

---

# **Xonotic Server Management Suite Documentation**

***Release 0.3.0***

**Tyler Mulligan**

December 23, 2016



<b>1</b>	<b>Intro</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
2.1	Requirements . . . . .	5
2.2	Install . . . . .	5
<b>3</b>	<b>Configuration</b>	<b>7</b>
3.1	Defining Servers . . . . .	7
3.2	Custom Server Configuration . . . . .	8
<b>4</b>	<b>Usage</b>	<b>9</b>
4.1	With Docker . . . . .	9
4.2	Without Docker . . . . .	9
<b>5</b>	<b>API</b>	<b>11</b>
5.1	Server . . . . .	11
5.2	Engine . . . . .	12
5.3	Engines . . . . .	12
5.4	Command . . . . .	13
<b>6</b>	<b>Tests</b>	<b>15</b>
<b>7</b>	<b>License</b>	<b>17</b>
<b>8</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Python Module Index</b>	<b>21</b>



Xonotic Server Management Suite is a collection of tools and best practices for managing infrastructure, tests, and deployments of Xonotic game servers.

```
version: '1'
servers:
  vanilla:
    title: "-z- Simple vanilla"
    motd: "Welcome to ${hostname} | Owner: -z-"
    port: 26000
    maxplayers: 16
    net_address: ""
    use_smbmod: false
    exec: ./all run dedicated +serverconfig vanilla.cfg
  insta:
    title: "(SMB) Instagib+Hook USA"
    motd: "Welcome to ${hostname} | Owner: AllieWay | Admins: Mario, muffin, -z- | Hello from xsms"
    port: 26010
    maxplayers: 64
    net_address: ""
    use_smbmod: true
    exec: ./all run dedicated -game modpack -game data_csprogs -game data_insta -sessionid insta +se
```

### Features

- Dockerized [Xonotic](#) git and stable builds
- [SMB configurations](#) integrated
- [SMB Modpack](#) support
- [Xonotic Map Manager](#) integration
- Define your servers in *YAML* or *cfg*
- **Works with your existing workflow using:**
  - [screen](#)
  - [tmux](#)
  - [supervisor](#)
  - [docker](#)

### Get Started



---

## Intro

---

Game server administration shouldn't be a full-time job, it should be enjoyable, like the game. Great ideas should be tested, saved and reproducible. Xonotic Server Management Suite works with your existing workflow by *automating the boring stuff*.

Generate configurations for popular server management software tools, henceforth referred to as *engines*, through a normalized YAML format without compromise:

- `screen`
- `tmux`
- `supervisor`
- `docker`

Optionally, use the built in `xsms`, and `xmm` tools to manage your servers and their engines.

The configuration files are generated with a combination of `yml`, `conf` and `cfg` and provide many opportunities to inject existing assets and configurations for any supported engine.

Start with a simple `servers.yml` to define a single server that generates a server configuration for a vanilla server, `vanilla.cfg`. This gets put in `~/.xonotic/data/server.pk3dir`, which to the DarkPlaces engine appears as `~/.xonotic/data`, making it available to reference as `vanilla.cfg`

```
# This file is read from ~/.xsms/servers.yml make sure that's where you are editing it
version: '1'
servers:
  vanilla:
    title: "-z- Simple vanilla"
    motd: "Welcome to ${hostname} | Owner: -z-"
    port: 26000
    maxplayers: 16
    net_address: ""
    exec: ./all run dedicated +serverconfig vanilla.cfg
```

The following command will generate the `vanilla.cfg` file the `exec` line above references:

```
xsms servers build
```

The contents of that generated file will look similar to:

```
// Last Generated: 2016-10-30 19:39:17.026331
hostname "-z- Simple vanilla"
sv_motd "Welcome to ${hostname} | Owner: -z-"
port 26000
```

```
maxplayers 16  
net_address ""
```

Start up the servers with your engine of choice, for example:

```
xsms servers -e screen start
```

.

- genindex
- modindex
- search



---

## Installation

---

### 2.1 Requirements

- Python 3
- A supported engine (screen, tmux, supervisor)

#### 2.1.1 With Docker

- docker
- docker-compose

#### 2.1.2 Without Docker

- **Local installation of Xonotic:**
  - [Xonotic releases](#) are available at Xonotic.org
  - Instructions for git are [available in the Xonotic wiki](#)

Use the Dockerfiles in `docker/containers` for inspiration.

### 2.2 Install

Install with `setuptools`:

```
python3 setup.py install # this clones the server configs and modpack
xsms smbmod init        # setup SMB modpack and assets (optional)
```

If using docker, all custom server assets go in `~/xonotic-smb` on the host which gets mounted to `~/xonotic` in the containers.

- `genindex`
- `modindex`
- `search`



---

## Configuration

---

The defaults should work out of the box, if you want to make changes, edit the `~/.xsms.cfg` file.

```
[default]
# Xonotic
xonotic_root = /opt/Xonotic
xonotic_userdir = ~/.xonotic
xonotic_server_pk3dir = ~/.xonotic/servers.pk3dir
xonotic_servers = ~/.xsms/servers.yml
xonotic_server_template = ~/.xsms/templates/xonotic/xonotic.server.cfg.tpl
xonotic_smbmod_server_template = ~/.xsms/templates/xonotic/xonotic.smbmod-server.cfg.tpl

# Engines
supervisor_server_template = ~/.xsms/templates/engines/supervisor.server.conf.tpl
supervisor_conf_template = ~/.xsms/templates/engines/supervisor.conf.tpl
supervisor_conf = ~/.xsms/generated/engines/supervisor.conf

# SMB
smb_init_script = bin/smb_init.sh
smb_update_script = ~/.xonotic-smb/modpack/update.sh
smb_build_script = ~/.xonotic-smb/modpack/build.sh
smb_cache_path = ~/.xonotic-smb/modpack/.cache
smb_data_csprogs = ~/.xonotic-smb/data_csprogs
```

### 3.1 Defining Servers

XSMS provides a YAML specification for defining the basic meta information for servers.

You can think of this as *xonotic-compose*.

#### Example:

```
# This file is read from ~/.xsms/servers.yml make sure that's where you are editing it
version: '1'
servers:
  insta:
    title: "(SMB) Instagib+Hook USA"
    motd: "Welcome to ${hostname} | Owner: AllieWay | Admins: Mario, muffin, -z- | Hello from xsms"
    port: 26010
    maxplayers: 64
    net_address: ""
    use_smbmod: true
```

```
exec: ./all run dedicated -game modpack -game data_csprogs -game data_insta -sessionid insta +se
overkill:
  title: "(SMB) Overkill USA"
  motd: |
    This is my long message of the day.
    On multiple lines
  port: 26004
  maxplayers: 32
  net_address: ""
  use_smbmod: true
exec: ./all run dedicated -game modpack -game data_csprogs -game data_overkill -sessionid overki
```

This YAML file will generate a xonotic-compatible `.cfg` in `~/.xsms/generated/servers/`.

## 3.2 Custom Server Configuration

The configuration files for `xsms` exist in `~/.xsms`, below is a tree showing demonstrating the structure:

```
~/.xsms
-- generated
|   -- engines
|   |   -- supervisor.conf
|   -- servers
|   |   -- insta.cfg
|   |   -- overkill.cfg
|   |   -- vanilla.cfg
-- servers.yml
-- templates
  -- engines
  |   -- supervisor.conf.tpl
  |   -- supervisor.server.conf.tpl
  -- servers
  |   -- insta.cfg.tpl
  -- xonotic
  |   -- xonotic.server.cfg.tpl
  |   -- xonotic.smbmod-server.cfg.tpl
```

The generated folder is build artifacts that can be deleted and regenerated. These are based on the files in templates.

Custom server templates are defined in `~/.xsms/templates/servers/<server_name>.cfg.tpl` where `<server_name>` corresponds with the name of the server defined in the YAML. See the `tests` folder for an example of a custom `.cfg.tpl`.

Custom engine configs likewise belong in `~/.xsms/templates/servers/<engine_name>`. Currently only `supervisord` has a custom template.

- `genindex`
- `modindex`
- `search`

---

## Usage

---

### 4.1 With Docker

The easiest way to get started is with docker. The `docker-compose.yml` file contains containers for running either `xonotic_git`, `xonotic_stable` or both.:

```
docker-compose up          # this brings up the arch described in docker-compose.yml
# or
docker-compose up xonotic_git # this brings up only the xonotic_git container
docker-compose down        # this takes it down
```

#### 4.1.1 Using XMM to manage maps

The link between XMM and servers is defined in `docker/containers/xonotic/xmm/servers.json`.

In the example below, the server `insta` is used.:

```
docker-compose exec xonotic_git /bin/bash # connect to the docker container
xmm update                               # get the latest package list
xmm -s insta discover                     # finds any maps in this server's data dir
xmm -s insta install eggandscrambled.pk3 # install a new map
xmm -s insta list                         # list all maps tracked for this server
```

### 4.2 Without Docker

Without docker, XSMS can manage game servers a few different ways using `xsms servers start` to start up your servers defined in `~/.xsms/servers.yml`. Supported (or planned) methods include: `screen`, `tmux` for interactive management and `supervisor`, `circus` for daemon management. If you want simple map management, XMM Can be installed separately.

For daemons, conf files need to be generated with `xsms servers build`.

- `genindex`
- `modindex`
- `search`



## 5.1 Server

```
class xsms.server.Server (name, exec, title, motd= 'Welcome to ${hostname}!', port= '26000', maxplayers= '32', net_address= '', use_smbmod= True)
```

This class defines the Server object

### Parameters

- **name** (str) – A name for the collection of servers
- **exec** (str) – The executable line for when this server starts
- **title** (str) – The title of this server
- **motd** (str) – The message of the day for this server
- **port** (str) – The port this server is served on
- **maxplayers** (int) – The maximum number of players of this server
- **net\_address** (str) – The ip address of this server
- **use\_smbmod** (bool) – Whether this server uses SMB Modpack or not

Returns object `Server`

### Example

```
>>> from xsms.server import Server
>>> server = Server(name='insta', exec='./all run dedicated +serverconfig vanilla.cfg', title='M
```

```
class xsms.server.Servers (name, servers=None)
```

This class defines the Servers object

### Parameters

- **name** (str) – A name for the collection of servers
- **servers** (list) – A list of Server objects

Returns object `Servers`

### Example

```
>>> from xsms.server import Servers, Server
>>> server = Server(name='insta', exec='./all run dedicated +serverconfig vanilla.cfg', title='M
>>> servers = Servers(name='Xonotic Server Collection', servers=[server])
```

## 5.2 Engine

**class** `xsms.engine.Engine` (*conf*)

This is the base class for engines, the `read_servers_manifest` is fired on init reading in `conf['servers_manifest']`.

**Parameters** `conf` (dict) – A dict `conf` from `config.py`

**Returns** `Engine`

**read\_servers\_manifest** (*filename*)

This reads in a `servers.yml` file, turns it into a dict. sets `self.servers` to that value and then returns it.

**Parameters** `filename` (str) – A yaml file using the syntax of `servers.yml`

**Returns** `Servers`

```
>>> from xsms.engine import Engine
>>> from xsms.config import conf
>>> session = Engine(conf=conf)
>>> servers = session.read_servers_manifest(filename=conf['servers_manifest'])
```

## 5.3 Engines

**class** `xsms.engines.screen.Session` (*conf*)

This is the engine class for screen

**start** (*xonotic\_root*, *servers\_manifest=None*)

This engine enables programmatic control of screen

**Parameters**

- **xonotic\_root** (str) – The directory for the exec command
- **servers\_manifest** (str) – A file in the `servers.yml` format

```
>>> from xsms.engines.screen import Session as screen
>>> from xsms.config import conf
>>> session = screen(conf=conf)
>>> servers = session.start(xonotic_root=conf['xonotic_root'])
```

**class** `xsms.engines.tmux.Session` (*conf*)

This is the engine class for tmux

**start** (*xonotic\_root*, *servers\_manifest=None*)

This engine enables programmatic control of tmux

**Parameters**

- **xonotic\_root** (str) – The directory for the exec command
- **servers\_manifest** (str) – A file in the `servers.yml` format

```
>>> from xsms.engines.tmux import Session as tmux
>>> from xsms.config import conf
>>> session = tmux(conf=conf)
>>> servers = session.start()
```



---

**Note:** This tries to create a new session with Popen() but it does not always work. It is therefore better to already have a tmux session running before using this command.

---

**class** `xsms.engines.supervisor.Session(conf)`

This is the engine class for supervisor

**start** (*servers\_manifest=None*)

This engine enables pass-through control of servers managed by supervisor.

**Parameters** `servers_manifest` (str) – A file in the `servers.yml` format

```
>>> from xsms.engines.supervisor import Session as supervisor
>>> from xsms.config import conf
>>> session = supervisor(conf=conf)
>>> servers = session.start()
```

## 5.4 Command

**class** `xsms.command.Command(conf)`

This class handles the `xsms servers` subcommand

**Parameters** `conf` (dict) – The conf dictionary from `config.py`

**Returns object** `Command` The session for a `xsms servers` subcommand

**Example**

```
>>> from xsms.command import Command
>>> from xsms.config import conf
>>> session = Command(conf=conf)
```

**generate\_engine\_configs()**

This method generates engine configs

**generate\_server\_configs()**

This method generates `cfg server` configs from YAML

**start** (*engine='screen'*)

This method starts servers defined in `~/.xsms/servers/yml` with an engine

**Parameters** `engine` (str) – A supported engine (screen, tmux, supervisor)

Available engines:

- screen
- tmux
- supervisor

**Example**

```
>>> from xsms.command import Command
>>> from xsms.config import conf
>>> session = Command(conf=conf)
>>> session.start(engine='tmux')
```

- genindex

- [modindex](#)
- [search](#)

---

# Tests

---

Unit tests can be run with `py.test` and coverage tests with `tox`:

```
make tests
make tests-coverage
make lint
make clean
```

- `genindex`
- `modindex`
- `search`



---

### License

---

Copyright (c) 2016 Tyler Mulligan ([z@xnz.me](mailto:z@xnz.me)) and contributors.

Distributed under the MIT license. See the `LICENSE` file for more details.

- `genindex`
- `modindex`
- `search`



---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`





## X

- `xsms.command`, [13](#)
- `xsms.engine`, [12](#)
- `xsms.engines.screen`, [12](#)
- `xsms.engines.supervisor`, [13](#)
- `xsms.engines.tmux`, [12](#)
- `xsms.server`, [11](#)



## C

Command (class in `xsms.command`), 13

## E

Engine (class in `xsms.engine`), 12

## G

`generate_engine_configs()` (`xsms.command.Command` method), 13

`generate_server_configs()` (`xsms.command.Command` method), 13

## R

`read_servers_manifest()` (`xsms.engine.Engine` method), 12

## S

Server (class in `xsms.server`), 11

Servers (class in `xsms.server`), 11

Session (class in `xsms.engines.screen`), 12

Session (class in `xsms.engines.supervisor`), 13

Session (class in `xsms.engines.tmux`), 12

`start()` (`xsms.command.Command` method), 13

`start()` (`xsms.engines.screen.Session` method), 12

`start()` (`xsms.engines.supervisor.Session` method), 13

`start()` (`xsms.engines.tmux.Session` method), 12

## X

`xsms.command` (module), 13

`xsms.engine` (module), 12

`xsms.engines.screen` (module), 12

`xsms.engines.supervisor` (module), 13

`xsms.engines.tmux` (module), 12

`xsms.server` (module), 11