Section VII: Reference Section

TouchCut REST API

Note: Touchcut7 also supports an internal webserver: Touchut Web Server 458

This reference outlines using the REST API with Kinetic TouchCut7 machines. The same information also applies to older Kinetic TouchCut6 machines using TouchCut6.9.664 or later.

The TouchCut REST API provides data access and control to the TouchCut Web Interface. It also provides support for third party application to gain external machine status monitoring. The API is currently in version 1 and is still subject to change. However it is most likely that new functionality will be added rather than existing functionality modified (unless otherwise stated).

It can be used to monitor the machine with a high degree of granularity. Specific aspects can be queried/polled on a regular basis and the results collated by external software.

The REST API requires HTTPS access for remote access and authentication requests. HTTP access can be used for localhost access for non authentication requests. TouchCut creates its own self signed certificate for the HTTPS access. The API requires authentication via a login from a TouchCut user (with a valid web access security privilege) or an API Key. When using a TouchCut user, a successful login will issues a JSON Web Token (JWT) that can then be supplied via the "Authorization: Bearer " header key. If using an API Key then it can be supplied via the "X-TouchCut-API-Key: " header key. TouchCut uses the JWT and API Keys to look up the security privileges from their matching security group.

Note: we also allow the JWT to be accessed via a "__Secure-jwt" cookie.

The base URL for the Rest API is: https://<machine IP/name>/rest/v1/

An HTML (webpage) interface is available as well providing a subset of the Rest API functionality direct to any browser, and you can navigate to this by typing https://<machine IP/name> in your browser.

Using the TouchCut REST API

The REST API uses communication protocols similar to HTTP/HTTPS. As such, the REST server is implemented using a web server and can be communicated with using a basic network socked or a web browser. The majority of the TouchCut REST API requires authentication. The authentication tokens (JSON Web Token 457 (JWT) or API Key 457) are required to be passed as an HTTP header field (or in a cookie in the case of the JWT), which is not available from a web browser. So to quickly and easily test the TouchCut REST API you can use a REST API Test Client such as the Advanced REST Client application (https://install.advancedrestclient.com/install).

A REST API Test Client application allows you to:

- Enter the HTTP method (e.g.: GET, POST)
- Enter the URL (Uniform Resource Locator), i.e. the web address
- Enter query parameters
- Enter headers
- Enter body data

The Test Client application will then allow you to send the request and receive the reply from the TouchCut server. When you send the request, the test client application performs the following steps for you:

- Makes a network connection to the TouchCut Server
- Negotiates a secure connection if required (for an HTTPS connection)
- Forms a request (from the HTTP method, URL, parameters, headers and body data) and sends it
- Receives the reply from the TouchCut Server

- Closes the network connection

If the connection was successful, the response is then parsed and displayed. The raw response headers and data can also be viewed. When connecting to the TouchCut REST API from your own application the above steps need to be performed manually but most programming languages provide networking library functions that can wrap HTTP/HTTPS calls for you.

Note: The TouchCut REST API requires secure connections via the HTTPS protocol for all requests that involve communicating passwords or other security related information. Other requests may allow a non-secure connection via the HTTP protocol (i.e. without SSL) to be used when making a localhost connection. For these requests, where authentication is still required, a JSON Web Token (JWT) or API Key is still required. JSON Web Token's will only be returned from authentication requests via a secure connection.

For secure connections the TouchCut Server generates a self signed certificate by default. This allows an encrypted connection but does not protect against the Man-In-The-Middle attack. When using the self signed certificate ensure that you do not perform SSL certificate verification. If a more secure connection is required, and you wish to perform SSL certificate verification, you can obtain a certificate from a certificate authority. Contact your service provider for information on how configure TouchCut to use a signed certificate.

Enabling the REST API

In order for TouchCut's web server to function, the following files must be present in the *C:\TouchCut7* folder (or *C:\TouchCut6* folder). The /dll and .pem files are created automatically when the WimServer.ini file exists and a TouchCut update is applied.

- *libcrypto-1_1.dll*
- *libssl-1_1.dll*
- *ssl_cacertificates.pem*
- *WimServer.ini*
- msvcr120.dll is no longer required. That was for an older SSL library. (Double checked and confirmed by Software)

To enable the REST API:

1. First, manually create the **WimServer.ini** file in the TouchCut folder on the machine. **WimServer.ini** should contain this text:
```
[WimServer]
Enabled=1
Port=80
Allow IPv4=1
Allow IPv6=0
Web Directory=
```
2. Once the WimServer.ini file exists in the TouchCut folder, performing a TouchCut update (even if it's the same version as current) will auto extract the security .dll's and .pem file.
3. After the first connection, TouchCut will generate a self signed certificate and append these details in the WimServer.ini
4. **[PrimeCutNE only]** If the machines controller has the MIME Encoding settings enabled in the PrimeCutNE nesting software, additional information including part completion feedback can be accessed via the REST API. See Edit Mode and PrimeCutNE 151. When MIME Encoding is enabled, PrimeCutNE can generate a multipart NC file encoded with MIME, which includes not just the "G codes" but also a nestmap encoded as a DXF.

Kinetic techs and software developers: The dlls above may be found in *C:_SVN\TRUNK_XE\Library\Third party\OpenSSL\Win32\bin\*

Managing REST API Keys

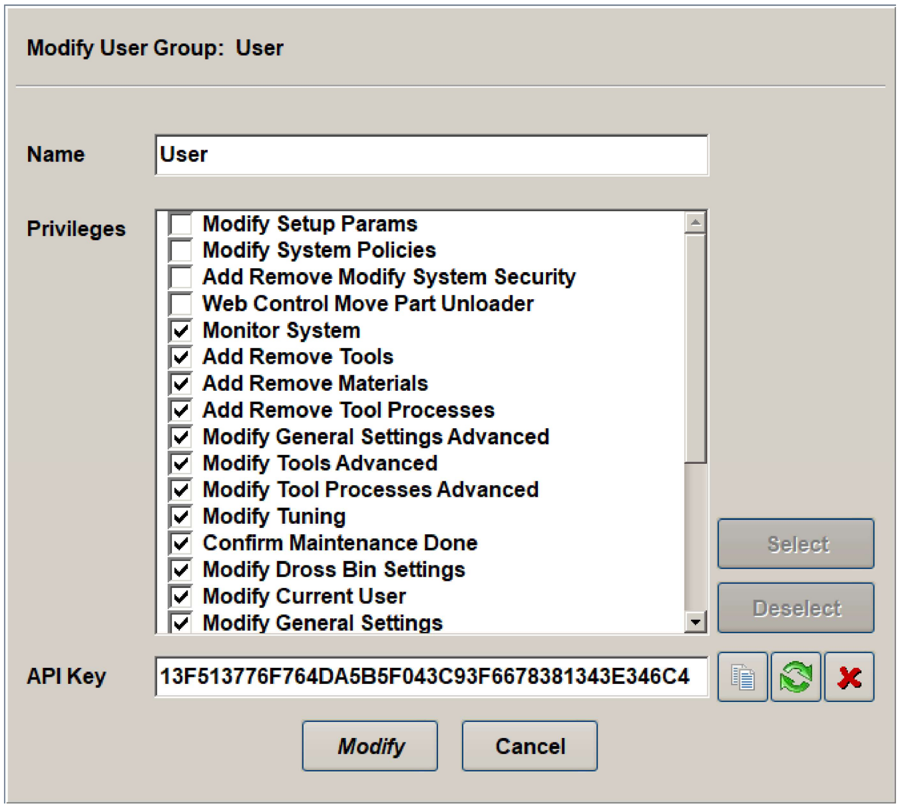
TouchCut allows authentication to the REST API 456 via API Keys 457 to allow third party applications to access and control TouchCut. API Keys can be generated and assigned to user groups, where the particular user group provides the security privileges for the API Key.

If you are using an API Key to provide authentication for accessing the REST API requests then it must be supplied via the "X-TouchCut-API-Key: " header key. It should not be used in conjunction with a JSON Web Token 457 (JWT), but if it is the JWT privileges will take precedence. The JWT can be provided via the "Authorization: Bearer " header key or via a "__Secure-jwt" cookie.

To TouchCut user with "Add Remove Modify System Security" privileges must be logged in to manage API Keys. To manage an API Key for a user group:

- Go to the Advanced, Users tab.
- Select a group to manage and then select the Modify button.
- At the bottom of the User Group edit page you can Copy, Refresh or Clear the API Key for the group
- Select the Modify button to save the changes

e.g.


Modify User Group: User

Name: User

Privileges:
- [ ] Modify Setup Params
- [ ] Modify System Policies
- [ ] Add Remove Modify System Security
- [ ] Web Control Move Part Unloader
- [✓] Monitor System
- [✓] Add Remove Tools
- [✓] Add Remove Materials
- [✓] Add Remove Tool Processes
- [✓] Modify General Settings Advanced
- [✓] Modify Tools Advanced
- [✓] Modify Tool Processes Advanced
- [✓] Modify Tuning
- [✓] Confirm Maintenance Done
- [✓] Modify Dross Bin Settings
- [✓] Modify Current User
- [✓] Modify General Settings

API Key: 13F513776F764DA5B5F043C93F6678381343E346C4

Figure 201: Managing API keys for the group

By default there are no API Keys allocated to the user groups. Select the Refresh button above to allocate a new API Key. It can then be copied to the clipboard with the Copy button or cleared with the Delete button.

Unlike JWT's API Keys do not expire. To invalidate an existing API Key use the above dialog to allocate a new API Key for the group, using the Refresh button, or clear the API Key using the Delete button. The old API Key will no longer work once you select the Modify button to confirm the change.

Some REST API requests are not available via API Keys, such as “rest/v1/auth/change-password” which requires an authenticated user to be logged in (via a JWT token).

REST API Connection Examples

The Advanced REST Client application provides example code for various platforms for your REST API request. You can use the application to test your API calls and then copy the example code into your application as a starting point.

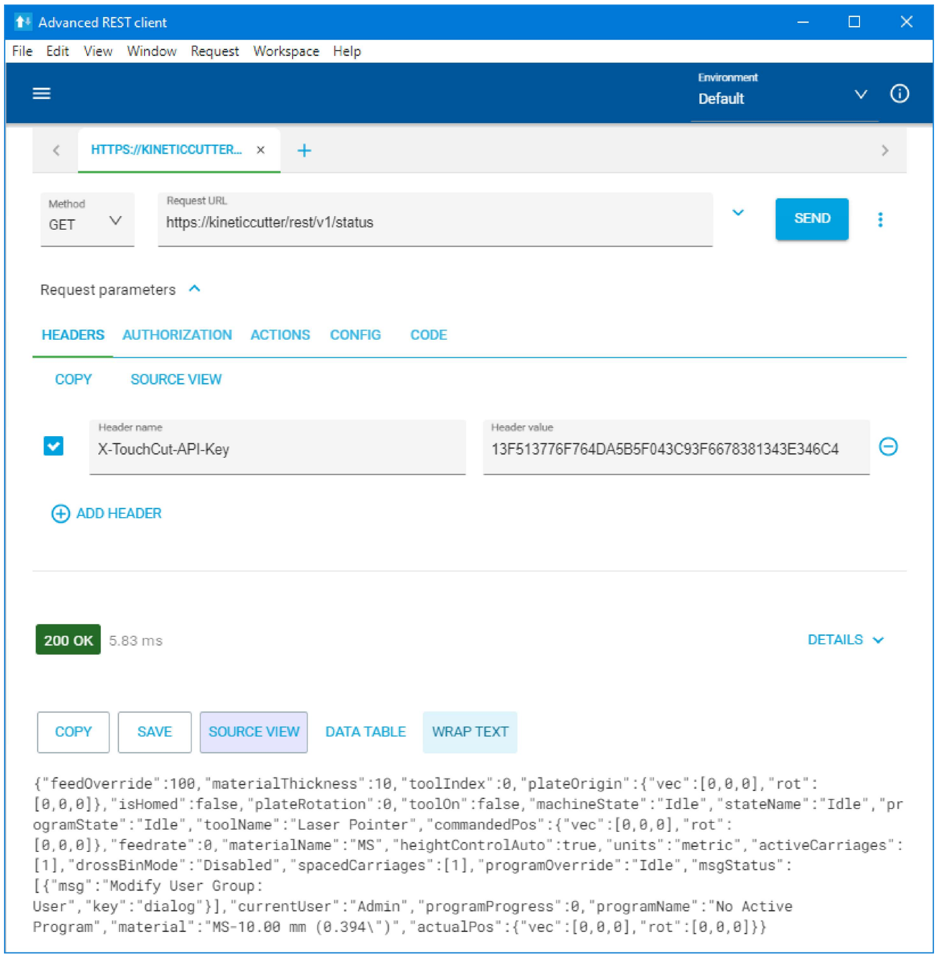
Below are a couple of examples using an external call to the cUrl utility and using the Curl library in the C programming language.

Note: JSON objects are an unordered set of name/value pairs. This means the order of the returned JSON objects name/value pairs can not be relied on. However array items are ordered.

Advanced REST Client Example

You can use a REST client application to test the TouchCut REST API. This example uses the Advanced REST Client (ARC) application available from https://install.advancedrestclient.com/install.

The following example shows a request to the status resource.


Advanced REST client
File Edit View Window Request Workspace Help
Environment
Default
HTTPS://KINETICCUTTER...
SEND
Request parameters
HEADERS AUTHORIZATION ACTIONS CONFIG CODE
COPY SOURCE VIEW
ADD HEADER
DETAILS
COPY SAVE SOURCE VIEW DATA TABLE WRAP TEXT

Method: GET
Request URL: https://kineticcutter/rest/v1/status

Header name: X-TouchCut-API-Key
Header value: 13F513776F764DA5B5F043C93F6678381343E346C4

200 OK 5.83 ms

```
{"feedOverride":100,"materialThickness":10,"toolIndex":0,"plateOrigin":{"vec":[0,0,0],"rot":
[0,0,0]},"isHomed":false,"plateRotation":0,"toolOn":false,"machineState":"Idle","stateName":"Idle","pr
ogramState":"Idle","toolName":"Laser Pointer","commandedPos":{"vec":[0,0,0],"rot":
[0,0,0]},"feedrate":0,"materialName":"MS","heightControlAuto":true,"units":"metric","activeCarriages":
[1],"drossBinMode":"Disabled","spacedCarriages":[1],"programOverride":"Idle","msgStatus":
[{"msg":"Modify User Group:
User","key":"dialog"}],"currentUser":"Admin","programProgress":0,"programName":"No Active
Program","material":"MS-10.00 mm (0.394\")","actualPos":{"vec":[0,0,0],"rot":[0,0,0]}}
```

Figure 202: REST client application

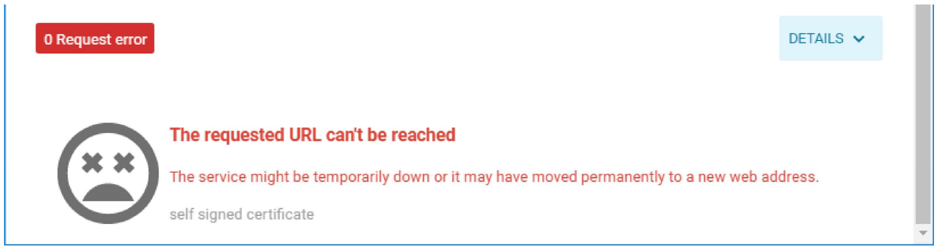
The above request is made up of the following components:

- **Method** - GET
- **Protocol** - https

- **Hostname** - kineticcutter *(this can also be an IP address, however if the controller connects to the network via DHCP care should be taken to ensure it is configured to use a reserved IP address so that it does not change.)*
- **Port** - *(not specified as the default port is being used)*
- **Resource** - rest/v1/status
- **Parameters** - *(none specified, the units of return values will be the current TouchCut units)*
- **Headers** - X-TouchCut-API-Key: 13F513776F764DA5B5F043C93F6678381343E346C4 *(Authentication is being provided via an API Key)*

After selecting the SEND button the TouchCut REST server has return a success response of "200 OK" with the content containing the JSON status information structure.

If the request returns a request error, check the error description. If the connection fails due to the application encountering a Self Signed Certificate then you will get the error below:



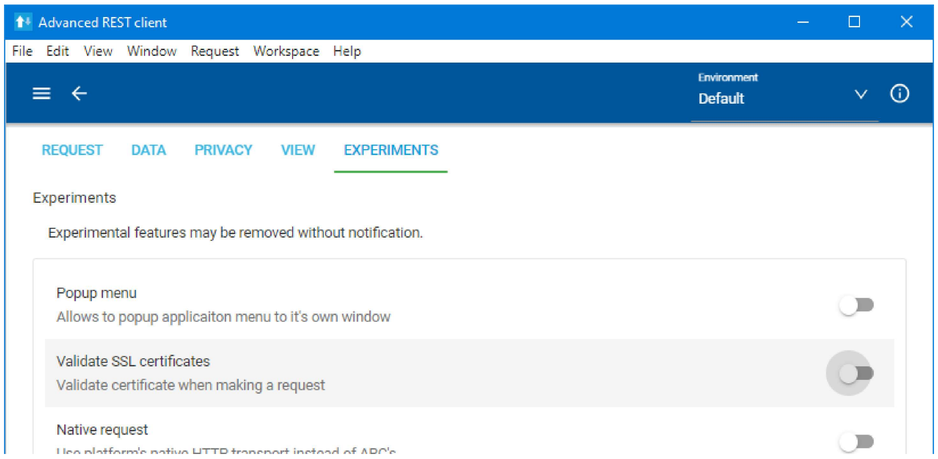
The requested URL can't be reached

The service might be temporarily down or it may have moved permanently to a new web address.

self signed certificate

Error indicating failed connection when application encountered a Self Signed Certificate

To avoid validating self signed certificates you will need to turn off the Validate SSL certificates option (from the File, Settings menu, Experiments tab).



Experiments

Experimental features may be removed without notification.

Popup menu
Allows to popup applicaiton menu to it's own window

Validate SSL certificates
Validate certificate when making a request

Native request

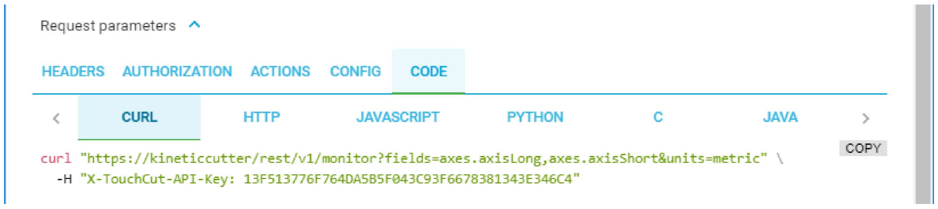
Figure 203: Turn off the Validate SSL certificates option

The following example shows a request to the monitor resource. The example adds a couple of query parameters to the URL. You can enter the parameters directly in the Request URL entry box, or expand the item to enter them as separate options (as below).

The ARC application also provides a CODE option under the Request parameters section to provide code templates for various systems for the entered REST request.

The CURL tab shows a command that can be used on the commandline using the cUrl utility. Note: the ARC application is providing an example for a Linux / Unix command line. If you are pasting it into a Windows command line you will need to remove the \ character. Also, if TouchCut is using a Self Signed Certificate then you will need to also add the -k or --insecure option to allow insecure server connections when using SSL.

See the cUrl Utility Example 441 for more information.

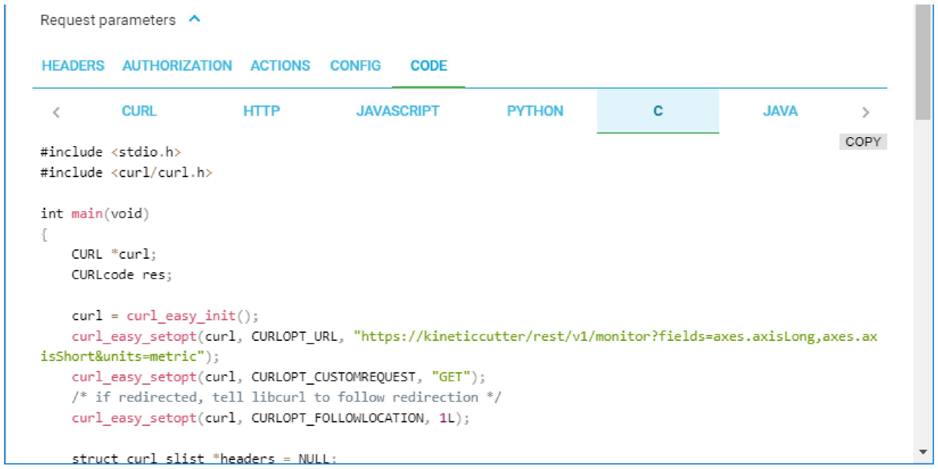


```
curl "https://kineticcutter/rest/v1/monitor?fields=axes.axisLong,axes.axisShort&units=metric" \
  -H "X-TouchCut-API-Key: 13F513776F764DA5B5F043C93F6678381343E346C4"
```

Figure 204: CURL tab

The HTTP tab shows a raw HTTP request. This is the raw text that is sent to the REST web server via the REST protocol (which is based on the HTTP protocol). You can directly connect to the REST web server via a network socket and send this information to make the request. However you will also need to full process the response yourself, parsing the response headers and extracting the response body. You will also need to take care of any SSL requirements (for HTTPS connections) yourself. Third party libraries, such as the curl library, can also automatically deal with redirects and proxies automatically if required.

The C code tab shows an example C program using the curl library. This is a bare bones request example that does not deal with obtaining any response body. It also does not set the option to turn SSL verification off. Refer to the c using the curl library 443 example for a more fully fledged c example program.



```
#include <stdio.h>
#include <curl/curl.h>

int main(void)
{
    CURL *curl;
    CURLcode res;

    curl = curl_easy_init();
    curl_easy_setopt(curl, CURLOPT_URL, "https://kineticcutter/rest/v1/monitor?fields=axes.axisLong,axes.ax
isShort&units=metric");
    curl_easy_setopt(curl, CURLOPT_CUSTOMREQUEST, "GET");
    /* if redirected, tell libcurl to follow redirection */
    curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);

    struct curl_slist *headers = NULL;
```

Figure 205: Example C program using the curl library

Lastly, if you select the POST method the ARC application will show a BODY tab under the Request parameters section. If the REST request requires any request body data it can be entered here.

cUrl Utility Example

For simple requests you can use the cUrl utility which is included in Windows 10 build 17063 onwards by default. If you are making an HTTPS request use the -k or --insecure option to allow insecure server connections when using SSL. This will avoid getting the following error due to TouchCut's self signed certificate:

```
curl: (77) schannel: next InitializeSecurityContext failed: SEC_E_UNTRUSTED_ROOT
(0x80090325) - The certificate chain was issued by an authority that is not
trusted.
```

This example requests the status information from the machine via the curl utility. If the request is made on the local controller you can use the HTTP protocol as the hostname will resolve to localhost. However, if the request is not on the local controller then HTTPS will be required and the -k option will turn off certificate verification to avoid a self signed certificate error.

```
curl  "https://kineticcutter/rest/v1/status?units=metric"  -k  -H  "X-TouchCut-API-
Key: 13F513776F764DA5B5F043C93F6678381343E346C4"
```

The command is made up of the following components:

- **Method** - GET *(Note: to do a POST specify the -d, --data <data> option)*
- **Protocol** - https
- **Hostname** - kineticcutter *(this can also be an IP address, however if the controller connects to the network via DHCP care should be taken to ensure it is configured to use a reserved IP address so that it does not change.)*
- **Port** - *(not specified as the default port is being used)*
- **Resource** - rest/v1/status
- **Parameters** - units=metric *(request to return values in metric)*
- **Headers** - X-TouchCut-API-Key: 13F513776F764DA5B5F043C93F6678381343E346C4 *(Authentication is being provided via an API Key)*

If successful the command will return with a JSON response similar to this:

```
{
  "feedOverride":100,
  "materialThickness":10,
  "toolIndex":0,
  "plateOrigin":{
    "vec":[0,0,0],
    "rot":[0,0,0]
  },
  "isHomed":false,
  "plateRotation":0,
  "toolOn":false,
  "machineState":"Idle",
  "stateName":"Idle",
  "programState":"Idle",
  "toolName":"Laser Pointer",
  "commandedPos":{
    "vec":[0,0,0],
    "rot":[0,0,0]
  },
  "feedrate":0,
  "materialName":"MS",
  "heightControlAuto":true,
  "units":"metric",
  "activeCarriages":[1],
  "drossBinMode":"Disabled",
  "spacedCarriages":[1],
  "programOverride":"Idle",
  "msgStatus":[],
  "currentUser":"Operator",
  "programProgress":0,
  "programName":"No Active Program",
  "material":"MS-10.00 mm (0.394\")",
  "actualPos":{
    "vec":[0,0,0],
    "rot":[0,0,0]
  }
}
```

c using the curl Library

If you want to use a programming language direct it is easier to use a library such as curl. You can find further information on how to use the curl library here: https://curl.haxx.se/

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <curl/curl.h>

struct MemoryStruct {
  char *memory;
  size_t size;
};

static size_t WriteMemoryCallback(void *contents, size_t size,
  size_t nmemb, void *userp)
{
  size_t realsize = size * nmemb;
  struct MemoryStruct *mem = (struct MemoryStruct *)userp;

  char *ptr = realloc(mem->memory, mem->size + realsize + 1);
  if(ptr == NULL) {
    /* out of memory! */
    printf("not enough memory (realloc returned NULL)\n");
    return 0;
  }

  mem->memory = ptr;
  memcpy(&(mem->memory[mem->size]), contents, realsize);
  mem->size += realsize;
  mem->memory[mem->size] = 0;

  return realsize;
}

int main(void)
{
  CURL *curl;
  CURLcode res;

  struct MemoryStruct chunk;

  curl_global_init(CURL_GLOBAL_DEFAULT);

  curl = curl_easy_init();
  if(curl) {
    chunk.memory = malloc(1);  /* will be grown as needed by the realloc above */
    chunk.size = 0;            /* no data at this point */

    /* set the request url */
    curl_easy_setopt(curl, CURLOPT_URL,
        "https://kineticcutter/rest/v1/monitor"
        "?fields=axes.axisLong,axes.axisShort&units=metric");
    /* specify the GET method */
    curl_easy_setopt(curl, CURLOPT_CUSTOMREQUEST, "GET");
    /* if redirected, tell libcurl to follow redirection */
    curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);

    /* add the API Key to the header list */
    struct curl_slist *headers = NULL;
    headers = curl_slist_append(headers,
        "X-TouchCut-API-Key: 13F513776F764DA5B5F043C93F6678381343E346C4");
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);

    /*
```

```
     * If you want to connect to a site who isn't using a certificate that is
     * signed by one of the certs in the CA bundle you have, you can skip the
     * verification of the server's certificate. This makes the connection
     * A LOT LESS SECURE.
     *
     * If you have a CA cert for the server stored someplace else than in the
     * default bundle, then the CURLOPT_CAPATH option might come handy for
     * you.
     */
    curl_easy_setopt(curl, CURLOPT_SSL_VERIFYPEER, 0L);

    /* send all data to this function  */
    curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION, WriteMemoryCallback);
    /* we pass our 'chunk' struct to the callback function */
    curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, (void *)&chunk);


    /* Perform the request, res will get the return code */
    res = curl_easy_perform(curl);
    if (res != CURLE_OK) {
      fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));
    }
    else {
      /*
      * Now, our chunk.memory points to a memory block that is chunk.size
      * bytes big and contains the remote file.
      *
      * Do something nice with it!
      */
      printf("%lu bytes retrieved:\n", (unsigned long)chunk.size);
      printf("%s\n", chunk.memory);
    }

    free(chunk.memory);

    /* Cleanup curl stuff */
    curl_easy_cleanup(curl);
  }

  /* we're done with libcurl, so clean it up */
  curl_global_cleanup();

  return 0;
}
```

The application calls a command made up of the following components:

- **Method** - GET
- **Protocol** - https
- **Hostname** - kineticcutter *(this can also be an IP address, however if the controller connects to the network via DHCP care should be taken to ensure it is configured to use a reserved IP address so that it does not change.)*
- **Port** - *(not specified as the default port is being used)*
- **Resource** - rest/v1/monitor
- **Parameters** - fields=axes.axisLong,axes.axisShort *(request information for axisLong and axisShort only)*
- **Parameters** - units=metric *(request to return values in metric)*
- **Headers** - X-TouchCut-API-Key: 13F513776F764DA5B5F043C93F6678381343E346C4 *(Authentication is being provided via an API Key)*

If successful the application will return with a JSON response similar to this:

```
{
  "io":[],
  "units":"metric",
  "axes":[
    {
      "errorCode":0,
      "canMoveRev":true,
      "canMoveFwd":true,
      "errorLimit":1,
      "isPosErrorInputOn":false,
      "isEnabled":true,
      "torque":0,
      "speedUnitType":"utFeed",
      "commandedPos":0,
      "isMoving":false,
      "posError":0,
      "index":1,
      "isFwdSoftLimOn":false,
      "unitType":"utLength",
      "isHomeInputOn":false,
      "torqueLimit":150,
      "io":[],
      "address":"MC, axisLong",
      "isRevLimInputOn":false,
      "scale":1,
      "isRevSoftLimOn":false,
      "name":"axisLong",
      "isFwdLimInputOn":false,
      "actualVel":0,
      "actualPos":0
    },
    {
      "errorCode":0,
      "canMoveRev":true,
      "canMoveFwd":true,
      "errorLimit":1,
      "isPosErrorInputOn":false,
      "isEnabled":true,
      "torque":0,
      "speedUnitType":"utFeed",
      "commandedPos":0,
      "isMoving":false,
      "posError":0,
      "index":10,
      "isFwdSoftLimOn":false,
      "unitType":"utLength",
      "isHomeInputOn":false,
      "torqueLimit":180,
      "io":[],
      "address":"MC, axisShort",
      "isRevLimInputOn":false,
      "scale":1,
      "isRevSoftLimOn":false,
      "name":"axisShort",
      "isFwdLimInputOn":false,
      "actualVel":0,
      "actualPos":0
    }
  ]
}
```

REST API v1 Reference

The following resources are available from the version 1 REST API.

rest/v1/auth

The auth resource allows you to attempt a login. If successful a login will return a JSON Web Token (JWT) that can then be passed to subsequent rest calls via the "Authorization: Bearer " header key.

HTTPS required

Request:

HTTP Method:
POST
Headers:
None
Parameters:
username
password

Success Response:

HTTP/1.0 Status 200
Headers:
Content-Type: application/json; charset=utf-8
Content-Length: <length of json content>
Content:
```
{ "success":"true", "token":"<JWT>" }
```

Failure Responses:

HTTP/1.0 Status 400 (Bad Request)
Headers:
Content-Type: application/json; charset=utf-8
Content-Length: <length of json response>
Content (missing username):
```
{ "code":"1003", "msg":"Required parameter missing: username" }
```
Content (missing password):
```
{ "code":"1003", "msg":"Required parameter missing: password" }
```
Content (application still initialising):
```
{ "code":"100", "msg":"Application not ready" }
```
Content (username not found):
```
{ "code":"101", "msg":"Invalid Username" }
```
Content (password not correct:
```
{ "code":"102", "msg":"Invalid Password" }
```

HTTP/1.0 Status 401 (Unauthorized, user does not have web access privileges)

HTTP/1.0 Status 405 (Method not allowed, sub method below auth/ is invalid)

HTTP/1.0 Status 500 (Internal Error)
Headers:
Content-Type: application/json; charset=utf-8
Content-Length: <length of json response>
Content (error creating JSON Web Token):
```
{ "code":"2000", "msg":"Error creating JWT" }
```

rest/v1/auth/change-password

The auth/change-password resource allows you to attempt a password change for the JWT related user. An active JSON Web Token (JWT) is required to be passed to the "Authorization: Bearer " header key. If the password is successfully changed a new JWT is returned.

HTTPS required

Request:

HTTP Method:
POST
Headers:
Authorization: Bearer <JWT>
Parameters:
old-password
new-password
confirm-password

Success Response:

HTTP/1.0 Status 200
Headers:
Content-Type: application/json; charset=utf-8
Content-Length: <length of json content>
Content:
```
{ "success":"true", "token":"<JWT>" }
```

Failure Responses:

HTTP/1.0 Status 400 (Bad Request)
Headers:
Content-Type: application/json; charset=utf-8
Content-Length: <length of json response>
Content (missing old password):
```
{ "code":"1003", "msg":"Required parameter missing: old-password" }
```
Content (missing new password):
```
{ "code":"1003", "msg":"Required parameter missing: new-password" }
```
Content (missing confirm password):
```
{ "code":"1003", "msg":"Required parameter missing: confirm-password"}
```
Content (confirm password does not match new password):
```
{ "code":"1002", "msg":" Invalid parameter: confirm-password" }
```
Content (application still initialising):
```
{ "code":"100", "msg":"Application not ready" }
```
Content (username from JWT not found):
```
{ "code":"101", "msg":"Invalid Username" }
```
Content (password not correct):
```
{ "code":"102", "msg":"Invalid Password" }
```

HTTP/1.0 Status 401 (Unauthorized, inactive / invalid JWT)

HTTP/1.0 Status 401 (Unauthorized, user does not have change password privileges)

HTTP/1.0 Status 403 (Forbidden, JWT user does not have the required privileges)

HTTP/1.0 Status 405 (Method not allowed, sub method below auth/change-password/ is invalid)

HTTP/1.0 Status 500 (Internal Error)
Headers:
Content-Type: application/json; charset=utf-8
Content-Length: <length of json response>
Content (error creating replacement JSON Web Token):

```
{ "code":"2000", "msg":"Error creating JWT" }
```

rest/v1/machine-info

General machine information. Does not require an active login.

HTTPS required for remote connection
HTTP/HTTPS for localhost connection

Request:

HTTP Method:
GET
Headers:
None
Parameters:
None

Success Response:

HTTP/1.0 Status 200
Headers:
Content-Type: application/json; charset=utf-8
Content-Length: <length of json content>
Content:

```
{
  "programName":"<string>",
  "exePath":"<string>",
  "exeName":"<string>",
  "exeVersion":"<string>",
  "licensedTo":"<string>",
  "primeCutInfo":"<string>",
  "registrationInfo":"<string>",
  "contactInfo":{
    "companyName":"<string>",
    "postalAddr":"<string>",
    "streetAddr":"<string>",
    "countryName":"<string>",
    "phone":"<string>",
    "fax":"<string>",
    "email":"<string>",
    "webSite":"<string>"
  },
  "tools":[
    {
      "toolType":"<string>",
      "toolName":"<string>",
      "position":<integer>
    }
  ]
}
```

Failure Responses:

HTTP/1.0 Status 405 (Method not allowed, sub method below machine-info/ is invalid)

rest/v1/status

General machine status information. An active JSON Web Token (JWT) is required to be passed to the "Authorization: Bearer " header key, or a valid API Key is required to be passed to the "X-TouchCut-API-Key: " header key.

HTTPS required for remote connection
HTTP/HTTPS for localhost connection

Request:

HTTP Method:

GET

Headers:

Authorization: Bearer <JWT>

X-TouchCut-API-Key: <API Key>

Parameters:

units	one of: 'default', 'metric', 'imperial' *(optional)*

Success Response:

HTTP/1.0 Status 200

Headers:

Content-Type: application/json; charset=utf-8

Content-Length: <length of json content>

Content:

```
{
  "currentUser":"<string>",
  "isHomed":<boolean>,
  "machineState":"<string>",
  "programState":"<string>",
  "programOverride":"<string>",
  "stateName":"<string>",
  "toolIndex":<integer>,
  "toolName":"<string>",
  "toolOn":<boolean>,
  "spacedCarriages":[<integer>],
  "activeCarriages":[<integer>],
  "heightControlAuto":<boolean>,
  "drossBinMode":"<string>",
  "programName":"<string>",
  "programProgress":<double>,
  "plateOrigin":{
    "vec":[<double>,<double>,<double>],
    "rot":[<double>,<double>,<double>]
  },
  "plateRotation":<double>,
  "material":"<string>",
  "materialName":"<string>",
  "materialThickness":<double>,
  "commandedPos":{
    "vec":[<double>,<double>,<double>],
    "rot":[<double>,<double>,<double>]
  },
  "actualPos":{
    "vec":[<double>,<double>,<double>],
    "rot":[<double>,<double>,<double>]
  },
  "feedrate":<double>,
  "feedOverride":<double>,
  "msgStatus":[
    {
      "key":"<string>",
      "msg":"<string>"
    }
  ],
  "units":"<string>"
}
```

Failure Responses:

HTTP/1.0 Status 401 (Unauthorized, inactive / invalid JWT)

HTTP/1.0 Status 405 (Method not allowed, sub method below status/ is invalid)

rest/v1/stats

Machine stats information. An active JSON Web Token (JWT) is required to be passed to the "Authorization: Bearer " header key, or a valid API Key is required to be passed to the "X-TouchCut-API-Key: " header key.

This endpoint will retrieve the Machine Totals and Resettable Machine Totals information from stats.ini. It will also retrieve session and consumable information for a selected date range from the daily stats %d.ini files. The date range is automatically clipped to the available stats logging range (i.e. weekly, monthly, yearly).

If condensed is specified the output will not include records that have zero values.

HTTPS required for remote connection
HTTP/HTTPS for localhost connection

Request:

HTTP Method:
GET
Headers:
Authorization: Bearer <JWT>
X-TouchCut-API-Key: <API Key>
Parameters:

start-date	the first day to retrieve information for (in ISO8601 format, i.e. yyyy-mm-dd) *(optional, default: tomorrow minus logging range)*
end-before-date	retrieve data up till this date (in ISO8601 format, i.e. yyyy-mm-dd) *(optional, default: tomorrow)*
condensed	boolean value (e.g.: true, false, 1, 0) *(options, default: false)*
units	one of: 'default', 'metric', 'imperial' *(optional)*

Success Response:

HTTP/1.0 Status 200
Headers:
Content-Type: application/json; charset=utf-8
Content-Length: <length of json content>
Content:
```
{ "machine": [
    { "tool": <integer, toolNumber>,
      "name": "<string, toolName>",
      "time": <double, seconds>,
      "distance": <double, utLength>,
      "starts": <integer, count>,
      "errors": <integer, count>,
      "carriages": [
        { "carriage": <integer, boxNumber>,
          "time": <double, seconds>,
          "distance": <double, utLength>,
          "starts": <integer, count>,
          "errors": <integer, count>
        }
      ]
    }
  ],
  "machineResettable": [
    { "tool": <integer, toolNumber>,
      "name": "<string, toolName>",
      "time": <double, seconds>,
      "distance": <double, utLength>,
      "starts": <integer, count>,
      "errors": <integer, count>,
      "lastReset": "<string, ISO8601ToDateTime>",
      "carriages": [
        { "carriage": <integer, boxNumber>,
          "time": <double, seconds>,
          "distance": <double, utLength>,
          "starts": <integer, count>,
          "errors": <integer, count>,
          "lastReset": "<string, ISO8601ToDateTime>"
        }
      ]
    }
  ],
  "sessions": [
    { "user": "<string, session user name>",
      "startTime": "<string, ISO8601ToDateTime>",
      "finishTime": "<string, ISO8601ToDateTime>",
      "activeTime": <double, seconds>,
      "errorTime": <double, seconds>,
      "idleTime": <double, seconds>,
      "xyMovingTime": <double, seconds>,
      "productiveTime": <double, seconds>,
      "unproductiveTime": <double, seconds>,
      "toolOnTime": <double, seconds>,
      "toolTotals": [
        { "tool": <integer, toolNumber>,
          "name": "<string, toolName>",
          "time": <double, seconds>,
          "distance": <double, utLength>,
          "starts": <integer, count>,
          "errors": <integer, count>,
          "carriages": [
            { "carriage": <integer, boxNumber>,
              "time": <double, seconds>,
              "distance": <double, utLength>,
              "starts": <integer, count>,
              "errors": <integer, count>
            }
          ]
        }
      ],
      "programs": [
        { "startTime": "<string, ISO8601ToDateTime>",
          "name": "<string, programName>"
        }
      ],
      "errors": [
        { "errorTime": "<string, ISO8601ToDateTime>",
          "error": "<string, message>"
        }
      ]
    }
  ],
  "consumables": [
    { "date": "<string, ISO8601ToDateTime>",
      "toolName": "<string, toolName>",
      "material": "<string, materialName>",
      "thickness": <double, utThickness>,
      "process": "<string, processCompareName>",
      "consType": "<string, consumableType>",
      "consID": "<string, consumableID>",
      "pierces": <integer, count>,
      "distance": <double, utLength>,
      "time": <double, seconds>
    }
  ],
  "units":"<string>"
}
```

Failure Responses:

HTTP/1.0 Status 401 (Unauthorized, inactive / invalid JWT)

HTTP/1.0 Status 405 (Method not allowed, sub method below status/ is invalid)

rest/v1/monitor

Information about the machines IO and Axes. An active JSON Web Token (JWT) is required to be passed to the "Authorization: Bearer " header key, or a valid API Key is required to be passed to the "X-TouchCut-API-Key: " header key.

HTTPS required for remote connection
HTTP/HTTPS for localhost connection

Request:

HTTP Method:
  GET
Headers:
  Authorization: Bearer <JWT>
  X-TouchCut-API-Key: <API Key>
Parameters:

fields	a comma separated list of fields to return, if blank returns all *(optional)* fields can use * as a wildcard match to return all items under a group
units	one of: 'default', 'metric', 'imperial' *(optional)*

Success Response:

HTTP/1.0 Status 200
Headers:
  Content-Type: application/json; charset=utf-8
  Content-Length: <length of json content>
Content:

```
{
  "io":[
    "din":[
      {
        "name":"<string>",
        "address":"<string>",
        "value":<boolean>
      }
    ],
    "dout":[
      {
        "name":"<string>",
        "address":"<string>",
        "value":<boolean>
      }
    ],
    "ain":[
      {
        "name":"<string>",
        "address":"<string>",
        "scale":<double>,
        "value":<double>,
        "unitType":"<string>"
      }
    ],
    "aout":[
      {
        "name":"<string>",
        "address":"<string>",
        "scale":<double>,
        "value":<double>,
        "unitType":"<string>"
      }
    ]
  ],
  "axes":[
    {
      "name":"<string>",
      "address":"<string>",
      "index":<TAxis>,
      "unitType":"<string>",
      "speedUnitType":"<string>",
      "scale":<double>,
      "errorCode":<uint64>,
      "commandedPos":<double>,
      "actualPos":<double>,
      "posError":<double>,
      "errorLimit":<double>,
      "actualVel":<double>,
      "torque":<double>,
      "torqueLimit":<double>,
      "isEnabled":<boolean>,
      "isMoving":<boolean>,
      "isPosErrorInputOn":<boolean>,
      "isHomeInputOn":<boolean>,
      "isRevLimInputOn":<boolean>,
      "isFwdLimInputOn":<boolean>,
      "isRevSoftLimOn":<boolean>,
      "isFwdSoftLimOn":<boolean>,
      "canMoveFwd":<boolean>,
      "canMoveRev":<boolean>,
      "io":[
        "din":[
          {
            "name":"<string>",
            "address":"<string>",
            "value":<boolean>
          }
        ],
        "dout":[
          {
            "name":"<string>",
            "address":"<string>",
            "value":<boolean>
          }
        ],
        "ain":[
          {
            "name":"<string>",
            "address":"<string>",
            "scale":<double>,
            "value":<double>,
            "unitType":"<string>"
          }
        ],
        "aout":[
          {
            "name":"<string>",
            "address":"<string>",
            "scale":<double>,
            "value":<double>,
            "unitType":"<string>"
          }
        ]
      ]
    }
  ],
  "units":"<string>"
}
```

Failure Responses:

HTTP/1.0 Status 401 (Unauthorized, inactive / invalid JWT)

HTTP/1.0 Status 405 (Method not allowed, sub method below monitor/ is invalid)

rest/v1/programs

Information about the programs loaded within TouchCut. An active JSON Web Token (JWT) is required to be passed to the "Authorization: Bearer " header key, or a valid API Key is required to be passed to the "X-TouchCut-API-Key: " header key.

Note: interface subject to change.

HTTPS required for remote connection
HTTP/HTTPS for localhost connection

Request:

HTTP Method:
GET
Headers:
Authorization: Bearer <JWT>
X-TouchCut-API-Key: <API Key>
Parameters:

onlyChangedData	*(optional)*
nestmapPercentGenerated	*(optional)*
displayingProgramFileId	*(optional)*
displayingProgramModified	*(optional)*
offset	pagination offset *(optional)*
limit	pagination limit *(optional)*
fields	a comma separated list of fields to return, if blank returns all *(optional)*
units	one of: 'default', 'metric', 'imperial' *(optional)*

Success Response:

HTTP/1.0 Status 200
Headers:
Content-Type: application/json; charset=utf-8
Content-Length: <length of json content>
Content:

```
{
  Interface subject to change,
  "units":"<string>"
}
```

Failure Responses:

HTTP/1.0 Status 401 (Unauthorized, inactive / invalid JWT)

HTTP/1.0 Status 405 (Method not allowed, sub method below programs/ is invalid)

rest/v1/unloader

Information about the part unloader, the pallet stations and the unloaded parts. An active JSON Web Token (JWT) is required to be passed to the "Authorization: Bearer " header key, or a valid API Key is required to be passed to the "X-TouchCut-API-Key: " header key.

Note: this request is only available in TouchCut 7.

Note: interface subject to change.

HTTPS required for remote connection
HTTP/HTTPS for localhost connection

Request:

HTTP Method:
  GET
Headers:
  Authorization: Bearer <JWT>
  X-TouchCut-API-Key: <API Key>
Parameters:

history	*(optional)*
start-date	*(optional)*
end-before-date	*(optional)*
units	one of: 'default', 'metric', 'imperial' *(optional)*

Success Response:

HTTP/1.0 Status 200
Headers:
  Content-Type: application/json; charset=utf-8
  Content-Length: <length of json content>
Content:

```
{
  Interface subject to change,
  "units":"<string>"
}
```

Failure Responses:

HTTP/1.0 Status 401 (Unauthorized, inactive / invalid JWT)

HTTP/1.0 Status 405 (Method not allowed, sub method below programs/ is invalid)

rest/v1/screenshot

Returns a .png screen capture of the main TouchCut window. An active JSON Web Token (JWT) is required to be passed to the "Authorization: Bearer " header key, or a valid API Key is required to be passed to the "X-TouchCut-API-Key: " header key.

Note: the screenshot will be updated no faster than once per second. If a second request is made within a second of the previous request, the previous screenshot will be returned.

HTTPS required for remote connection
HTTP/HTTPS for localhost connection

Request:

HTTP Method:
  GET
Headers:
  Authorization: Bearer <JWT>
  X-TouchCut-API-Key: <API Key>
Parameters:
  None

Success Response:

HTTP/1.0 Status 200
Headers:
  Content-Type: image/png

Content-Length: <length of png content>

Content:

Screenshot png picture

Failure Responses:

HTTP/1.0 Status 401 (Unauthorized, inactive / invalid JWT)

HTTP/1.0 Status 405 (Method not allowed, sub method below screenshot/ is invalid).

Terminology

This section describes various terminology used in the TouchCut REST API documentation.

What is a REST API

API is the acronym for **A**pplication **P**rogramming **I**nterface, which is a software intermediary that allows two applications to talk to each other in a defined fashion.

REST is the acronym for **RE**presentational **S**tate **T**ransfer. It is an architectural style for distributed hypermedia systems and uses communication protocols similar to the web http/https protocols.

Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API to communicate with a server on the internet. Today those API's are most likely to be based on the REST architectural style.

What is a REST Architectural Style

Like any other architectural style, REST also has it's own 6 guiding constraints which must be satisfied if an interface needs to be referred as RESTful. These principles are listed below.

Guiding Principles of REST

- **Client–server** – By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.
- **Stateless** – Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client.
- **Cacheable** – Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.
- **Uniform interface** – By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behavior of components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.
- **Layered system** – The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot "see" beyond the immediate layer with which they are interacting.
- **Code on demand (optional)** – REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented.

(Reference: https://restfulapi.net/)

What is a JSON Web Token (JWT)

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed.

The TouchCut Server uses the JWT for user authorization. Once the user is logged in TouchCut returns a JWT representing the session. Each subsequent request then includes the JWT, allowing the user to access resources that are permitted with that token. The JWT can be included by either the "Authorization: Bearer " header key or the "__Secure-jwt" cookie.

Call rest/v1/auth 446 to request a user login and return a JWT. The rest/v1/auth 446 request requires a secure HTTPS connection to ensure the user credentials are protected during transmission.

Refer to https://jwt.io/introduction/ for more information.

What is an API Key

An API key or Application Programming Interface Key is a code that gets passed in by a client application to allow the TouchCut Server to authorize the client without requiring a login. The API Key is provided by TouchCut and provides access privileges to the TouchCut Server. The client should keep their copy of the API Key secret and secure to avoid unauthorized access being gained. The API Key can be revoked by TouchCut if access is no longer desired by the key.

TouchCut REST API requests using the API Key can can include the key using the "X-TouchCut-API-Key: " header key. All API request are required to either use a secure HTTPS connection, or a non-secure HTTP connection if called on the localhost, to ensure the API Key is protected during transmission.