Xonotic Map Manager Documentation

Release 0.8.1.dev0

Tyler Mulligan

December 22, 2016

Contents

A command-line package manager for the xonotic-map-repository project.

Features

- Quickly install maps from the command-line
- Discovery and optional addition of your existing maps to XMM's library
- Repository-less install of map packages from a URL
- *Multi-Server* support (track different maps per server)
- *Multi-repo* support
- Plugin system
- Developer API, use xmm as a module

Get Started

CHAPTER 1

Intro

Xonotic Map Manager is the command-line package manager component of the xonotic-map-repository project. Whether you're a server admin looking to manage map lists, or a player trying to learn more about their map collection, `xmm` is a tool to help you do this quickly and efficiently.

Use it like `apt`:

```
xmm update
xmm install eggandscrambled.pk3
xmm list
```

If you get stuck, try using a `xmm -h`, or reference the *Usage* page.

By default, the unofficial, general purpose Xonotic map repository, xonotic.co, is enabled. Advanced users can add additional repositories and/or host their own.

- genindex
- modindex
- search

CHAPTER 2

Installation

2.1 Requirements

- Python 3

2.1.1 Debian/Ubuntu

If you do not already have **pip** and **setuptools** for Python3:

```
sudo apt install python3-pip python3-setuptools
```

2.2 Install

Install using pip:

```
pip3 install xmm --user
```

If you get an error trying to run `xmm`, you probably need `$HOME/.local/bin` in your path, put the following in your `~/.bashrc` or `~/.zshrc` etc:

```
export PATH=$PATH:$HOME/.local/bin
```

Alternatively, install the development version manually with setuptools:

```
git clone https://github.com:z/xonotic-map-manager.git
cd xonotic-map-manager
python3 setup.py install
```

- genindex
- modindex
- search

CHAPTER 3

Configuration

3.1 Core

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

```
# This file is read from ~/.xmm.ini, make sure that's where you are editing it
[xmm]

# Where should xmm manage maps?
target_dir = ~/.xonotic/data/

# Default repo if no sources specified
download_url = http://dl.xonotic.co/
api_data_url = http://xonotic.co/resources/data/maps.json
api_data_file = ~/.xmm/maps.json

# This is only preference
use_curl = False

# configuration of servers to use with multiple servers
servers_config = ~/.xmm/servers.json

# configuration of repositories
sources_config = ~/.xmm/sources.json
```

3.2 Logging

Logging can be configured in `~/.xmm/xmm.logging.ini`, again, the defaults should be sufficient.

```
# ~/.xmm/xmm.logging.ini
[loggers]
keys = root

[logger_root]
level    = NOTSET
handlers = stream, info

[handlers]
keys = stream, info
```

```
[handler_stream]
class = StreamHandler
args = (sys.stdout,)
level = ERROR
formatter = generic

[handler_debug]
class = handlers.RotatingFileHandler
formatter = generic
level = DEBUG
args = ('%(log_filename)s', 'a', 50000000, 5)

[handler_info]
class = handlers.RotatingFileHandler
formatter = generic
level = INFO
args = ('%(log_filename)s', 'a', 50000000, 5)

[formatters]
keys = generic

[formatter_generic]
format = %(asctime)s %(levelname)-5.5s [%(name)s] [%(threadName)s] %(message)s
datefmt = %Y-%m-%d %H:%M:%S
class = logging.Formatter
```

3.3 Multi-Server

xmm can facilitate the management of multiple servers with `~/.xmm/servers.json` which defines the configure of settings, example below:

```
{
  "myserver1": {
    "target_dir": "~/.xonotic/myserver1/data/",
    "library": "~/.xmm/myserver1/library.json",
    "sources": "~/.xmm/sources.json"
  },
  "myserver2": {
    "target_dir": "~/.xonotic/myserver2/data/",
    "library": "~/.xmm/myserver2/library.json",
    "sources": "~/.xmm/myserver2/sources.json"
  }
}
```

3.4 Multi-repo

xmm can use multiple repositories, edit the `~/.xmm/sources.json` file to configure them, example below:

```
{
  "default": {
    "download_url": "http://dl.xonotic.co/",
    "api_data_file": "~/.xmm/maps.json",
    "api_data_url": "http://xonotic.co/resources/data/maps.json"
```

```
    }
}
```

- genindex
- modindex
- search

CHAPTER 4

Usage

4.1 Basic Usage

CLI help docs for `xmm`:

```
usage: xmm [-h] [--version] [-S [SERVER]] [-T [TARGET]] [-R [REPOSITORY]]
           {search,install,remove,discover,list,show,export,servers,repos,update,hello}
           ...

Xonotic Map Manager is a tool to help manage Xonotic maps

positional arguments:
  {search,install,remove,discover,list,show,export,servers,repos,update,hello}
    search              search for maps based on bsp names
    install             install a map from the repository, or specify a URL.
    remove              remove based on pk3 name
    discover            discover packages in a target directory
    list                list locally installed packages
    show                show details of remote or locally installed packages
    export              export locally managed packages to a file
    servers             subcommands on servers described in servers.json
    repos               subcommands on repos described in sources.json
    update              update sources json
    hello               hello is an example plugin

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -S [SERVER], --server [SERVER]
                        target server as defined in servers.json
  -T [TARGET], --target [TARGET]
                        target directory
  -R [REPOSITORY], --repository [REPOSITORY]
                        repository to use (defaults to all available)
```

4.1.1 Searching

by bsp name:

```
xmm search snowdance
Searching for packages with the following criteria:
```

```
bsp: snowdance
---

snowdance2.pk3 [snowdance2]
http://dl.xonotic.co/snowdance2.pk3

snowdance_xon.pk3 [snowdance2]
http://dl.xonotic.co/snowdance_xon.pk3
---
Total packages found: 2
```

with pk3 name and detailed results:

```
xmm search --pk3 bloodrage_v2.pk3 --long
Using repo 'default'
Searching for packages with the following criteria:
pk3: bloodrage_v2.pk3
---

         pk3: bloodrage_v2.pk3
         bsp: bloodrage_v2
       title:  Bloodrage
 description:  Small, brutal and violent 1on1 map
      author:  Cortez and FruitieX
      shasum: 488b05976e73456bf6f9833e353f72d3a8d0cbce
      shasum: bloodrage_v2.pk3
        date: 2009-10-17
        size: 1MB
          dl: http://dl.xonotic.co/bloodrage_v2.pk3
---
Total packages found: 1
```

Inline help is available on all sub-commands:

```
xmm search -h
usage: xmm search [-h] [--gametype [GAMETYPE]] [--pk3 [PK3]] [--title [TITLE]]
                  [--author [AUTHOR]] [--shasum [SHASUM]] [--long] [--short]
                  [--color]
                  [string]

positional arguments:
  string                bsp name found in a package, works on packages with
                        many bsps

optional arguments:
  -h, --help            show this help message and exit
  --gametype [GAMETYPE]
                        filter by gametype
  --pk3 [PK3]           filter by pk3 name
  --title [TITLE]       filter by title
  --author [AUTHOR]     filter by author
  --shasum [SHASUM]     filter by shasum
  --long, -l            show long format
  --short, -s           show short format
  --color, -c           highlight search term in results
```

4.1.2 Installing from the repository

Installing a new pk3:

```
xmm install snowdance_xon.pk3
Installing map: snowdance_xon.pk3
...100%, 5 MB, 2438 KB/s, 2 seconds passed. Done.
```

You will be prompted to overwrite an existing pk3:

```
xmm install snowdance_xon.pk3
Installing map: snowdance_xon.pk3
snowdance_xon.pk3 already exists.
continue? [y/N] N
Canceled.
```

You cannot install a pk3 that doesn't existent in the repo:

```
Installing map: fake.pk3
package does not exist in the repository. cannot install.
```

Example below is also showing the use of curl instead of python's urllib if you prefer:

```
xmm install http://somerepo.org/snowdance2.pk3
Adding map: http://somerepo.org/snowdance2.pk3
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 5530k  100 5530k    0     0   205k      0  0:00:26  0:00:26 --:--:--  179k
Done.
```

You can install from any URL (buy lack detailed meta information about maps):

```
xmm install http://somerepo.org/some-other-map.pk3
Adding map: http://somerepo.org/snowdance2.pk3
...100%, 5 MB, 2438 KB/s, 2 seconds passed. Done.
```

4.1.3 Removing

Remove a map:

```
xmm remove snowdance2.pk3
Removing map: snowdance2.pk3
Done.
```

You cannot remove a map that doesn't exist:

```
xmm remove snowdance2.pk3
Removing package: snowdance_xon.pk3
package does not exist or is not tracked. try removing using full path if not tracked.
```

4.1.4 Discover

You can pulled additional meta information about pk3s and verify their shasums against the repo with the discover command.

A summary of discovered packages:

```
xmm -S myserver1 discover

sxb1_testing_6.pk3 [sxb1_-1, sxb1_-2, sxb1_-3, sxb1_1-1, sxb1_1-2, sxb1_1-3, sxb1_1-4, sxb1_2-1, sxb1
http://dl.xonotic.co/sxb1_testing_6.pk3

bloodprisonctf.pk3 [bloodprisonctf]
http://dl.xonotic.co/bloodprisonctf.pk3
bloodprisonctf.pk3 hash does not match repository's

gasoline_02.pk3 [gasoline_02, gasoline_3teams_02, gasoline_4teams_02, gasoline_noteams_02]
http://dl.xonotic.co/gasoline_02.pk3

testie3.pk3 [testie3]
http://dl.xonotic.co/testie3.pk3

map-derail_v1r5.pk3 [derail_v1r5]
http://dl.xonotic.co/map-derail_v1r5.pk3
map-derail_v1r5.pk3 hash does not match repository's

disarray_v1r2.pk3 [disarray_v1r2]
http://dl.xonotic.co/disarray_v1r2.pk3

eggandscrambled.pk3 [eggandscrambled]
http://dl.xonotic.co/eggandscrambled.pk3
```

Add discovered maps:

```
xmm -S myserver1 discover --add
```

4.1.5 List Map Packages

simple list:

```
xmm list

gasoline_02.pk3 [gasoline_02, gasoline_3teams_02, gasoline_4teams_02, gasoline_noteams_02]
http://dl.xonotic.co/gasoline_02.pk3

dance.pk3 [dance]
http://dl.xonotic.co/dance.pk3

Total packages found: 2
```

detailed list:

```
xmm list -l

         pk3: gasoline_02.pk3
         bsp: gasoline_02
       title:  Gasoline Powered
 description:  Retextured and glowy
      author:  FruitieX, Kid, Mario
         bsp: gasoline_3teams_02
       title:  Gasoline Powered
 description:  Retextured and glowy with 3 teams
      author:  FruitieX, Kid, Mario, Freddy
         bsp: gasoline_4teams_02
```

```
       title:  Gasoline Powered
 description:  Retextured and glowy with 4 teams
      author:  FruitieX, Kid, Mario
         bsp: gasoline_noteams_02
       title:  Gasoline Powered - Teamless
 description:  Retextured and glowy
      author:  FruitieX, Kid, Mario
      shasum: 099b0cc16fe998e5e29893dbecd5673683a5b69d
        date: 2015-10-17
        size: 14MB
          dl: http://dl.xonotic.co/gasoline_02.pk3

         pk3: dance.pk3
         bsp: dance
       title:  <TITLE>
 description:  <DESCRIPTION>
      author:  <AUTHOR>
      shasum: ef00d43838430b2d1673f03bbe1440eef100ece6
        date: 2008-03-16
        size: 7MB
          dl: http://dl.xonotic.co/dance.pk3


Total packages found: 3
```

4.1.6 Show Map Package Details

simple:

```
xmm show dance.pk3

dance.pk3
dance
http://dl.xonotic.co/dance.pk3
```

detailed:

```
xmm show dance.pk3 -l

         pk3: dance.pk3
         bsp: dance
       title: <TITLE>
 description: <DESCRIPTION>
      author: <AUTHOR>
      shasum: ef00d43838430b2d1673f03bbe1440eef100ece6
        date: 2008-03-16
        size: 7MB
          dl: http://dl.xonotic.co/dance.pk3
```

4.1.7 Export

You can export local maps from your library, or maps from a repository in different formats:

```
usage: xmm export [-h] [--format {json,shasums}] {local,repos} [filename]

positional arguments:
```

```
  {local,repos}         what context to export?
  filename              filename to export to

optional arguments:
  -h, --help            show this help message and exit
  --format {json,shasums}, -f {json,shasums}
```

For example, export a maplist to a map-repo-friendly json format:

```
% xmm export local test.json -f json
% cat test.json
[{"mapinfo": ["maps/dance.mapinfo"], "date": 1205715512, "title": "<TITLE>", "radar": [], "waypoints"
```

Or a list of pk3s and their respective shasums:

```
xmm export repos -f shasums
tail all-repos-maps.json.shasums
d88957aeff231471453f41e8ab2dad326b1875b2 acrossanocean12.pk3
e3059ee1979985151fade8b0d317422dc71ec9bb cloisterctf_vehicles.pk3
3f15789118762f469c9179f8f799747dced948cb dastower_vehicles.pk3
5af57ca19b69560cd9b00f67cbbb7ee4526bc8ac duster_mod_01.pk3
e06724125a3438a23bad4f0d3ec3b6a5ce89666a greatwall_remix_vehicles.pk3
abc9e153c37784563e4e3c2669cc88af05649399 ons-reborn_vehicles.pk3
```

4.1.8 Servers

List servers with `xmm servers list`:

```
xmm servers list
myserver2
myserver1
```

4.1.9 Repos

List repositories with `xmm repos list`:

```
xmm repos list
default
gpl_only
```

4.1.10 Update

Get the latest list of maps from the repository:

```
xmm update
Updating sources json.
...100%, 7 MB, 2559 KB/s, 3 seconds passed. Done.
```

4.2 Advanced Usage

4.2.1 Multi-server support

xmm can facilitate the management of multiple servers with a `~/.xmm/servers.json` file to configure their settings, example below:

```
{
  "myserver1": {
    "target_dir": "~/.xonotic/myserver1/data/",
    "library": "~/.xmm/myserver1/library.json",
    "sources": "~/.xmm/sources.json"
  },
  "myserver2": {
    "target_dir": "~/.xonotic/myserver2/data/",
    "library": "~/.xmm/myserver2/library.json",
    "sources": "~/.xmm/myserver2/sources.json"
  }
}
```

An example is available in `./config/example.servers.json`

To use these servers, use the `-S` flag to target the server:

```
xmm -S myserver1 install dance.pk3
xmm -S myserver1 list
xmm -S myserver1 remove dance.pk3
```

4.2.2 Multi-repository support

xmm can use multiple repositories, edit the `~/.xmm/sources.json` file to configure them, example below:

```
{
  "default": {
    "download_url": "http://dl.xonotic.co/",
    "api_data_file": "~/.xmm/maps.json",
    "api_data_url": "http://xonotic.co/resources/data/maps.json"
  }
}
```

An example is available in `./config/example.sources.json`

To use these servers, use the `-R` flag to target the server:

```
xmm -R myrepo install dance.pk3
xmm -R myrepo list
xmm -R myrepo remove dance.pk3
```

4.2.3 Targeting Directories

Sometimes you may want to install a package to an arbitrary directory:

```
xmm -T /path/to/directory/ install dance.pk3
```

Note: This install will not be tracked in the library.

- genindex
- modindex
- search

CHAPTER 5

Upgrading

5.1 0.7.0 -> 0.8.0

5.1.1 Configuration Updates

xmm.cfg -> xmm.ini

- Renamed section: `[default]` to `[xmm]`
- Renamed keys:
  - `map_dir` -> `target_dir`
  - `repo_url` -> `download_url`
  - `api_data` -> `api_data_file`
  - `servers` -> `servers_config`
- New keys:
  - `servers_config` with default `~/.xmm/servers.json`

servers.json

The `package_db` key has been dropped, `library` and `sources` have been added.

Pickle is no longer used for serialization, data is now stored as JSON:

```
{
  "myserver1": {
    "target_dir": "~/.xonotic/myserver1/data/",
    "library": "~/.xmm/myserver1/library.json",
    "sources": "~/.xmm/sources.json"
  },
  "myserver2": {
    "target_dir": "~/.xonotic/myserver2/data/",
    "library": "~/.xmm/myserver2/library.json",
    "sources": "~/.xmm/myserver2/sources.json"
  }
}
```

sources.json *[NEW]*

This is a new feature that enables support for more than one repository:

```
{
  "default": {
    "download_url": "http://dl.xonotic.co/",
    "api_data_file": "~/.xmm/maps.json",
    "api_data_url": "http://xonotic.co/resources/data/maps.json"
  }
}
```

For more information please see the *Configuration* page.

xmm.logging.ini *[NEW]*

Logging is now configurable through `~/.xmm/xmm.logging.ini`.

For more information please see the *Configuration* page.

5.1.2 Migrating Your Library

The easiest way to migrate your library is with:

```
xmm discover --add
```

- genindex
- modindex
- search

CHAPTER 6

Developers

6.1 JSON structure

Same structure used by xonotic-map-repository:


```
{
  "data": [
    {
      "date": 1453749340,
      "filesize": 7856907,
      "bsp": {
        "vapor_alpha_2": {
          "radar": "gfx/vapor_alpha_2_mini.tga",
          "waypoints": "",
          "title": "Vapor",
          "description": "Such CTF. Many Vehicles. Wow.",
          "map": "maps/vapor_alpha_2.map",
          "entities": {
            "info_player_deathmatch": 4,
            "info_player_team1": 11,
            "info_player_team2": 11,
            "item_armor_big": 10,
            "item_armor_large": 4,
            "item_armor_medium": 16,
            "item_armor_small": 124,
            "item_bullets": 10,
            "item_cells": 14,
            "item_flag_team1": 1,
            "item_flag_team2": 1,
            "item_health_large": 6,
            "item_health_medium": 30,
            "item_health_mega": 2,
            "item_health_small": 100,
            "item_invincible": 1,
            "item_rockets": 20,
            "item_strength": 1,
            "weapon_crylink": 4,
            "weapon_devastator": 6,
            "weapon_electro": 2,
            "weapon_grenadelauncher": 6,
            "weapon_hagar": 4,
            "weapon_machinegun": 6,
            "weapon_vortex": 4
```

```
            },
            "mapinfo": "maps/vapor_alpha_2.mapinfo",
            "author": "-z-",
            "gametypes": [
              "ctf",
              "DM"
            ],
            "license": true,
            "mapshot": "maps/vapor_alpha_2.jpg"
          }
        },
        "shasum": "3df0143516f72269f465070373f165c8787964d5",
        "pk3": "map-vapor_alpha_2.pk3"
      }
  ]
}
```

6.2 Plugin System

Checkout the examples in the `./xmmc/plugins` directory.

```
from xmm.plugins import pluginbase
from xmm.util import zcolors
from xmm.util import cprint

config = pluginbase.get_config()


def get_args():
    command='hello'
    command_help={'help': 'hello is an example plugin'}
    args=['-f', '--foo']
    kwargs={'type': int, 'nargs': '?', 'help': 'this is a help line'}
    return command, command_help, args, kwargs


def run():
    print("Hello from a plugin!")
    cprint("I share the xmm util module", style='INFO')
    print("{}Look, I have access to the config: {}".format(zcolors.SUCCESS, config['api_data']))
```

Warning: This plugin system needs to be revisited and will likely change by the next minor release.

6.3 Debugging

The default logging configuration comes with two file handlers, info and debug, which info enabled by default.

To enable debug, in *~/.xmm/xmm.logging.ini* change:

```
[logger_root]
level    = NOTSET
handlers = stream, info
```

```
[handlers]
keys = stream, info
```

To:

```
[logger_root]
level    = NOTSET
handlers = stream, debug

[handlers]
keys = stream, debug
```

- genindex
- modindex
- search

CHAPTER 7

API

7.1 Server

class xmm.server.**LocalServer**(*server_name='default'*, *source_name=None*, *make_dirs=False*)

This class sets up the *LocalServer* object

During instantiation, new objects are created based on configuration.

The hierarchy of these objects looks like:

```
•LocalServer
  –Library
    *Store
    *MapPackage
      ·Bsp
  –Collection
    *Repository
```

Parameters

- **server_name** (str) – Used to reference the server by name
- **source_name** (str) – If specified, the server will be associated with this one source
- **make_dirs** (bool) – If directories aren't found that are required by xmm on server init, optionally create them

Returns object LocalServer Commands are available off self.library.

Example

```
>>> from xmm.server import LocalServer
>>> server = LocalServer(server_name='myserver1')
>>> print(server)
```

to_json()

Returns A **JSON** encoded version of this object

class xmm.server.**ServerCollection**(*servers*)

A *ServerCollection* is a group of *LocalServer* objects. Currently unused.

list_servers()
Prints a list of servers

to_json()

Returns A **JSON** encoded version of this object

7.2 Library

class xmm.library.**Library**(*repositories*, *store*, *map_dir*)
A *Library* is a collection of *MapPackage* objects and commands for managing maps in the *Library*

Parameters

- **repositories** (Collection) – A *Collection* object with *Repository* objects
- **store** (Store) – A *Store* object for communicate with the data store for this *Library*
- **map_dir** (str) – The directory this *Library* is associated with

Returns object Library

add_map_package(*package*)
Adds a *MapPackage* object to self.maps

Parameters **package** (MapPackage) – A *MapPackage* object for the *Library*

discover_maps(*add=False*, *repository_name=None*, *detail=None*)
Searches the *Server*'s map_dir for map packages known by the *Repository*

Parameters

- **add** (bool) – Whether to add the discovered maps or not
- **repository_name** (str) – A name of a repository in the repository *Collection*
- **detail** (str) – How much detail to show, [short, None, long]
- **highlight** (bool) – Whether to highlight the results

```
>>> from xmm.server import LocalServer
>>> server = LocalServer()
>>> server.library.discover_maps(add=False)
```

export_hash_index(*filename=None*)

Parameters **filename** (str) – Name for the exported json file, default maps.json.shasums

Returns False if fails

```
>>> from xmm.server import LocalServer
>>> # Setup the store automatically with an instance of *LocalServer*
>>> server = LocalServer()
>>> server.library.export_hash_index(filename='test.maps.shasums')
```

export_map_packages(*filename=None*)
Exports all *MapPackage* objects from the *Library Store*

Parameters **filename** (str) – Name for the exported json file, default xmm-export.json

Returns False if fails

```
>>> from xmm.server import LocalServer
>>> # Setup the store automatically with an instance of *LocalServer*
>>> server = LocalServer()
>>> server.library.export_packages(filename='test.maps.json')
```

export_maplist (*filename=None*)

Parameters **filename** (str) – Name for the exported text file, default xmm-export.maps.txt

Returns False if fails

```
>>> from xmm.server import LocalServer
>>> # Setup the store automatically with an instance of *LocalServer*
>>> server = LocalServer()
>>> server.library.export_hash_index(filename='test.maps.txt')
```

get_repository_sources (*server_name*)

Gets the *Collection* from the *Library* of the specified *LocalServer* from `self.repositories` as cache, or from the `sources.json` targeted by `servers.json` if it is not already set.

Parameters **server_name** (str) – Server name

Returns Collection

install_map (*pk3_name*, *repository_name=None*, *overwrite=False*, *add_to_store=True*)

Install a *MapPackage* from a *Repository*

Parameters

- **pk3_name** (str) – A pk3 name such as `vinegar_v3.pk3`, to install from the repository.' Optionally prefixed with a URL to install map not in the repository. URL-only maps will not include rich metadata available to maps installed via the repo.
- **repository_name** (str) – A name of a repository in the repository *Collection*
- **overwrite** (bool) – Whether to overwrite the file on the file system
- **add_to_store** (bool) – Whether to add the map to the store (tracked)

```
>>> from xmm.server import LocalServer
>>> server = LocalServer(server_name='myserver1')
>>> server.library.install_map(pk3_name='vinegar_v3.pk3')
>>> print(server.library.maps)
```

list_installed (*detail=None*)

List maps currently tracked by the *Library*

Parameters **detail** (str) – How much detail to show, [short, None, long]

Returns int total count

```
>>> from xmm.server import LocalServer
>>> server = LocalServer()
>>> server.library.list_installed()
```

remove_map (*pk3_name*)

Removes a map from the *Library* and the package from the file system

Parameters **pk3_name** (str) – The name of a pk3, such as `vinegar_v3.pk3`

```
>>> from xmm.server import LocalServer
>>> server = LocalServer(server_name='myserver1')
>>> server.library.install_map_package(pk3_name='vinegar_v3.pk3')
>>> print(server.library.maps)
>>> server.library.remove_map(pk3_name='vinegar_v3.pk3')
>>> print(server.library.maps)
```

show_map(*pk3_name*, *detail=None*, *highlight=False*)
Convenience function to use the show_map_details helper

Parameters

- **pk3_name** (str) – The name of a pk3, such as vinegar_v3.pk3
- **detail** (str) – How much detail to show, [short, None, long]
- **highlight** (bool) – Whether to highlight the results

Returns MapPackage

```
>>> from xmm.server import LocalServer
>>> server = LocalServer()
>>> server.library.show_map('vinegar_v3.pk3', detail='long')
```

to_json()

Returns A **JSON** encoded version of this object

7.3 Repository

class xmm.repository.**Collection**
A *Collection* is a collection of *Repository* objects

Returns object *Collection*

```
>>> from xmm.repository import Collection
>>> from xmm.repository import Repository
>>> repositories = Collection()
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.xmm/
>>> repositories.add_repository(repository)
```

add_repository(*repository*)
Add a *Repository* to the *Collection*

Parameters **repository** (Repository) –

```
>>> from xmm.repository import Collection
>>> from xmm.repository import Repository
>>> repositories = Collection()
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> repositories.add_repository(repository)
>>> print(repositories.get_repository('default'))
```

export_all_hash_index(*filename=None*)

Parameters **filename** (str) – Name for the exported json file, default all-repos-maps.json.shasums

Returns False if fails

```
>>> from xmm.repository import Collection
>>> from xmm.repository import Repository
>>> repositories = Collection()
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> repositories.export_all_hash_index()
```

export_all_packages (*filename=None*)

Parameters **filename** (str) – Name for the exported json file, default maps.json

Returns False if fails

```
>>> from xmm.repository import Collection
>>> from xmm.repository import Repository
>>> repositories = Collection()
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> repositories.export_all_packages()
```

get_repository (*repository_name*)

Parameters **repository_name** (str) –

Returns A **Repository** object or false if name not found

```
>>> from xmm.repository import Collection
>>> from xmm.repository import Repository
>>> repositories = Collection()
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> repositories.add_repository(repository)
>>> print(repositories.get_repository('default'))
```

list_repositories ()

Prints a list of servers

search_all (*bsp_name=False*, *gametype=False*, *author=False*, *title=False*, *pk3_name=False*, *shasum=False*, *detail=None*, *highlight=False*)

Searches all *Repository* objects in the *Collection* for maps matching criteria

Parameters

- **bsp_name** (str) – Search by bsp name
- **gametype** (str) – Search by gametype
- **author** (str) – Search by author
- **title** (str) – Search by title
- **pk3_name** (str) – Search by pk3_name
- **shasum** (str) – Search by shasum
- **detail** (str) – How much detail in the results, [short, None, long]
- **highlight** (bool) – Whether to highlight the search string

```
>>> from xmm.repository import Collection
>>> from xmm.repository import Repository
>>> repositories = Collection()
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
```

```
>>>                          api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> repositories.add_repository(repository)
>>> print(repositories.search_all(bsp_name='vinegar_v3'))
```

to_json()

Returns A **JSON** encoded version of this object

update_all()

Update the data for all *Repository* objects in the *Collection*

```
>>> from xmm.repository import Collection
>>> from xmm.repository import Repository
>>> repositories = Collection()
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> repositories.update_all()
```

class xmm.repository.**Repository**(*name*, *download_url*, *api_data_url*, *api_data_file*)

A *Repository* contains a url which hosts content matching the **JSON** format described in the documentation

Parameters

- **name** (str) – A name for this *Repository*
- **download_url** (str) – The url where the pk3 files should be downloaded from
- **api_data_url** (str) – URL serving maps in the *JSON* format described in the documentation
- **api_data_file** (str) – Local cache fo the repo data.

Returns object Repository

```
>>> from xmm.repository import Repository
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.xmm/
```

export_hash_index(*filename=None*)

Parameters **filename** (str) – Name for the exported json file, default maps.json.shasums

Returns False if fails

```
>>> from xmm.repository import Repository
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> repository.export_hash_index('test.shasums')
```

export_packages(*filename=None*)

Parameters **filename** (str) – Name for the exported json file, default maps.json

Returns False if fails

```
>>> from xmm.repository import Repository
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> repository.export_packages('test.json')
```

get_hash_index()

Gets a list of all pk3s and their shasums

Returns False if fails

```
>>> from xmm.repository import Repository
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> print(repository.get_hash_index())
```

get_packages()

Gets the cached map list from *Repository* or reads from file if cache not available

Returns `json`

```
>>> from xmm.repository import Repository
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> print(repository.get_packages())
```

search_maps(*bsp_name=False*, *gametype=False*, *author=False*, *title=False*, *pk3_name=False*, *shasum=False*, *detail=None*, *highlight=False*)

Searches the repository for maps matching criteria

Parameters

- **bsp_name** (`str`) – Search by bsp name
- **gametype** (`str`) – Search by gametype
- **author** (`str`) – Search by author
- **title** (`str`) – Search by title
- **pk3_name** (`str`) – Search by pk3_name
- **shasum** (`str`) – Search by shasum
- **detail** (`str`) – How much detail in the results, [short, None, long]
- **highlight** (`bool`) – Whether to highlight the search string

```
>>> from xmm.repository import Repository
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> repository.search_maps(bsp_name='dance' gametype='ctf')
```

show_map(*pk3_name*, *detail=None*, *highlight=False*)

Convenience function to use the show_map_details helper

Parameters

- **pk3_name** (`str`) – The name of a pk3, such as `vinegar_v3.pk3`
- **detail** (`str`) – How much detail to show, [short, None, long]
- **highlight** (`bool`) – Whether to highlight the results

Returns `MapPackage`

```
>>> from xmm.repository import Repository
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> repository.show_map(pk3_name='vinegar_v3.pk3')
```

to_json()

Returns A **JSON** encoded version of this object

update_repo_data()
Updates sources cache with latest maps from *Repository*

```
>>> from xmm.repository import Repository
>>> repository = Repository(name='default', download_url='http://dl.repo.url/',
>>>                         api_data_url='http://api.repo.url/maps.json', api_data_file='~/.
>>> repository.update_repo_data()
```

7.4 Map Packages

class xmm.map.**Bsp**(*pk3_file=''*, *bsp_name=''*, *bsp_file=''*, *map_file=''*, *mapshot=''*, *radar=''*, *title=''*, *description=''*, *mapinfo=''*, *author=''*, *gametypes=None*, *entities=None*, *waypoints=''*, *license=False*)
A *Bsp* is a child of a *MapPackage* that holds metadata about this map

Parameters

- **pk3_file** (str) – The pk3_file name of the package this bsp is in
- **bsp_name** (str) – The bsp_name of the bsp_file
- **bsp_file** (str) – The bsp_file
- **map_file** (str) – The map_file for the bsp_file if it exists
- **mapshot** (str) – The mapshot for the bsp_file if it exists
- **title** (str) – The title for the bsp_file if it exists
- **description** (str) – The description for the bsp_file if it exists
- **mapinfo** (str) – The mapinfo for the bsp_file if it exists
- **author** (str) – The author for the bsp_file if it exists
- **gametypes** (list) – The gametypes for the bsp_file if it exists
- **entities** (str) – The entities for the bsp_file if they exists
- **waypoints** (str) – The waypoints for the bsp_file if it exists
- **license** (str) – The license for the bsp_file if it exists

Returns object Bsp

to_json()

Returns A **JSON** encoded version of this object

class xmm.map.**MapPackage**(*map_package_json*)
MapPackage contains top-level metadata about a pk3 file and list of *Bsp* objects inside this package

Parameters **map_package_json** (string|dict) – A dict or *JSON* string that matches "specification" in the Developers section of the documentation.

See basic example below:

```
{
  "data": [
    {
      "date": 1453749340,
      "filesize": 7856907,
      "bsp": {
```

```
                "vapor_alpha_2": {
                  "radar": "gfx/vapor_alpha_2_mini.tga",
                  "waypoints": "",
                  "title": "Vapor",
                  "description": "Such CTF. Many Vehicles. Wow.",
                  "map": "maps/vapor_alpha_2.map",
                  "entities": {
                    "info_player_deathmatch": 4,
                    "info_player_team1": 11,
                    "info_player_team2": 11,
                    "item_armor_big": 10,
                    "item_armor_large": 4,
                    "item_armor_medium": 16,
                    "item_armor_small": 124,
                    "item_bullets": 10,
                    "item_cells": 14,
                    "item_flag_team1": 1,
                    "item_flag_team2": 1,
                    "item_health_large": 6,
                    "item_health_medium": 30,
                    "item_health_mega": 2,
                    "item_health_small": 100,
                    "item_invincible": 1,
                    "item_rockets": 20,
                    "item_strength": 1,
                    "weapon_crylink": 4,
                    "weapon_devastator": 6,
                    "weapon_electro": 2,
                    "weapon_grenadelauncher": 6,
                    "weapon_hagar": 4,
                    "weapon_machinegun": 6,
                    "weapon_vortex": 4
                  },
                  "mapinfo": "maps/vapor_alpha_2.mapinfo",
                  "author": "-z-",
                  "gametypes": [
                    "ctf",
                    "DM"
                  ],
                  "license": true,
                  "mapshot": "maps/vapor_alpha_2.jpg"
                }
              },
              "shasum": "3df0143516f72269f465070373f165c8787964d5",
              "pk3": "map-vapor_alpha_2.pk3"
            }
          ]
        }
```


Returns object `MapPackage`

```
>>> from xmm.map import MapPackage
>>> with open('my_map.json') as f:
>>>     data = f.read()
>>>     my_map = MapPackage(map_package_json=data)
```

show_map_details(*detail=None*, *search_string=''*, *highlight=False*)

Helper function for pretty printing details about a *MapPackage*

Convenience function to use the show_map_details helper

Parameters

- **detail** (str) – How much detail to show, [short, None, long]
- **search_string** (str) – A string to highlight with highlight=True
- **highlight** (bool) – Whether to highlight the results

Returns MapPackage

to_json()

Returns A **JSON** encoded version of this object

7.5 Store

class xmm.store.**Store**(*package_store_file*)

Store is for interacting with the datastore for a *Library*

Parameters **package_store_file** (str) – The file where the data is stored

```
>>> import os
>>> from xmm.store import Store
>>> package_store_file = os.path.expanduser('~/.xmm/library.json')
>>> store = Store(package_store_file=package_store_file)
```

Returns object Store

add_package(*package*)

Adds a *MapPackage* to the *Library Store*

Parameters **package** (MapPackage) – MapPackage to add

```
>>> import os
>>> from xmm.map import MapPackage
>>> from xmm.store import Store
>>> package_store_file = os.path.expanduser('~/.xmm/library.json')
>>> with open('my_map.json') as f:
>>>     data = f.read()
>>>     my_map = MapPackage(map_package_json=data)
>>> store = Store(package_store_file=package_store_file)
>>> store.add_package(my_map)
```

Returns False if fails

export_packages(*filename=None*)

Exports all *MapPackage* objects from the *Library Store*

Parameters **filename** (str) – Name for the exported json file, default xmm-export.json

Returns False if fails

```
>>> from xmm.server import LocalServer
>>> # Setup the store automatically with an instance of *LocalServer*
>>> server = LocalServer()
>>> server.library.store.export_packages(filename='test.json')
```

get_package_db()
Searches the repository for maps matching criteria

```
>>> import os
>>> from xmm.store import Store
>>> package_store_file = os.path.expanduser('~/.xmm/library.json')
>>> store = Store(package_store_file=package_store_file)
>>> store.get_package_db()
```

Returns dict

remove_package(*package*)
Removes a *MapPackage* to the *Library Store*

Parameters **package** (MapPackage) – MapPackage to remove

```
>>> import os
>>> from xmm.map import MapPackage
>>> from xmm.store import Store
>>> package_store_file = os.path.expanduser('~/.xmm/library.json')
>>> with open('my_map.json') as f:
>>>     data = f.read()
>>>     my_map = MapPackage(map_package_json=data)
>>> store = Store(package_store_file=package_store_file)
>>> store.remove_package(my_map)
```

Returns False if fails

to_json()

Returns A **JSON** encoded version of this object

7.6 Utility

class xmm.util.**ObjectEncoder**(*skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)
JSONEncoder subclass that leverages an object's *__json__()* method, if available, to obtain its default JSON representation.

xmm.util.**check_if_not_create**(*file*, *template*)
Checks for a file, if it doesn't exist, it will be created from a template.

Parameters

- **file** (str) – filename with path to file
- **template** (str) – filename with path to template file

xmm.util.**convert_size**(*number*)
Convert and integer to a human-readable B/KB/MB/GB/TB string.

Parameters **number** (int) – integer to be converted to readable string

Returns *str*

xmm.util.**cprint**(*string*, *style='INFO'*)
Terminal formatting convenience function.

Parameters

- **string** (str) – A string to print.
- **style** (str) – A style to print.

  Options:

  - HEADER
  - INFO
  - SUCCESS
  - WARNING
  - FAIL
  - ENDC (end color)
  - BOLD
  - UNDERLINE

```
>>> cprint("Success", style='SUCCESS')
```

xmm.util.**create_if_not_exists**(*file*, *contents*)
Checks for a file, if it doesn't exist, it will be created from a template.

Parameters

- **file** (str) – filename with path to file
- **contents** (str) – string contents of the file being created

xmm.util.**download_file**(*filename_with_path*, *url*, *use_curl=False*, *overwrite=False*)
downloads a file from any URL

Parameters

- **filename_with_path** (str) – filename with path to download file to
- **url** (str) – URL to download map from
- **use_curl** (bool) – Whether or not to use curl to download the file, default False
- **overwrite** – Whether or not to overwrite the existing file, default False

xmm.util.**file_is_empty**(*filename*)
Checks to see if a file is empty

Parameters **filename** (str) – string filename

Returns bool

xmm.util.**hash_file**(*filename*)
Returns the SHA-1 hash of the file passed into it

Parameters **filename** (str) – string filename

Returns str

xmm.util.**parse_config**(*config_file*)
downloads a file from any URL

Parameters **config_file** (str) – filename with path to config file

Returns dict

xmm.util.**query_yes_no**(*question*, *default='yes'*)
Ask a yes/no question via raw_input() and return their answer.

Parameters

- **question** (str) – a string that is presented to the user.
- **default** (str) – is the presumed answer if the user just hits <Enter>. It must be "yes" (the default), "no" or None (meaning an answer is required of the user).

The "answer" return value is True for "yes" or False for "no".

xmm.util.**replace_last**(*string*, *old*, *new*)
Replace the last occurrence of a pattern in a string

Parameters

- **string** (str) – string
- **old** (str) – string to find
- **new** (str) – string to replace

Returns str

xmm.util.**reporthook**(*count*, *block_size*, *total_size*)
Pretty progress for urllib downloads.

```
>>> import urllib.request
>>> urllib.request.urlretrieve(url, filename, reporthook)
```

https://github.com/yahoo/caffe/blob/master/scripts/download_model_binary.py

class xmm.util.**zcolors**
Terminal formatting.

Options:

•HEADER

•INFO

•SUCCESS

•WARNING

•FAIL

•ENDC (end color)

•BOLD

•UNDERLINE

```
>>> "{}eggs{}: {}spam{}".format(zcolors.INFO, zcolors.ENDC, zcolors.UNDERLINE, zcolors.ENDC)
```

7.7 Exceptions

exception xmm.exceptions.**HashMismatchError**
Raise when a file hash mismatches

exception xmm.exceptions.**PackageLookupError**
Raise when Package does not exist in Repository

exception xmm.exceptions.**PackageMetadataWarning**
Raise when Package installs from a URL and has no metadata associated with it.

exception xmm.exceptions.**PackageNotTrackedWarning**
Raise when Package is not tracked in the local db

exception xmm.exceptions.**RepositoryLookupError**
Raise when Repository lookup fails

exception xmm.exceptions.**RepositoryUpdateError**(*reason*, *filename=None*)
Raise when Repository update fails

exception xmm.exceptions.**ServerLookupError**
Raise when Server lookup fails

- genindex
- modindex
- search

CHAPTER 8

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

CHAPTER 9

License

- genindex
- modindex
- search

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

X

S

T

U

X

Z