OMS (OpenM++) Web-Service

Parsa Abadi

March 27, 2025

1 Introduction

`oms` is part of the OpenM++ project and serves as a flexible JSON web-service. It can also act as a simple HTTP server for OpenM++ UI pages. Essentially, `oms` helps you:

- Browse and modify model databases.
- Run OpenM++ models stored in `models/bin`.
- Optionally serve static files (HTML, CSS, JavaScript) if you have a front-end interface.
- Provide special admin endpoints (like `/admin/kill`) for forced termination.

This doc will explains how to build `oms`, start it with the settings you need (like writing out a PID file), and how to use the "kill route" in case you need to shut it down quickly from a script or remote location.

2 Building `oms`

1. **Set up Go:** Make sure you have a recent version of Go.
2. **Get the source code:** If you haven't already, clone or download the `oms` source. Navigate to the folder that contains `oms.go`.
3. **Build:**
   ```
   go build -o oms
   ```
   This command creates an executable file named `oms`.
4. **Verify:** If the build succeeds, you'll have an `oms` binary in the current directory.

3 Running `oms`

You can run `oms` with many different arguments. For instance:

```
./oms \
    -oms.Listen localhost:4040 \
    -oms.PidSaveTo /path/to/oms.pid \
    -oms.ModelDir models/bin
```

The above command:

- Listens on `localhost:4040`.
- Writes the server's process ID to `/path/to/oms.pid`.
- Uses `models/bin` to locate model executables and databases.

3.1 Handy Arguments

`-oms.PidSaveTo` Specifies where to write `oms`'s current PID. Great for automated scripts that need to kill or manage `oms`.

`-oms.Listen` Sets the address and port for `oms` to listen on (for example, `localhost:8080` or `0.0.0.0:4040`).

`-oms.ModelDir` Points to a folder containing your model executables and databases (`models/bin` by default).

`-oms.ApiOnly` If set to `true`, only the JSON-based API endpoints are served; no HTML interface is provided.

`-oms.NoAdmin` If `true`, it disables local admin endpoints (like `/admin/kill`).

4 PID (Process ID) File

By including `-oms.PidSaveTo pidfile.txt` when launching `oms`, the service will write its process ID to the specified file *immediately* after successfully binding to the TCP port. This is useful if you need a script to:

- Automatically kill `oms` (`kill -9 <pid>` on Linux or `taskkill /PID <pid>` on Windows).
- Pre-build or post-build tasks in a CI/CD pipeline (for example, ensuring the old `oms` is dead before starting a new one).

4.1 Example

```
./oms -l localhost:4040 -oms.PidSaveTo ./oms.pid
```

Once the server is up, `oms.pid` will contain a number (like 12345). If the file can't be written, `oms` will stop itself to avoid confusion about whether or not a PID file is valid.