

Introduction to the MCAM Web API

Date: August 30, 2022

Updated: October 6, 2022

By: Ramona Optics Inc.

The MCAM provides a web API for seamless integration into automated workflows. At its core, the MCAM web API tries to approach a REST API while enabling control over the MCAM's mechanical capabilities.

System Architecture

The MCAM server computer, that runs the web service for the MCAM acquisition control and image analysis, listens on port 8800 for incoming requests. The system is assumed to be connected to the local network via ethernet. The MCAM and the MCAM server are assumed to always be connected to one other. The controlling scheduling system (e.g. Green Button Go) is assumed to be installed on a different computer, and able to communicate with the MCAM server on the same secure network.

Network Connection

The MCAM is assumed to be connected to the local network via ethernet. At minimum, a 1 Gbps network connection is required, and a 2.5 Gbps connection is recommended. Depending on the demands of the application, a 10 Gbps can be accommodated on the MCAM server.

Security Model

The MCAM assumes that it is placed in a secured facility. The server is configured to listen to any requests that arrive at its IP address. The server can be configured to require certain credentials described below, but these are only advised to prevent accidental requests, and not malicious requests to the MCAM server.

HTTP Basic Authentication

To enable HTTP Basic Authentication, the server must be started with the environment variable `RAMONAOPTICS_WEBMCAM_AUTHENTICATION` set to "httpbasic". The username and password can be set from the `RAMONAOPTICS_WEBMCAM_USERNAME` and `RAMONAOPTICS_WEBMCAM_PASSWORD` environment variables.

Shared File Server Installation

Upon installation, the Ramona Optics will work with the client to mount the shared file server so that it appears as a local folder on the MCAM server. We briefly describe a few known share file configurations that are compatible with Ramona Optics' MCAM server.

API Function Calls

There are 8 basic API endpoints:

1. GET /v1/status
 - a. The response contains an entry with information pertaining to the MCAM server application. The main information returned in this endpoint is the version of the server application described in `owl_version`
 - i. `owl_version`: The version of the MCAM server application.
2. GET /v1/mcam/search
 - a. This call is generally issued once upon instrument startup.
 - b. It returns a list of available serial numbers that may be available to clients to open.
3. GET /v1/mcam/{serial_number}
 - a. Return status information for the MCAM serial number specified in the request
4. POST /v1/mcam/{serial_number}
 - a. The MCAM server application claims control over the MCAM instrument.
 - b. The MCAM will automatically enter a retracted state when it is open.
 - c. This endpoint can take 10-30 seconds to startup and ensures that the motion subsystem of the MCAM is ready to take commands.
 - d. This command may fail for the following reasons:
 - i. The MCAM is powered off while the MCAM server is powered on
 - ii. Another application is connected to the MCAM (e.g. the native application).
5. DELETE /v1/mcam/{serial_number}
 - a. The MCAM may enter a retracted state when the communication connection is closed.
 - b. This call is generally called if the user wishes to use the MCAM instrument with the native graphical application for troubleshooting.
6. GET /v1/mcam/{serial_number}/state/search
 - a. Provides the list of available states for a particular MCAM.
7. GET /v1/mcam/{serial_number}/state
8. POST /v1/mcam/{serial_number}/state
 - a. Requests that the MCAM enters a particular state.
 - b. "acquisition" state:
 - i. In this state the sample holder (if present) is retracted and the MCAM is ready to acquire images.
 - ii. The MCAM will automatically enter this state when the `/run_assay` endpoint is called.
 - c. "loading" state:
 - i. The sample holder is extended into the defined loading position.
 - d. "unloading" state:
 - i. The sample holder is extended into the defined unloading position.
9. GET /v1/mcam/{serial_number}/assay/search

- a. Returns a list of strings, each an identifier of the different types of the available assays (or experiments) that can be run on the MCAM.
- 10. GET /v1/mcam/{serial_number}/assay/{assay_name}/configuration
 - a. Return a list of strings, each containing a valid configuration of the different available assay configurations.
- 11. GET /v1/mcam/{serial_number}/assay/{assay_name}
 - a. Returns the assay schema.
- 12. POST /v1/mcam/{serial_number}/assay/{assay_name}
 - a. This is the main function to call to acquire and process data. It is described in detail in subsequent sections.

API Function Workflow

The API is documented in detail in the staging website. The openapi.json can be downloaded from the staging website. Most requests are HTTP Post requests and take in their parameters through a json request body.

The general workflow can be summarized by:

1. Only a single client application is assumed to be communicating with the MCAM at once.
 - a. Initially, no locking mechanism exists to avoid conflicts between multiple clients.
2. The client application (installed a different computer) checks the GET /v1/mcam/search endpoint.
 - a. If the returned value of serial_numbers is an empty list, then the webserver is not connected to the MCAM instrument (this can occur if a user was trying to access the MCAM with the native application).
 - b. If the returned value of the serial_numbers contains one or more strings, it indicates the serial number of the MCAM connected by the server.
3. The client application initiates a connection with the MCAM of its choice through the POST /v1/mcam/{serial_number} endpoint
4. The client application then probes:
 - a. GET /v1/mcam/{serial_number}/state/search
 - i. To get a list of the available mechanical states of the particular MCAM.
 - b. GET /v1/mcam/{serial_number}/assay/search
 - i. To get a list of the available assays for the MCAM
5. The client application sets the state of the MCAM to load the sample by issuing a post request to POST /v1/mcam/{serial_number}/state with the desired state.
6. The client application issues a call to the POST /v1/mcam/{serial_number}/assay/{assay_name} endpoint as described below

- a. After the call to `run_assay`, the MCAM may be in an arbitrary state. Typically, it will be left in the “acquisition” state, though this can change between assays and the particular assay configuration.
7. The client application sets the state of the MCAM to unload the sample by issuing a call to the `POST /v1/mcam/{serial_number}/state` with the particular state of interest.
8. Repeat steps 5-7 as desired.
9. Once the connection is no longer needed, one can call the endpoint `DELETE /v1/mcam/{serial_number}` to release the resources associated with the MCAM from the server.

The assay endpoint

We focus our attention on the `POST /v1/mcam/{serial_number}/assay/{assay_name}` endpoint.

The endpoint call specifies both the MCAM serial number that will complete the operation, and the assay name that will be executed. The body of the request contains 4 keys:

- `save_location`: a string containing the location of the output files.
 - Required
 - This path is relative to the mcam server itself
- `metadata`: a dictionary containing metadata to be added to the acquired dataset.
 - Optional
 - This dictionary can contain multiple key value pairs. All keys must be strings. The values may be strings, floating point numbers, or integers.
- `configuration`: A string containing preconfigured settings for the particular assay.
 - Optional
 - If not provided the default parameters for the given assay will be used.
 - The default parameters are almost surely not the ones you want to use.
- `parameters`: A dictionary containing key value pairs to overwrite in the assay parameters specified in the `assay_filename`.
 - The default key value pairs are those provided by the settings file defined in `assay_filename`.

For example, they may be specified as json application data. As a cURL request, these are written as

```
curl -X 'POST' \
'http://mcam_ip_address.local:8800/v1/mcam/0x4EADBEEFCAFE1010BA5EDA11
/assay/acquire_and_save \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
```

```
-d '{
  "save_location": "/shared_folder/data/my_file_location",
  "metadata": {
    "barcode": "A329812",
    "operator_name": "John Doe"
  },
  "parameters": {}
}'
```

Environment Variables used by the MCAM Server

The MCAM server can be configured via the following environment variables to adjust the server settings:

Environment Variable	Usage	Allowed values	Example
RAMONAOPTICS_WEBMCAM_AUTHENTICATION	Specifying the authentication method.	httpbasic or empty (no authentication required)	httpbasic
RAMONAOPTICS_WEBMCAM_USERNAME	Username when httpbasic authentication is used.	Any string, spaces are discouraged.	ramona
RAMONAOPTICS_WEBMCAM_PASSWORD	Password when httpbasic authentication is used.	Any string, spaces are discouraged.	gigapixel
RAMONAOPTICS_WEBMCAM_EXTRA_AVAILABLE_SERIAL_NUMBERS	Specifying additional serial numbers that “appear available” when testing the MCAM.	A list of comma separated strings.	RAMONAOPTICS_WEBMCAM_EXTRA_AVAILABLE_SERIAL_NUMBERS=0x4EADBEEFCAFE1010BA5EDA11
RAMONAOPTICS_WEBMCAM_STARTUP_SERIAL_NUMBER	Serial number to which the WebMCAM Server connects to upon startup.	A single serial number (string)	RAMONAOPTICS_WEBMCAM_STARTUP_SERIAL_NUMBER=0x4EADBEEFCAFE1010BA5EDA11
RAMONAOPTICS_WEBMCAM_PORT	Network port on which the server lists.	A number specifying the port on which the server listens	RAMONAOPTICS_WEBMCAM_PORT=8800
RAMONAOPTICS_WEBMCAM_SSL_KEYFILE	SSL keyfile used for https communication.	A valid local path.	RAMONAOPTICS_WEBMCAM_SSL_KEYFILE=/home/ramona/cert/webcamdemo.ramonalabs.com+3-key.pem
RAMONAOPTICS_WEBMCAM_SSL_CERTFILE	SSL certfile used for https communication.	A valid local path.	RAMONAOPTICS_WEBMCAM_SSL_CERTFILE=/home/ramona/cert/webcamdemo.ramonalabs.com+3.pem

MCAM Server Setup

Remote Access Via SSH

Ensure that the MCAM and MCAM server are powered on, and connected to the network.

- The MCAM computer hosts an SSH server.
- If no SSH key is setup on the machine, a local login can be used to install the

```
cat id_rsa.pub >> ~/.ssh/authorized_keys
```

If working behind a router or firewall, you will have to set up the correct firewall rules and port forwarding rules to the connection through.

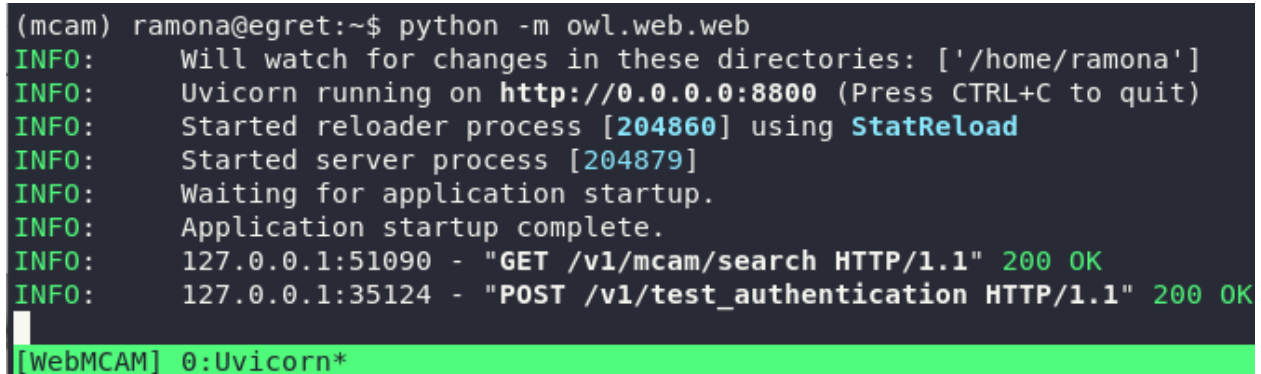
Troubleshooting Firewall Issues

The MCAM computer will attempt to launch the web server in a tmux session on bootup.

However, access from another computer may be blocked due to network security. To verify that the MCAM server is correctly started, use a terminal on the MCAM server to connect to the tmux session. You can connect to the tmux session through the command:

```
tmux a
```

and you will have access to the running web server.



```
(mcam) ramona@egret:~$ python -m owl.web.web
INFO: Will watch for changes in these directories: ['/home/ramona']
INFO: Uvicorn running on http://0.0.0.0:8800 (Press CTRL+C to quit)
INFO: Started reloader process [204860] using StatReload
INFO: Started server process [204879]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:51090 - "GET /v1/mcam/search HTTP/1.1" 200 OK
INFO: 127.0.0.1:35124 - "POST /v1/test_authentication HTTP/1.1" 200 OK
[WebMCAM] 0:Uvicorn*
```

From the MCAM computer, you can check that the server is running by connecting to the website through the installed web browser (Firefox or Google Chrome):

<https://localhost:8800/docs>

Verify that the screen looks similar to the above image of the correctly running web server.

WebMCAM API by Ramona Optics

0.18.156

OAS3

[/openapi.json](#)

The Ramona Optics WebAPI for MCAM control.

WebMCAM

GET `/v1/mcam/{serial_number}/state/search` List of available states for the specific MCAM.

GET `/v1/mcam/{serial_number}/state` Get the current state of the MCAM

POST `/v1/mcam/{serial_number}/state` Set the state of the MCAM

GET `/v1/mcam/{serial_number}/assay/search` List available assays for the particular MCAM.

GET `/v1/mcam/{serial_number}/assay/{assay_name}` MCAM Assay Information

POST `/v1/mcam/{serial_number}/assay/{assay_name}` MCAM Run Assay

Verify you can connect to the documentation page in a web browser by navigating to the ip address and the port shown in the tmux session.

Once you have verified that the server is correctly configured on the MCAM computer, you should disconnect from the tmux session:

Ctrl+b, then d

To list the IP addresses of the MCAM computer, you can use the command

```
ip a
```

The IP addresses will be listed next to network each interface. The IP address will often start with 10.X.X.X where each X represents a number between 0 and 255 (inclusively). Once the IP address of the MCAM server has been identified, you can attempt to connect to the MCAM server on your workstation through the address:

<https://10.X.X.X:8800/docs>

If you are unable to connect to the WebMCAM webpage, a temporary solution to get around a problematic firewall is to use a [WiFi hotspot](#) on the MCAM computer and connect to it directly. This should allow direct communication without any firewall interference. If you are resorting to this solution, you should contact your network administrator to ensure that a long term solution can be found to connect to the MCAM server through a wired ethernet connection.

Windows SAMBA Setup

If a Windows SAMBA is required Ramona Optics will install the required [cifs](#) utilities on the MCAM server. Users may work with the Ramona Optics engineering team to ensure that the shared drive configuration is correctly added so that the shared drives are permanently mounted.

If a Windows SAMBA is required Ramona Optics will install the required [cifs](#) utilities on the MCAM server. Users may work with the Ramona Optics engineering team to ensure that the shared drive configuration is correctly added so that the shared drives are permanently mounted. A sample configuration may resemble:

```
sudo apt install cifs-utils
# Create the credentials file
echo username=myusername > /home/ramona/.smbcredentials
echo password=mypassword >> /home/ramona/.smbcredentials
# echo domain=domain >> /home/ramona/.smbcredentials
# Add the line to /etc/fstab
//localhost/smbashare /shared cifs
credentials=/home/ramona/.smbcredentials,file_mode=0755,dir_mode=0755,user=ramona,group=ramona,nobootwait
0 0
sudo mount -a
# Ensure that the drive is visible
sudo reboot
# Ensure that the drive is still visible
```

Staging Website

<https://webmcamdemo.ramonalabs.com:8800/docs/>

Gaining access to the virtual MCAM shell

If you need to gain access to a virtual MCAM bash shell, please contact Ramona Optics by email and provide them your public key for SSH at help@ramonaoptics.com . We will provide you information about the login IP and the port.

Customization of Assays

The MCAM API allows for a large amount of customization based on the required experiments. All experimental parameters are exposed to the end user based on key-value pairs stored in a json dictionary. Many of the parameters are described in the endpoint `GET`

`/v1/mcam/{serial_number}/assay/{assay_name}` . Please contact Ramona Optics for more information about assay parameters.

Known Limitations

1. Long running assay may cause hangups with blocking http requests.
 - a. Currently assays are expected to run between 1-second and 60-seconds.
 - b. However, longer assays may run up to 300-600 seconds (5 minutes) which may cause TCP requests to drop.
2. Multiple client requests can conflict.
 - a. Currently there is no locking mechanism stopping multiple clients from issuing calls.
 - b. All API calls are handled sequentially meaning that once an assay is started, it cannot be polled.
3. Assay parameters cannot be easily validated.
 - An endpoint that can validate the choice of parameters for a given assay without executing it.
 - In future versions of the MCAM server, we plan to create validation endpoints that simply validate, without executing the assay.
4. The software of the MCAM Server cannot be updated through a web API call.
5. Mounting network attached storage cannot be accomplished through a web API call.