# Accessing DR Hub using Curl

## Contents

## *About this Document*

The purpose of this document is to provide examples of the web service functionality available for retrieving and action pending Tasks.

The information is provided for learning and demonstration only and does not represent actual tasks or other factual event or account information.

The below examples are for demonstration only. Additional coding and integration may be required to me your companies needs.

Curl is viewed as a quick and easy way to make web service calls but it is not without its own set of traps and challenges. Curl is syntactically particular and small changes can have big impacts on your script. Once you have established a few good scripts and basic practices for implementing, re-use is strait forward.

Curl is available for Windows or Linux. Please check with your Tech Support for details.

## Release History

| Release | Date | Description |
|---|---|---|
| 1.0.0 | December 19, 2016 | Initial Document Created |
| 1.0.1 | April 5, 2023 | Added format for retrieving a Token using a CERT. |

## *Securing a Token using Curl*

This request retrieves a token from the PJM authentication server. In the below example the response is copied to the token variable and then the actual token is parsed form the json response.

With the token stored in the variable, it can then be re-used for subsequent requests. Note the request for a token is a POST request. The user information is POSTED which yields a json response to include a new token. If the request fails, the response may include a large HTML response from the authentication server with no decipherable information.

```
echo "Retrieve a new Token"
```

(if needed to create the .pem file from the .p12 file) Open a bash window and run…

```
openssl pkcs12 -in glenn_test.p12 -out glenn_test.key.pem -nocerts –nodes
```

Again from the bash prompt.

```
token=$(curl -v -k --request POST --key glenn_test.key.pem --cert 'glenn_test.crt:cert_password' --header "X-OpenAM-Username: drhub_username" --header 'X-OpenAM-Password: drhub_password' 'https://ssotrain.pjm.com/access/authenticate/pjmauthcert')

echo "Parse the token into a variable to be used later in the script"

tokenParse=`echo "$token" | python -m json.tool | sed -n -e '/"tokenId":/ s/^.*"\(.*\)".*/\1/p'`

#Echo the token value to the screen

echo $tokenParse
```

## *Perform a REST GET Request using Curl*

This request retrieves an existing Location using Curl's GET option. Curl is intuitive enough to know a GET is being requested, but the keyword helps differentiate the script from a POST. This request does not save the response but echos it back to the user's screen. Additional Curl options exist for saving responses. Use curll –help to retrieve a list.

Note the use of the token variable as it is passed as an authentication Cookie for the request.

You may also be required to include the CERT information as shown in the above example to retrieve a token.

```
echo "Get Location"
```

```
curl -H "Accept: application/xml" -H "Content-Type: application/xml" -H
"Cookie:pjmauthtrain=$tokenParse" -X GET
https://drhubtrain.pjm.com/drhub/rest/secure/download/location/65855
```

## *Perform a REST POST Request using Curl*

This request creates a new Location using Curl's POST option. As above the POST is inferred in the request. Curl is very fussy about the source and target path for requests. Be paths match your environment and path structure exactly. Check that all files to be uploaded have the appropriate permissions so that Curl can access them for reading.

The reason we upload the source file data and  use a file name on the end of the request is that DR Hub is expecting the file name as a parameter to the data which is marshalled over HTTP.

```
echo "Create Location"

curl -v -H "Content-Type: application/xml" -X PUT --data-binary "@c:/cygwin64/drhub/newLocation.xml"
-H "Cookie:pjmauthtrain=$tokenParse"
https://drhubtrain.pjm.com/drhub/rest/secure/upload/location/create/newLocation.xml
```

## *Perform a REST Request using PHP*

This examples uses the GuzzleHttp utility to perform both the get Token and PUT request via the PHP libaries. When testing be sure to have the GuzzleHttp client installed and setup according to your environment.

```
<?php
use GuzzleHttp\Client;

$client = new Client();
$tokenId = getToken($client);

$xml = '/path/to/your/locations/create/name-of-your-file.xml';
$url =
'https://drhubtrain.pjm.com/drhub/rest/secure/upload/location/create/name-of-
your-file.xml'; // this filename should match filename above
```

```
$response = $client->request('PUT',
    $url,
    [
        'headers' => [
            'Cookie' => "pjmauthtrain=".$tokenId,
            'Content-Type' => 'application/xml'
        ],
        'body' => fopen($xml,'r'),
        'debug' => false // set this to true to see output
    ]
);

if($response->getStatusCode() == 200)
{
    echo $response->getBody();

file_put_contents("/path/to/your/saved/direction/location_created_".date("Y-
m-d-his").".xml",$response->getBody()); // optional to save the xml file
locally
} else {
    echo $response->getStatusCode();
}

function getToken($client)
{
    try {
        $response = $client->request('POST',
            'https://ssotrain.pjm.com/access/authenticate',
            [
                'headers' => [
                    'X-OpenAM-Username' => 'your-username',
                    'X-OpenAM-Password' => 'your-password',
                ],
            ]);
        $body     = json_decode($response->getBody());
        $tokenId    = $body->tokenId;
        return $tokenId;
    } catch(Exception $e) {
        return $e->getMessage();
    }
}
```

End of Document