RESTful API

The purpose of this guides is two fold:

First, to provide an introduction to those unfamiliar with using the FileWave API

Second, to be a reference for commands that can be used within the API

Getting Started

Purpose of an API

The RESTful API is code built into FileWave server (starting with version 6.x) providing the capability of two-way communication between the FileWave Postgres database and other external systems. Using the calls in the RESTful API, a FileWave administrator can generate queries, exchange information, and more efficiently integrate inventory data. The RESTful API is designed so that queries and tools created using the current version API will continue to function as future versions of the FileWave server and admin apps are introduced.

Many institutions using FileWave for inventory management of their various computing devices also have existing databases for tracking systems. These DB's could range from simple, in-house SQL databases designed to keep track of institutional assets to a full-blown, commercial data engine such as SCCM. The RESTful API contains the mechanisms to integrate those databases into FileWave through an authentication system, followed by a standardized set of queries using the JSON (JavaScript Object Notation - https://json.org) format.

Possible uses could be integrating existing SCCM data into FileWave queries; sending Fileset information from FileWave back to your help desk database, such as Invgate; or integrating purchase order system data into the application license data stored in FileWave.

The RESTful API provides you, the FileWave administrator, with the ability to expand the depth of information about your managed systems. Imagine being able to integrate years of detailed inventory stored in your site's commercial database with the ease of the FileWave Admin's Inventory queries, or being able to create a simple front end application that alerts you to sudden changes in the software licenses you have enabled across your entire company or campus. The RESTful API provides this kind of expansion of your ability to provide in-depth asset and license management on an as-needed basis.

The Token is the Key

For security reasons, we will need a token to access the RESTful API. Before version 12.9 there was only one token for inventory (Admin Preferences Inventory "Shared Key") that had total read and write access to all data. Starting in 12.9 there can be many application tokens, each with their own rights.

> For more information on the Application Tokens see the Manual page: Managing FileWave Administrators (Application Tokens)

We will need the base64 version of the token for the rest of the activity.

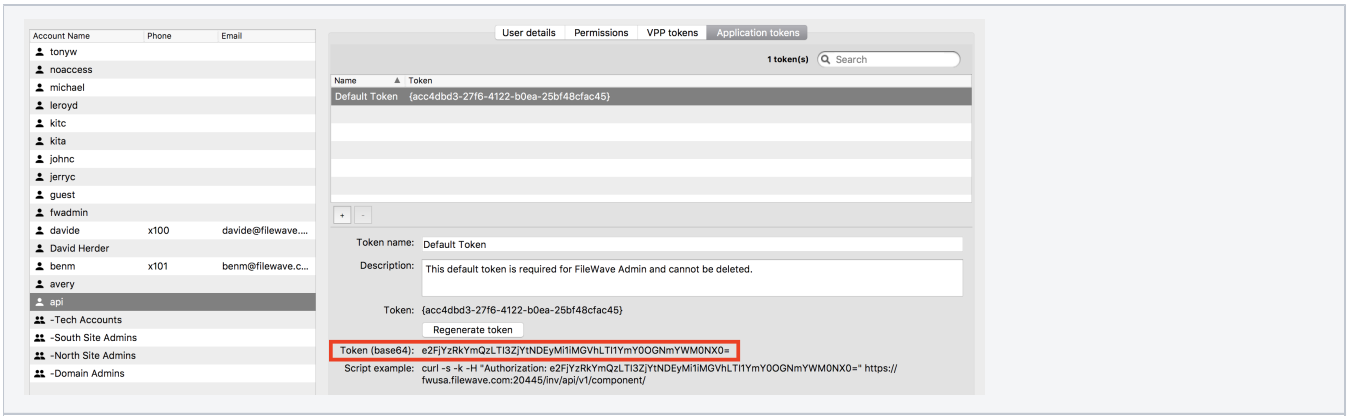


Figure 1.1 - Manage Administrators

Select everything (including the = at the end)

Application Token

```
e2FjYzRkYmQzLTI3ZjYtNDEyMi1iMGVhLTI1YmY0OGNmYWM0NX0=
```

Breaking down the JSON/Results

When you send (AKA POST) or receive (AKA GET) inventory information it will be sent/received in the form of JSON.

JSON is broken into keys and values. One key benefit of JSON over CSV is the ability to do lists inside of lists.

For example, doing a GET to look at a list of all the inventory queries in your server, and then the results of one of those queries would look like this:

GET all queries

Get all Queries

```
curl -s -k -H "Authorization: e2FjYzRkYmQzLTI3ZjYtNDEyMi1iMGVhLTI1YmY0OGNmYWM0NX0=" https://myserver.company.org:20445/inv/api/v1/query/ | python -mjson.tool
```

```
[
    {
        "id": 1,
        "name": "All Windows",
        "favorite": true,
        "group": 1,
        "version": 1
    },
    {
        "id": 2,
        "name": "Mac OS X 10.7-10.11",
        "favorite": false,
        "group": 1,
        "version": 5
    },
...
    {
        "id": 103,
        "name": "All Computers to retire",
        "favorite": false,
        "group": 3,
        "version": 2
    }
]
```

Key	Value	Description
id	number	The unique number for the query. To be used as reference
name	txt	The name given to the query
favorite	true /false	The state if the query has a check next to it or not to show in the sidebar of admin
group	number	The group number given. built-in queries – for example – would be in the "Sample Queries" group, which is group 1. If the user made new groups
version	number	The version for the query. How many times has the query been altered and saved, starting with 1

GET the Query Information

The query information, not the results of the query.

Get Query

```
curl -s -k -H "Authorization: e2FjYzRkYmQzLTI3ZjYtNDEyMi1iMGVhLTI1YmY0OGNmYWM0NX0=" https://myserver.company.org:20445/inv/api/v1/query/1 | python -mjson.tool
```

```
{
    "criteria": {
        "expressions": [
            {
                "column": "type",
                "component": "OperatingSystem",
                "operator": "=",
                "qualifier": "WIN"
            }
        ],
        "logic": "all"
    },
    "favorite": true,
    "fields": [
        {
            "column": "device_name",
            "component": "Client"
        },
        {
            "column": "filewave_client_name",
            "component": "Client"
        },
        {
            "column": "name",
            "component": "OperatingSystem"
        },
        {
            "column": "version",
            "component": "OperatingSystem"
        },
        {
            "column": "build",
            "component": "OperatingSystem"
        },
        {
            "column": "edition",
            "component": "OperatingSystem"
        }
    ],
    "main_component": "Client",
    "name": "All Windows",
    "id": 1,
    "version": 1,
    "group": 1
}
```


Key	Value	Description
`criteria`	`array`	Where all the criteria for the query is saved

<table>
<tr><td></td><td>Expressions - Repeated for each Criteria
<table>
<tr><th>Key</th><th>Value</th><th>Description</th></tr>
<tr><td>column</td><td>(too many to list. See "Show all component ")</td><td>The specific component to filter by (Figure 1.2 #1)</td></tr>
<tr><td>component</td><td>(too many to list. See "Show all component ")</td><td>The component group where the item was found (Figure 1.2 #2)</td></tr>
<tr><td>operator</td><td>(too many to list. See "Show all Fields")</td><td>(Figure 1.2 #3)</td></tr>
<tr><td>qualifier</td><td>(too many to list. See "Show all component")</td><td>(Figure 1.2 #4)</td></tr>
</table>
Logic
<table>
<tr><th>Key</th><th>Value</th><th>Description</th></tr>
<tr><td>logic</td><td>all / one / none</td><td>(Figure 1.2 #5)</td></tr>
</table></td></tr>
<tr><td>favorite</td><td>true /false</td><td>The state if the query has a check next to it or not to show in the sidebar of admin</td></tr>
<tr><td>fields</td><td>array</td><td>Where how the results are shown</td></tr>
<tr><td></td><td>Repeated for each field shown
<table>
<tr><th>Key</th><th>Value</th><th>Description</th></tr>
<tr><td>column</td><td>(too many to list. See "Show all component")</td><td>The specific component to show</td></tr>
<tr><td>component</td><td>(too many to list. See "Show all component")</td><td>The component group where the item was found</td></tr>
</table></td></tr>
<tr><td>main_component</td><td></td><td>(Figure 1.2 #6)</td></tr>
<tr><td>name</td><td>txt</td><td>The name given to the query</td></tr>
<tr><td>id</td><td>number</td><td>The unique number for the query. To be used as reference</td></tr>
<tr><td>version</td><td>number</td><td>The version for the query. How many times has the query been altered and saved, starting with 1</td></tr>
<tr><td>group</td><td>number</td><td>The group number given. built-in queries – for example – would be in the "Sample Queries" group, which is group 1. If the user made new groups</td></tr>
</table>

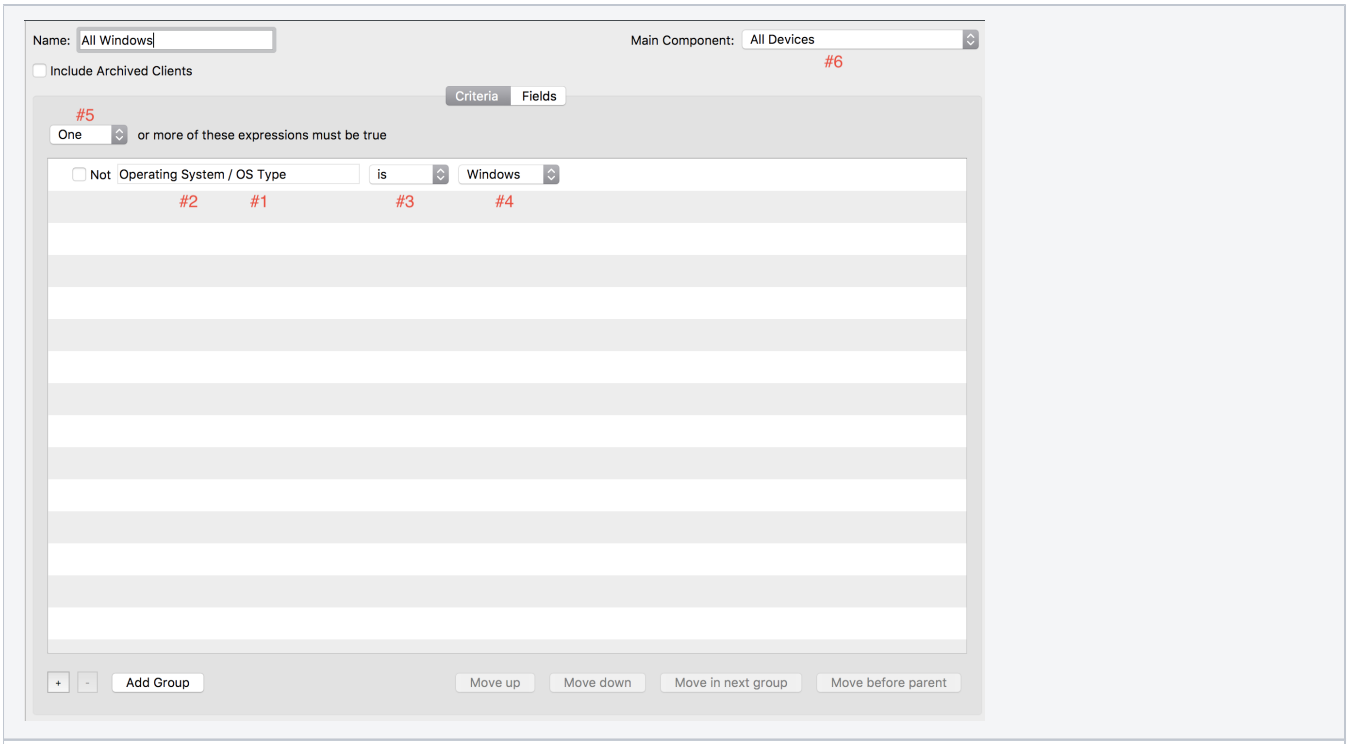


Figure 1.2 - Query Builder Criteria

GET the Query Results

Get Query Results

```
curl -s -k -H "Authorization: e2FjYzRkYmQzLTI3ZjYtNDEyMi1iMGVhLTI1YmY0OGNmYWM0NX0=" https://myserver.company.org:20445/inv/api/v1/query_result/1 | python -mjson.tool
```

```
{
    "total_results": 13,
    "filter_results": 13,
    "offset": 0,
    "values": [
        [
            "FW-BLUE-02",
            "FW-Blue-02",
            "Windows 10.0",
            "10.0.0",
            "10240",
            "Microsoft Windows 10 Home"
        ],
        [
            "LAPTOP-C6LLFGH6",
            "FH-History3",
            "Windows 10.0",
            "10.0.0",
            "14393",
            "Microsoft Windows 10 Home"
        ],
...
    ],
    "version": 3
}
```

Key	Value	Description

`total_results`	`number`	Total count of results
`filter_results`	`number`	
`offset`	`number`	
`values`	array	The results. Repeated for each result. Items depends on what your specified in the fields
`version`	`number`	The version for the query. How many times has the query been altered and saved, starting with 1

Useful Tools

Converting to/from base64

Website to encode and decode base64

- https://www.base64encode.org/ - Encode
- https://www.base64decode.org/ - Decode

Using Python

Python

```
#!/usr/bin/env python
import base64
import sys
print "Authorization:", base64.encodestring(sys.argv[1])
```

Save as base64encode.py

Use by:

```
./base64encode.py {780756eb-4cbd-455f-aaa7-d49db12de9d0}
```

JSON

Verify JSON Formatting:

- https://jsonlint.com/

API Application

- Postman https://www.getpostman.com/ (macOS, Windows, and Linux)

Browser Extensions

- Mod-Header https://mod-header.appspot.com/ Will allow you to use the Google Chrome Browser to view and interact with the FileWave API

Commands

Remember: All URLs start with

`https://myserver.company.org:20445/inv/api/v1/`

Must include the authorization header

Below are the options for use with it.

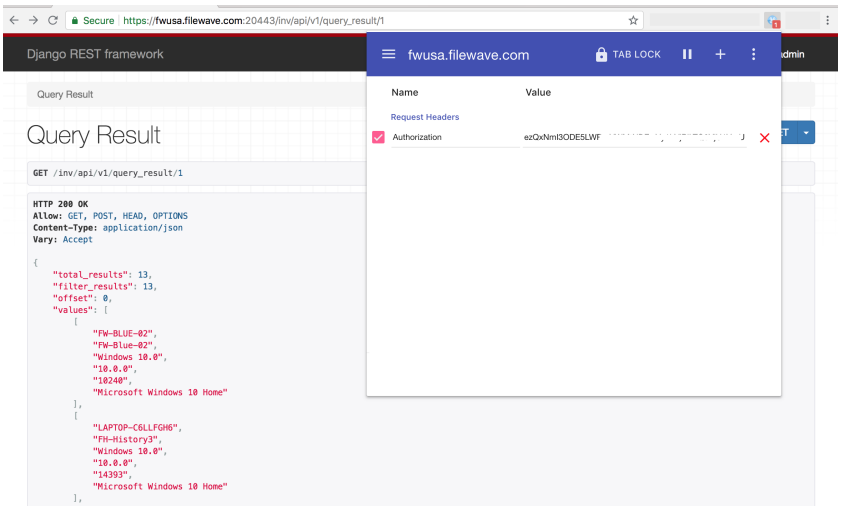
URLs

URL	Use	Options
Inventory		
`query`	Show all queries	`GET POST`
`query/#`	Show information on a single query Where # is the query ID	`GET PUT DELETE`
`query_group/`	Show all query group	`GET POST`
`query_group/#`	Detail information on a single group Where # is the group ID	`GET PUT DELETE`
`query_result/#`	Show the results of one query Where # is the query ID	`GET POST`
`query_count`		`POST`
`component`	Show all component options on your instance	`GET`
`field_type`	Show all fields on your instance	`GET`
License		
`license_definition`	Show all query	`GET`
`license_definition/#`	Show information on a single license Where # is the license ID	`GET`
Custom Fields		
`custom_field/`	Show all custom fields	
`custom_field/get_association`		`POST`
`custom_field/set_association`		`POST`
`custom_field/upload`		`POST`
`custom_field/usages/<Field_Name>`	Where <Field_Name> is the Internal Name (E.G "battery_cycle_count")	`GET`
`custom_field/values/`		`POST`
`custom_field/edit/`		`POST`

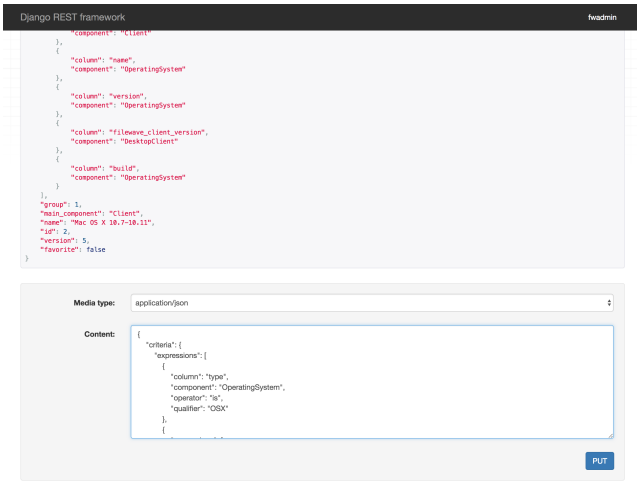
Examples

Using a browser extension

Using Mod-Header (see tools section), you can make Chrome a RESTful API browser by taking advantage of the FileWave Django Framework. Allowing you to look at the URLs and how things show up by telling the browser to send a token as an "Authorization" header each place you go to.



Even if you go to a url that is typically a POST url, it provides a box below you can post with



Using the curl command

Viewing all available queries (GET)

```
curl -s -k -H "Authorization: e2FjYzRkYmQzLTI3ZjYtNDEyMi1iMGVhLTI1YmY0OGNmYWM0NX0=" https://myserver.company.org:20445/inv/api/v1/query/ | python -mjson.tool
```

Posting a new query (POST)

```
curl -s -k -H "Authorization: e2FjYzRkYmQzLTI3ZjYtNDEyMi1iMGVhLTI1YmY0OGNmYWM0NX0=" --header "Content-Type: application/json" -X POST -d @<path/name of new query.json> https://myserver.company.org:20445/inv/api/v1/query/
```

Removing a query (DELETE)

```
curl -s -k -H "Authorization: e2FjYzRkYmQzLTI3ZjYtNDEyMi1iMGVhLTI1YmY0OGNmYWM0NX0=" -X DELETE https://myserver.company.org:20445/inv/api/v1/query/<id#>
```

For more curl help, see: Using the RESTful API to limit, sort, and offset values returned

Using PHP

Saved as a php file (like inv.php), update the url and auth code, then place the file on a web server where PHP has been enabled. This creates a webpage that is a view only version of your inventory. People can go to the URL for the query and hit refresh as many times as they like, always seeing the latest information in inventory All without having to hassle IT for the latest data.

The output of the query results isn't fancy, but this is to illustrate what can be done.

PHP Inventory viewer

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd"><html xmlns="http://www.w3.org/1999/xhtml">
<html>
<?php
$baseurl="myserver.company.org";
$port="20443";
$authcode="e2FjYzRkYmQzLTI3ZjYtNDEyMi1iMGVhLTI1YmY0OGNmYWM0NX0=";


ini_set('display_errors', 'On');
### do not edit below ###
if (!isset($_GET["qid"])){
        $url = "https://".$baseurl.":".$port."/inv/api/v1/query/";
} else {
        $url = "https://".$baseurl.":".$port."/inv/api/v1/query_result/".$_GET["qid"];
}
//  Initiate curl
$ch = curl_init();
// Set the url
curl_setopt($ch, CURLOPT_URL,$url);
// Disable SSL verification
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
curl_setopt($ch, CURLOPT_SSLVERSION, 1);
// Will return the response, if false it print the response
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
//authenticate
curl_setopt($ch, CURLOPT_HTTPHEADER,array('Authorization:<'.$authcode.'>='));
// Display errors if any
if (curl_errno($ch)) {
   print curl_error($ch);
}
// Execute
$result=curl_exec($ch);
if (curl_errno($ch)) {
   print curl_error($ch);
}
$output=json_decode($result, true);
curl_close($ch);

//create function for looping unknown dimensional array
function printAll($a) {
  if (!is_array($a)) {
    echo $a, ' <br/>';
    return;
  }
echo "<br/>";
  foreach($a as $v) {
    printAll($v);
  }
}

// Start html Page
```

```
echo '<head>'
.'<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />'
.'<title>'.$baseurl.'Inventory Page</title>'
.'</head>'
.'<body>'
.'<style type="text/css">'
."body {font-family:'Helvetica Neue Light', 'Helvetica Neue', Helvetica, Arial, 'Lucida Grande', sans-serif;}"
.'h1,h2,h3,h4,{font-weight:100;}'
.'div {padding:10px; color:#fff; background:#333;}'
.'div.output{border:1px solid rgba(0,0,0,0.1); background: rgba(0,0,0,0.03); color:#555;-webkit-border-radius:
3px;border-radius:3px;margin:15px 25px; padding:10px;}'
.'tr:nth-child(even) {background: rgba(255,255,255,0.85);}'
.'tr:nth-child(odd) {background: rgba(0,0,0,0.05);}'
.'a, a:hover {color:#ce1300; text-decoration:none;}'
.'</style>'
.'<div><h1><img src="https://www.filewave.com/images/filewave-1024x788.png" height="60" />'.$baseurl.'
Inventory</h1></div>'
.'<br/>';

// Default homepage
if (!isset($_GET["qid"])){
        echo "<table>"
        ."<thead>"
        ."<tr>"
        ."<th>&hearts;</th>"
        ."<th>Query Name</th>"
        ."<th>Query ID</th>"
        ."</tr>"
        ."</thead>"
        ."<tbody>";
        foreach ($output as &$value) {
                if ($value['favorite'] == true) { $fav="&hearts;";} elseif ($value['favorite'] == false)
{$fav=" ";}
                echo "<tr><td>".$fav."</td><td>".$value['name']."</td><td><a href='".$_SERVER['PHP_SELF']."?
qid=".$value['id']."&n=".$value['name']."'>".$value['id']."</a></td></tr>";
        }
        echo"</tbody></table>";
}
//If an individual query has been selected
elseif (isset($_GET["qid"],$_GET["n"]))  {
        echo "<h3>Home &gt; Query: "
        .$_GET["n"]
        ."</h3><hr/>"
        ."<strong>Total Results: </strong>"
        .$output['total_results']
        ."<br/>"
        .'<strong>First Column results: </strong>';

        foreach ($output['values'] as &$value) {
                echo $value['0'].' <strong>   |   </strong>';
        }
echo "<br/>"
."<strong> All Results: </strong><br/><div class='output'>";
printAll($output['values']);
echo "</div>";
#var_dump($result);
}
else {
        echo "<h1 style='color:#ff0000;'> An error has occurred</h1> Parhaps you used a bookmark and the URL
has changed";
}
?>
<hr/>
<center><font style=" font-size:9px;"><a href="http://filewave.com" target="_blank">&copy; BenM@ FileWave</a><
/font></center>
</body>
</html>
```