

NightFall API Server Documentation

The NightFall API server is a HTTP server implemented using [CivetWeb](#).

It allows modders to communicate to the outside world via HTTP/JSON.

Cvar Settings

sv_api

Name	sv_api
Allowed values	0 or 1
Description	Enables or disable API server. This setting is only affected at map start. It's value is effective at map start/change/restart. API server can't be started or stopped in the middle of the map. If you change it's value, change the map or restart the map.

sv_api_numthreads

Name	sv_api_numthreads
Allowed values	1 or more
Description	Specifies how many running worker threads API server should have. Ideally, usage shouldn't be more than 1.

sv_api_ports

Name	sv_api_ports
Allowed values	String containing comma separated ports eg. "8080" or "80,8080"
Description	Specifies ports that API server should listen to. Ideally, usage shouldn't need more than one port.

sv_api_acl

Name	sv_api_acl
Allowed values	String containing comma separated acls with optional subnet masks eg. "-0.0.0.0/0,+192.168.1.1/16,+127.0.0.1"
Description	Specifies access control lists(acls) that API server should use to decide to accept request from given IP addresses. This acts as a firewall for the api server. Ideally, usage shouldn't need more than default value. Negative sign means deny access, positive sign means allow. Postfixing /xx is a subnet mask. a subnet mask of /16 means ignore last 2 numbers in ip address. Eg. +192.168.1.1/16 means allow range from 192.168.0.1 to 192.168.255.254. You can use this tool to generate any wanted subnet mask.

Server Behaviour

Initialization

Once `sv_api` cvar is set to one, and upon map load, API server is started. API server will have a number of `sv_api_numthreads` idle running worker threads to serve the HTTP server. Server will listen to all specified `sv_api_ports`.

Run-time

Whilst the API server is running, modder can choose to register a callback script that will handle incoming requests. See [register_api_route](#) and [Script Usage](#) for more info.

If a request is made that matches a registered callback script(registered by [register_api_route](#)), API server will call that specified script with request information.

Please note that this process is not fast, since everything is synced with MOHAA's 20 fps server.

Usual calls made when a request arrives are as follows:

1. Request arrives.
2. API server thread receives request.
3. API server thread **waits/syncs** to add request to MOHAA thread.
4. MOHAA thread **waits/syncs** to take the next request/script passed by API server. Allowed rate is one request per frame.
5. MOHAA thread processes the request/script, waits for return value. If script is slow or has any `wait` or `waitframe` inside. return value is not returned immediately. When such case happens, script is considered pending and it's guaranteed that return value will not be fetched at least until next frame. MOHAA will handle finished pending requests at a rate of one pending request per frame.
6. MOHAA thread finally gets return value.
7. MOHAA thread **waits/syncs** to pass return value to API thread.
8. API thread was **waiting/syncing** for return value since step 4.
9. API thread finally receives return value, converts it to JSON string, and sends it as a response to the request that arrived at step 1.

Since this process is heavy for MOHAA thread, it was decided to make it at a minimal load. Therefore, limitation of one request/response per frame at steps 4 and 5 are made.

IMPORTANT NOTE: API server is an API server, it's not meant to be serving browsers, only APIs should call it.

To un-register a route, see [unregister_api_route](#).

Shutdown

At map end(or map change or map restart), API server will shutdown, clear all pending requests/responses and scripts.

Script Usage

When a user registers an API route using [register_api_route](#), the script callback function should be something like this:

```
get_handler local.query_strings:
    println "query_strings====="
    for( local.i = int 0 ; local.i < local.query_strings.size ; local.i++ )
    {
        println local.query_strings[local.i][key] " : "
    }
    local.query_strings[local.i][value];
    local.resp = "yay!"
end local.resp
```

`local.query_strings` is the query strings provided in the HTTP GET request.

Looping the query strings is done as specified in the callback example.

API server will return `local.resp` as JSON string, so expected HTTP response body is:

```
{"message":{"content":"yay!","type":"string"},"status":"success"}
```

The response JSON object has 2 keys, "message" and "status".

The possible values of "status" are as follows:

- In case of an internal error, a response of:

```
{"status":"error_internal", "message" : "Internal server error."}
```

is sent.

- In case of a not found error, a response of:

```
{"status":"error_not_found", "message" : "Not found."}
```

is sent.

- In case of success, a response of:

```
{"status":"success", "message" : variable_json_str_here}
```

where `variable_json_str_here` is a JSON string of the value of the variable returned by the callback script.

IMPORTANT NOTE: Non constant arrays have string indices instead of integer indices. Use [constarray](#) to convert an array to a constant array which will have integer indices.

IMPORTANT NOTE: Some variable types are not supported for conversion to JSON, these include:

- container, an array of listeners, \$player for example

- safe container, an array of listeners, \$player for example
- pointer, not usually used in scripts
- reference, a reference to array, usually not used in scripts