

# Orderman Classic Proxy

The Orderman Classic Proxy is a proxy application connecting the Orderman Emulation Apps (Max or Sol Emulator) with an existing POS system that is using the legacy Orderman drivers (Hermes SDK / omanDRV or Classic SDK / OCX) or a compatible integration.

The proxy is required for Sol/Max-Emulations connected via WiFi and/or OMB4, but not required for Sol,Max, Orderman5/7 and OMB3.

Please checkout the package for "Orderman Sol & Max Emulations" on our partner area. It will explain further details about the emulation apps and connectivity to the proxy.

In this document we will only cover the Classic Proxy application.

In the Windows (7+) operating system, Classic Proxy is installed and run as a service, while in Linux it's the system administrator's responsibility to manage the start-up and operation. The service/application is usually installed on the same machine as the POS system.

## Installation

### Windows

Before installing this update the old version (1.6.x) **must** be uninstalled. If a previous version is detected, the installer will not continue.

Run `Windows\Orderman_ClassicProxyInstaller_XXbit_2.0.1.xx.exe` (for Windows 7 run `Windows7\Orderman_ClassicProxyInstaller_XXbit_2.0.1.xx.exe` ) selecting the correct bit version (X86/32 bit or X86-64/64 bit) for your machine. The installer will start the proxy as a service and open the correct firewall ports.

### Linux

This proxy is currently only supporting Linux X86 / X86-64 platforms and the Max Emulation App, Sol Emulation is not supported.

Tested on Ubuntu 16.04 32bit and Ubuntu 22.04 64bit.

The package contains 4 parts:

- the proxy application itself `OMClassicProxyX86` when using the X86 / 32bit variant or the `OMClassicProxyX86-64` when using the X86-64/64bit variant.
- `libomchannel.so` which is the core communication library that the proxy and emulators are using to handle communication via Wi-Fi / Ethernet / Orderman OSR Radio (OMB4). It is not available for independent integrations but if you are interested you can interface the OCS SDK for Android/Java and C#.
- `runX86.sh` or `runX86-64.sh` which runs the proxy with the correct setup library path so that it can find `libomchannel.so`
- `webserver` folder contains web pages for console

## Usage

### Windows

The service will start automatically after the installation or a machine restart. Check the Windows service list to see if it's running or need to start.

### Linux

Minimum setup when the proxy is simply run on the command line on the same machine as the POS

- first unzip the file `Linux/OMClassicProxyX{architecture}.2.0.1.X.7z`
- then run the `.sh` file

```
./runX{architecture}.sh -cmd run
```

This will use default configuration values.

To run as a background service, depending on your host environment you will have different requirements to run the proxy as a daemon/service.

Some suggestions:

- Only log to file and not to the console by using `-logMode file`
- Define a different log directory to which you know the proxy process will have write permissions like `-logDirectory /var/log/omclassicproxy/`
- Tell the proxy to create a pid file in a location it has write permission to with `-pidfile /var/run/omclassicproxy.pid`

The full example would look like this:

```
./runX{architecture}.sh -cmd run -logMode file -logDirectory /var/log/omclassicproxy/ -pidfile /var/run/omclassicpro
```

You should be able to embed this in your target platform init environment like systemd or OpenRC or any other you are using.

## Configuration

For Windows the proxy configuration is persisted in a json file named `ClassicProxyService.config.json` in `NetworkService` app data folder and in Linux configuration is accepted as args in command line

This is as this example of default configuration created at startup as a Windows service at first time running:

```
{
  "ClassicEnabled": true,
  "HermesEnabled": true,
  "OcsPort": 24998,
  "OmanDRVPort": 25000,
  "ClassicPort": 24999,
  "WebServerPort": 5051,
  "PosIPClassic": "::1",
  "PosIPHermes": "127.0.0.1",
  "DisableUniqueSourceAddressForSol": false,
  "ScnAgentSyslogPort": 514,
  "LogDirectory": "./logs",
  "LogMaxSize": 10,
  "LogMaxBackups": 3,
  "LogLevel": "debug",
  "LogMode": "all",
  "PidFile": ""
}
```

A list of configuration voices with corresponding command argument:

- `ClassicEnabled ( -enableClassic=true )`: true/false to enable connections and monitoring for a POS with Max clients/OCX implementation
- `HermesEnabled ( -enableHermes=true )`: true/false to enable connections and monitoring for a POS with Sol clients/omanDRV - WINDOWS SUPPORT ONLY
- `OcsPort ( -ocsPort=24998 )`: the incoming TCP port that the mobile devices Sol/Max client apps will use to connect to the proxy
- `OmanDRVPort ( -hermesPort=25000 )`: the port on which your POS driver will listen for incoming Sol connections
- `ClassicPort ( -classicPort=24999 )`: the port on which your POS driver will listen for incoming Max connections
- `WebServerPort ( -webServerPort=5051 )`: port to access the web console via a browser on localhost
- `PosIPClassic ( -classicIP=: :1 )`: ip string on which your POS driver will listen for incoming Max connections
- `PosIPHermes ( -hermesIP=127.0.0.1 )`: ip string on which your POS driver will listen for incoming Sol connections
- `DisableUniqueSourceAddressForSol ( disableUniqueSourceAddressForSol=false )`: true/false to indicate if the device IP addresses are differentiated by the device serial number
- `ScnAgentSyslogPort` (not available as arg): for Windows only, log can be transfer with SCN Win Agent

- LogDirectory ( -logDirectory=./logs ): where file logs are saved (for Windows it's relative to appdata of NetworkService user, for Linux where app is launched)
- LogMaxSize ( -logMaxSize=10 ): the maximum size of the log file in MB
- LogMaxBackups ( -logMaxBackups=3 ): the maximum number of log files
- LogLevel ( -logLevel=debug ): the log level used and the possible values are *debug, info, warn, error*
- LogMode ( -logMode=all ): choose where to log, possible values are *console, file, all*
- PidFile ( -pidfile="" ): for Linux only, it persits a process ID to prevent more than one instance from running

This configuration file can be found in Windows on AppData folder of NetworkService user at

C:\Windows\ServiceProfiles\NetworkService\AppData\Roaming\Orderman\ClassicProxy\ .

As mentioned in the Usage chapter, Linux uses a command line with arguments to run applications.

## Web Console

This web console is an easy way to monitor aspects of the running Classic Proxy, such as the configuration, the number of connected devices and their serial numbers, the logs and the command to restart the service.

Open a browser and enter the URL `http://localhost:5051` (or `http://localhost:{webServerPort}` ) (or the value in the WebServerPort configuration property) to access the console. You can click on the menu on the right of the header to navigate to the different views: Configuration, Devices and Logs.

To prevent misuse, values on the configuration page cannot be edited.

On the Devices page, you will find a password protected button to restart the service. Ask an administrator to apply the command.

On the Logs page, click on the links to download the log file.