

# Command Line API (v1)

## Command Line RESTful API

It is probably easiest to consider this API as an interaction with Inventory Queries, allowing for example:

- Ad-hoc queries to be run to return results
- New Inventory Queries to be created
- The returning of Current Query results
- Deleting queries

There is even the ability to read or alter Custom Field values for devices.

## Examples:

### Returning an Inventory Query request

#### FileWave GUI example

Imagine it is desirable to return a list of macOS devices which have a network interface name that contains 'en', the Field to be returned is the Device Name and the Main Component is 'MacOS Device'. The criteria would appear as below:

The screenshot shows the FileWave GUI for creating an inventory query. The 'Name' field is set to 'query name' and the 'Main Component' is set to 'MacOS Device'. The 'Include Archived Clients' checkbox is unchecked. The 'Criteria' tab is selected, showing a rule: 'All of these expressions must be true'. The rule consists of one criterion: 'Not Network Interface / Interface Name contains en'.

### JSON Data Example

A RESTful API query, to return the same set of results, would have a JSON structure set out as:

```
{
  "criteria": {
    "column": "interface_name",
    "component": "NetworkInterface",
    "operator": "contains",
    "qualifier": "en"
  },
  "fields": [
    {
      "column": "device_name",
      "component": "Client"
    }
  ],
  "main_component": "MacOSClient"
}
```

It can be seen that the JSON block is just mimicking the FileWave Central Inventory Query.

### Creating an Inventory Query

On the KB page explaining what an API is, the following examples were shown:

FileWave Anywhere API:

```
curl -s -H "Authorization: e2M2sssYjIwLTxxx1hMzdiLTFmYyyGIwYTdjOH0="
https://myserver.filewave.net/api/inv/api/v1/query/ \
--header "Content-Type: application/json" -X POST -d @ios16_incompatible.json
```

Command Line API:

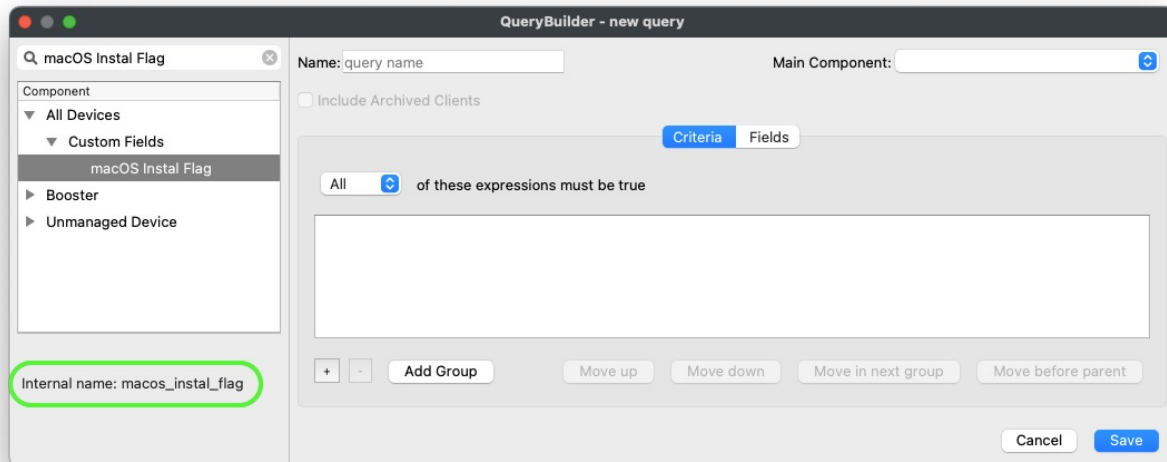
```
curl -s -H "Authorization: e2M2sssYjIwLTxxx1hMzdiLTfmyyyGIwYTdjOH0="
https://myserver.filewave.net:20445/inv/api/v1/query/ \
--header "Content-Type: application/json" -X POST -d @ios16_incompatible.json
```

These demonstrate a method of creating a new Inventory Query by providing the JSON as a file, rather than text in the actual script line, using the POST option. Also note the subtle difference in URL path.

## Interacting with Custom Fields

The first thing to appreciate, is all Inventory items have an 'Internal Name'. This is unique to each item and of course Custom Fields also therefore have their own Internal Name.

Internal Names may be viewed in FileWave Central when in the Inventory Query viewer. Select any item in the left hand sidebar and the bottom of the window will report the Internal Name.



## Reading Values

Perhaps it would be desirable to read the Custom Field value of the above 'macOS Instal Flag' Custom Field for a device, for example, based upon its serial number. The JSON data block might look something like:

```
{
  "criteria":
  {
    "column":"serial_number",
    "component":"Client",
    "operator":"is",
    "qualifier":"\"$serial_number\""
  },
  "fields":
  [
    {
      "column":"macos_instal_flag",
      "component":"CustomFields"
    }
  ],
  "main_component":"Client"
}
```

Using this method, it is possible to provide the Serial Number as a variable value; as such the same script could be ran on all devices, without the need to edit the script. Note the internal name used for the Custom Field as well as the single quotes around the \$serial\_number variable.

## Editing Values

As an alteration of a current value, the command would then use the PATCH option. In this instance, the Custom Field to alter has an 'internal name' of 'admin\_test', is a Boolean entry and the JSON data block could look like:

```
{
  "CustomFields":
  {
    "admin_test":
```

```
{
  "exitCode":null,
  "status":0,
  "updateTime":"'${current_time}'",
  "value":true
}
```

When Custom Fields are updated, a time stamp of when the update occurred is shown. When altering the value from the Command Line API, the time should be supplied, along with the value and a couple of other key/value pairs. There should be no reason to provide an other value than those shown for 'exitCode' and 'status', when adjusting using the API.

Date is important and should be of the format "2018-09-10T15:48:08Z":

macOS example:

```
current_time=$(date -u +"%FT%TZ")
```

Windows PowerShell:

```
$current_time = $(Get-Date -UFormat "%Y-%m-%d %R:%S")
```

The code would look something like the following, where \$query would be set as the content of the JSON:

macOS shell

```
curl -s -H "Authorization: $auth" \
  https://$server_dns:20445/inv/api/v1/query_result/ \
  -X PATCH \
  -data $query \
  -H "Content-Type: application/json"
```

Windows PowerShell

```
$header = @{Authorization="$auth"}

Invoke-RestMethod -Method PATCH \
  -Headers $header \
  -ContentType application/json \
  -Uri https://$server_dns:20445/inv/api/v1/query_result/ \
  -Body $query
```



As with any Inventory Query, the criteria can be based upon any inventory item. This makes the Command Line API very powerful.

🔄Revision #9

★Created 4 July 2023 14:44:10 by Sean Holden

✍Updated 21 February 2024 09:25:09 by Sean Holden