- **Administrator** - a flag to indicating if the API key has administrator level privilege. Avoid using this flag whenever possible.
- **Expiration time** - the date and time (based on the server) at which the API key expires and is no longer valid
- **Endpoints** - the path to the endpoint relative to the server.
  Example: To get the list of alarm categories, the end point would be `/_alarm_serviceprovider/api/v1/alarm/categories` and uses HTTP `GET` . You can use `*` to mean anything starting with the end point. So to allow “read” access to all system service provider operations, the end point pattern would be `/_alarm_serviceprovider/api/v1/*` and only `GET` is selected. We recommend that multiple end point mappings be defined rather than using the wild card.

To pass an API key, there are two options.

"cj-api-key" Method

The preferred method is using a custom "cj-api-key" header send with a value of `<key-reference-name>:<application key value>`. The “application key value” is accessible and can only be copied immediately after creating the API key. It starts with `CJAPIKEY`. These values should not be base 64 encoded in the header as the key is already base 64 encoded

Example: `cj-api-key: my_key_ref:CJAPIKEY:1:NDNjOGM3YzMtNTQ5YS00N2E3LWIyMzctYjdmNzBkN2FkNGRl`

Basic authentication

The second method is in the header as "Basic" authentication. The username is "key-reference name" of the key, and password is the "application key value". This option is suggested when the client or tool does not have the flexibility to add a custom header. Note that these combined values are base 64 encoded in the header. Most tools allow entry of the values and automatically combine and encode them into the header.

Example: Authorization: `Basic a2V5MjpDSkFQSUtFWToxOk5ETmpPR00zWXpNdE5UUTVZUzAwTjJFM0xXSXlNemN0WWpkbU56QmtOMkZrTkdSbA==`

Standard JSON response

Most REST responses follow a standard structure.

- **payload** - If successful, the payload is the specific response for the REST API endpoint. It is typically a structured JSON record but sometimes it can be a simple JSON primitive value (for example, a string or number) or a simple array.
- **success** - Returns a boolean value - True is typically a successful HTTP 200 response. False occurs when there are errors when processing the request and is typically a 404 or 500 error.
- **code** - The HTTP response code
- **messages** - A list of strings that are informational messages in the response - Most requests do not have messages and are simply successful, or have an error detail in the **rfc7807Error** field.
- **context** - contextual information about the request itself - It includes the version, the date/time the server received the request, the date/time the server sent the response, and the request URL.
- **rfc7807Error** - this is either “null” for a successful request or contains structured information about the error.
  - **detail** - the error message detail - Common errors are translated, unexpected failures typically are not.
  - **type** - not currently set and only displays "about:blank"
  - **status** - the same as code and is the HTTP response code
  - **title** - a summary of the error - It can be used as the title for an error message.
  - **langKey** - a deprecated field that may be removed in the future
  - **invalidParamDetailList** - an array of strings or messages - It contains additional specific error details about invalid parameters/fields if the request provides additional details. If there are no errors it is blank.

Example successful response

```
{
"payload": { ... },
"success": true,
"code": "200",
"messages": [ ... ],
"context": {
"version": "v2",
"requestDate": "2024-01-19T13:04:18.447-05:00",
"responseDate": "2024-01-19T13:04:18.463-05:00",
"requestURL": "http://localhost/_alarm_serviceprovider/api/v1/alarm/
categories"
},
"rfc7807Error": null
}
```

Example error response

```
{
"payload": null,
"success": false,
```

Standard JSON response


```
"code": "500",
"messages": [],
"context": {
"version": "v2.0.0",
"requestDate": "2024-01-19T13:05:53.374-05:00",
"responseDate": "2024-01-19T13:05:53.461-05:00",
"requestURL": "http://localhost/_alarm_serviceprovider/internal/api/v1/
alarm/configuration"
},
"rfc7807Error": {
"detail": "Error getting configuration for #oops : Invalid lookup string: #oops",
"type": "about:blank",
"status": 500,
"title": "Get location configuration system error.",
"langKey": "@lang_GetLocationConfigurationDetailsError",
"invalidParamDetailList": []
}
}
```

CURL command

The following example demonstrates using the “curl” command line tool to request the alarm configuration properties of a microblock with path “#vav1/m007”.

This example is using an API key with reference name “test” and API key value of “CJAPIKEY:1:NTg4YjEyZjYtYThkZi00YzE4LTg2NmEtM2I5MDAyMDEzYzNl”

```
curl -u test:CJAPIKEY:1:NTg4YjEyZjYtYThkZi00YzE4LTg2NmEtM2I5MDAyMDEzYzNl -X GET "http://
localhost/_alarm_serviceprovider/api/v1/alarm/configuration?locId=%23vav1%2Fmb007"
```

This example is using operator name and password to make the same request.

```
curl -u someuser:password1234 -X GET "http://localhost/_alarm_serviceprovider/api/v1/
alarm/configuration?locId=%23vav1%2Fmb007"
```

Available REST APIs

The complete set of REST APIs used both internally by the product, and licensed for general use, are self-documented using Swagger. The Swagger page explains the API and often provides sample input and output.

To view the Swagger page:

1. Login into i-Vu®.
2. Open a second tab with the Swagger URL mentioned below.

   NOTES
   - The Swagger interface also allows you to try out and execute the REST APIs.
   - All APIs are presented even if the current operator does not have the required privileges, or if the product is not licensed to allow the operation.

The convention for the REST APIs is typically in the form: https://<server>/<provider>/api/<version>/<endpoint…>

- **server -** specific to the installation
- **provider** - the built-in portion of the product to service a group of APIs, and typically starts with an underscore, for example, "_alarm_serviceprovider".
- **api -** the literal "api"
- **version -** the version of the api - As new interfaces are added or existing ones are changed, the version changes. This is normally in the form "v#", for example, "v1" or "v2".
- **endpoint -** specific to each API

The swagger documentation currently does not designate what APIs are licensed for general use. The current set of licensed endpoints for machine to machine REST API calls are:

- /_system_serviceprovider/api/v2/structure/tree/{treereferencename}/nodedetails
- /_system_serviceprovider/api/v2/structure/tree/{treereferencename}/accessibleroots
- /_system_serviceprovider/api/v2/structure/tree/{treereferencename}/nodetree
- /_system_serviceprovider/api/v2/product/traits
- /_system_serviceprovider/api/v2/systemsettings/datetimeformat
- /_system_serviceprovider/api/v2/report/run - requires REPORT_QUERY (licensed with *Advanced Reports*)
- /_alarm_serviceprovider/api/v1/

Service providers

System Service Provider

The system service provider provides a set of REST APIs to interrogate the main i-Vu® geographic and network trees. The REST APIs are documented at the path https://<server>/_system_serviceprovider/dist/index.html. This include create, read, update and delete APIs for the following.

- Areas
- Equipment
- Sites
- Networks
- Devices

Other APIs support

- Attaching equipment to devices
- Retrieving the tree structure
- Manipulating source trees.
- Managing API KEYs
- Managing downloads

NOTE Most of these APIs are not license for machine to machine.

Alarm Service Provider

The alarm service provider provides a set of REST APIs to query alarm records. There are no APIs yet to configure or manage the alarm sources or alarm records. The REST APIs are documented at the path https://<server>/_alarm_serviceprovider/dist/index.html. User should be mindful that large alarm queries can affect the overall system performance. This includes APIS for the following.

- Get all alarm categories
- Get the configuration for a single alarm source
- Count alarms matching the given filter
- Count alarms by location matching the given filter
- Count alarms by location and microblocks matching the given filter
- Count the number of alarms for each possible alarm state (normal, off normal, fault)
- Return information about the incident group for a given alarm.
- Get all alarm categories for a location
- Get all alarms records matching the given filter
- Running an advanced report and retrieving the output in JSON, CSV or PDF format - requires Advanced Reports

HTTP Verbs

The REST APIs for the 4 traditional “CRUD” operations typically follow the standard pattern of:

- HTTP “GET” for “read”
- HTTP “PUT” for “update”
- HTTP “POST” for “create”
- HTTP “DELETE” for “delete"

NOTE Some APIs duplicate “query” functionality with both a “GET” or “POST” method because the query payload can be large and is it easier for the client tool to execute a POST. In this case, the operation and response should be identical for either.

Alarms

Alarm Count

The alarm query filter has a large number of fields that can be used to define and restrict the record count database query. See the *Swagger* documentation for the complete description. The alarm query is a JSON structure and follows the following rules.

- Each field is labeled by a string in double quotes.
- String values must be in double quotes.
- Boolean values are unquoted “true” or “false”.
- Numbers are not quoted.
- Date/time strings are in “local time” and are formatted according to ISO 8601 local date time. The format is “yyyy-mm-ddThh:mm:ss” or “yyyy-mm-ddThh:mm:ss.hhZ” where “T” (time) and “Z” (UTC) are the literal letters. Example "2022-12-25T05:30:00" or "2022-12-25T05:30:00.00Z".
- Do not include a time zone offset.
- Do not include a comma after the last field. This is a common mistake that results in an "HTTP 500" error.

The following examples show various alarm count queries. For this example, the alarm query record should be placed in a file called “query.json” and run with the following command.

```
curl -u restuser:secretpassword -X POST "http://localhost/
_alarm_serviceprovider/api/v1/alarm/count" -H "accept: application/json"
-H "Content-Type: application/json" -d @query.json
```

Count all alarms in the database

An empty filter selects everything

```
{ }
```

Count all alarms for a location

The following alarm query record counts all alarms for a location

```
{
"location": "#room123"
}
```

To count open alarms (needing acknowledgement or return to normal) for the location use

```
{
"location": "#room123"
"byAcknowledgePending": true,
"byReturnToNormalPending": true
}
```

To count active (have not returned to normal) alarms for the location use

```
{
```

```
"location": "#room123"
"byReturnToNormalPending": true
}
```

Count all alarms for a date range

The following query counts all alarm records for the first week of April. This counts all new alarms and “return to normal” records.

```
{
"location": "#room123"
"fromDate": "2024-04-01T00:00:00Z",
"toDate": "2024-04-07T23:59:59Z"
}
```

The following query counts all new alarm (OFF_NORMAL/FAULT) occurrences for the first week of April.

```
{
"location": "#room123"
"toStates": [ "OFF_NORMAL", "FAULT" ],
"fromDate": "2024-04-01T00:00:00Z",
"toDate": "2024-04-07T23:59:59Z"
}
```

Count all alarms for a category

The following query counts all “HVAC Critical” and “HVAC General” alarms. The alarm category references names are supplied in the “includeCategories” field.

```
{
"location": "#room123"
"toStates": [ "OFF_NORMAL", "FAULT" ],
"fromDate": "2024-04-01T00:00:00Z",
"includeCategories": [ "hvac_critical", "hvac_general" ]
}
```

Alarm Record Query

An alarm query is similar to an alarm count but instead of a single number, the query returns a list of alarm records that meet the filter criteria. It follows the following rules.

- Includes a “next” and/or “previous” alarm filter record that can be used to select the next “page” of alarms
- When querying alarm records, a “limit” must also be supplied. If the number of records selected exceed the limit, then the “next” and/or “previous” alarm filter record in the result can be use to continue paging through the alarms.
- There is a system limit of 1000 alarm records so if the supplied limit exceeds 1000 or the supplied limit is zero, then it will be restricted to 1000. That upper limit is subject to change.

For example an alarm query selecting all “hvac_critical” alarms for a location with no other filter criteria that exceeded the limit would return with a “next” field that might look like the following:


```
{
"payload": {
"alarms": [{
"alarmId": "ALM:1:60abf330104c0",
...
}
],
"next": {
"location": "#area123",
"atLocationOnly": null,
"fromDate": null,
"toDate": null,
"fromAlarm": null,
"toAlarm": null,
"fromClusive": null,
"toClusive": null,
"activeAtDate": null,
"activeAtReversed": null,
"fromReceiptDate": null,
"toReceiptDate": null,
"fromReceiptClusive": null,
"toReceiptClusive": null,
"includeCategories": ["hvac_critical"],
"excludeCategories": null,
"fromStates": null,
"toStates": null,
"byAcknowledgePending": null,
"byReturnToNormalPending": null,
"byClosed": null,
"isCritical": null,
"nextPageId": "ALM:1:60e1ec96681d0",
"previousPageId": null
},
```

```
"previous": null
},
"success": true,
"code": "200",
"messages": [],
"context": {
"version": "v1",
"requestDate": "2024-01-22T09:00:00.000-05:00",
"responseDate": "2024-01-22T09:00:00.000-05:00",
"requestURL": "http://localhost/_alarm_serviceprovider/internal/api/v1/
alarm/query"
},
"rfc7807Error": null
}
```


In this example, the “next” filter query record includes a “nextPageId” that lets the server know from where to continue the query, and all of the other query fields (with the exception of “includeCategories”) are undefined so as to match the original filter criteria.

Document revision history

Important changes to this document are listed below. Minor changes such as typographical or formatting errors are not listed.

Date	Topic	Change description	Code*
2/11/25	Service providers	Clarified System Service Providers	X-PM-TC-J-DD
8/20/24	Authentication	Changed WebCTRL to i-Vu	C-D

* For internal use only