

---

# libloot-python Documentation

*Release latest*

**WrinklyNinja**

**Jun 05, 2023**



---

## Contents

---

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Usage</b>                           | <b>3</b> |
| 1.1      | Installing the wrapper . . . . .       | 3        |
| 1.2      | Using the wrapper . . . . .            | 3        |
| <b>2</b> | <b>API Reference</b>                   | <b>5</b> |
| 2.1      | Enumerations . . . . .                 | 5        |
| 2.2      | Public-Field Data Structures . . . . . | 6        |
| 2.3      | Functions . . . . .                    | 7        |
| 2.4      | Classes . . . . .                      | 7        |
|          | <b>Index</b>                           | <b>9</b> |



Contents:



## 1.1 Installing the wrapper

Build archives contain two binaries:

- `loot.*.pyd` is the Python wrapper
- `loot.dll` is the C++ library DLL that the Python wrapper was built against.

The C++ DLL requires the [Visual C++ 2019 Redistributable \(x86\)](#) to be installed.

To use the wrapper, copy both files to wherever you want to import them from (they must be in the same folder), and you're done!

## 1.2 Using the wrapper

### 1.2.1 Checking Compatibility

To check if the module loaded is compatible with the version of the API that you developed against:

```
>>> import loot
>>> loot.is_compatible(0,14,0)
True
>>> loot.is_compatible(0,9,0)
False
```

### 1.2.2 Getting a Plugin's Bash Tag Suggestions

To get a plugin's Bash Tag suggestions from a `masterlist.yaml` metadata file:

```
>>> import loot
>>> db = loot.create_game_handle(loot.GameType.tes4,
↳ 'C:\\path\\to\\oblivion\\directory')
>>> db.load_lists('masterlist.yaml')
>>> tags = db.get_plugin_tags(u'Unofficial Oblivion Patch.esp')
>>> tags.added
set([u'Scripts', u'Relations', u'C.Owner', u'Actors.AIPackages', u'Actors.Stats', u
↳ 'Actors.ACBS', u'C.Music', u'Factions', u'Invent', u'Relev', u'Names', u'C.Light', u
↳ 'Delev', u'C.Name', u'C.Climate', u'NPC.Class', u'Stats', u'Actors.DeathItem', u
↳ 'Creatures.Blood', u'Actors.CombatStyle', u'Actors.AIData'])
>>> tags.removed
set([u'C.Water'])
>>> tags.userlist_modified
False
```



As this API is just a wrapper for libloot's C++ API, its documentation is linked to for all non-Python-specific information.

### 2.1 Enumerations

The wrapped enumeration types below are classes in Python, but the distinction makes no difference in practice, so they're grouped here for semantics. All values are unsigned integer constants.

```
class loot.GameType
    Wraps loot::GameType to expose libloot's game codes.

    fo3
    fo4
    fonv
    tes4
    tes5
    tes5se

class loot.LogLevel
    Wraps loot::LogLevel to expose libloot's log level codes.

    trace
    debug
    info
    warning
    error
    fatal
```

**class** `loot.MessageType`

Wraps `loot::MessageType` to expose libloot's message type codes.

**error**

**say**

**warn**

**class** `loot.PluginCleanliness`

Codes used to indicate the cleanliness of a plugin according to the information contained within the loaded masterlist/userlist.

**clean**

Indicates that the plugin is clean.

**dirty**

Indicates that the plugin is dirty.

**do\_not\_clean**

Indicates that the plugin contains dirty edits, but that they are part of the plugin's intended functionality and should not be removed.

**unknown**

Indicates that no data is available on whether the plugin is dirty or not.

## 2.2 Public-Field Data Structures

Classes with public fields and no member functions.

**class** `loot.MasterlistInfo`

Wraps `loot::MasterlistInfo`.

**revision\_id**

A Unicode string containing a Git commit's SHA-1 checksum.

**revision\_date**

A Unicode string containing the date of the commit given by *revision\_id*, in ISO 8601 format (YYYY-MM-DD).

**is\_modified**

A boolean that is true if the masterlist has been modified from its state at the commit given by *revision\_id*.

**class** `loot.SimpleMessage`

Wraps `loot::SimpleMessage`.

**type**

A *loot.MessageType* giving the message type.

**language**

A Unicode string giving the message text language.

**text**

A Unicode string containing the message text.

**condition**

A Unicode string containing the message condition.

**class** `loot.PluginTags`

Wraps `loot::PluginTags`.

**added**

A set of Unicode strings giving Bash Tags suggested for addition.

**removed**

A set of Unicode strings giving Bash Tags suggested for removal.

**userlist\_modified**

A boolean that is true if the suggestions contain metadata obtained from a loaded userlist.

## 2.3 Functions

`loot.set_logging_callback(callback) → NoneType`

Set the callback function that is called when logging. Wraps `loot::SetLoggingCallback()`.

`loot.is_compatible(int, int, int) → bool`

Checks for API compatibility. Wraps `loot::IsCompatible()`.

`loot.create_game_handle(game : loot.GameType, game_path : unicode[, game_local_path : unicode = u'']) → loot.GameInterface`

Initialise a new game handle. Wraps `loot::CreateGameHandle()`.

## 2.4 Classes

**class** `loot.GameInterface`

Wraps `loot::GameInterface`.

`loot.get_database() → loot.DatabaseInterface`

Get a database handle. Wraps `loot::GetDatabase()`.

`loot.load_current_load_order_state() → NoneType`

Load the current load order state, discarding any previously held state. Wraps `loot::LoadCurrentLoadOrderState()`.

**class** `loot.DatabaseInterface`

Wraps `loot::DatabaseInterface`.

`get_masterlist_revision(loot.DatabaseInterface, unicode, bool) → loot.MasterlistInfo`

Gets the give masterlist's source control revision. Wraps `GetMasterlistRevision()`.

`get_plugin_metadata(loot.DatabaseInterface, plugin : unicode[, includeUserMetadata : bool = True[, evaluateConditions : bool = False]]) → loot.PluginMetadata`

Get all a plugin's loaded metadata. Wraps `GetPluginMetadata()`.

`get_plugin_cleanliness(loot.DatabaseInterface, plugin : unicode[, evaluateConditions : bool = False]) → loot.PluginCleanliness`

Determines the database's knowledge of a plugin's cleanliness. Outputs whether the plugin should be cleaned or not, or if no data is available.

`get_plugin_tags(loot.DatabaseInterface, plugin : unicode[, evaluateConditions : bool = False]) → loot.PluginTags`

Outputs the Bash Tags suggested for addition and removal by the database for the given plugin.

`load_lists(loot.DatabaseInterface, masterlist_path : unicode[, userlist_path : unicode = u'']) → NoneType`

Loads the masterlist and userlist from the paths specified. Wraps `LoadLists()`.

`update_masterlist(loot.DatabaseInterface, unicode, unicode, unicode) → bool`

Updates the given masterlist using the given Git repository details. Wraps `UpdateMasterlist()`.

**write\_minimal\_list** (*loot.DatabaseInterface, unicode, bool*) → NoneType

Writes a minimal metadata file containing only Bash Tag suggestions and/or cleanliness info from the loaded metadata. Wraps `WriteMinimalList()`.

**class** `loot.Version`

Wraps `loot::LootVersion`.

**major**

An unsigned integer giving the major version number.

**minor**

An unsigned integer giving the minor version number.

**patch**

An unsigned integer giving the patch version number.

**revision**

A Unicode string containing the SHA-1 of the Git revision that the wrapped C++ API was built from.

**static string** () → unicode

Returns the API version as a string of the form `major.minor.patch`

**class** `loot WrapperVersion`

Provides information about the version of libloot-python that is being run.

**major**

An unsigned integer giving the major version number.

**minor**

An unsigned integer giving the minor version number.

**patch**

An unsigned integer giving the patch version number.

**revision**

A Unicode string containing the SHA-1 of the Git revision that the wrapped C++ API was built from.

**static string** () → unicode

Returns the API version as a string of the form `major.minor.patch`

**class** `loot.PluginMetadata`

Wraps `loot::PluginMetadata`.

**get\_simple\_messages** (*loot.PluginMetadata, unicode*) → list<loot.SimpleMessage>

Get the plugin's messages as SimpleMessage objects for the given language. Wraps `GetPluginMessages()`.

## A

added (*loot.PluginTags attribute*), 6

## C

clean (*loot.PluginCleanliness attribute*), 6

condition (*loot.SimpleMessage attribute*), 6

## D

debug (*loot.LogLevel attribute*), 5

dirty (*loot.PluginCleanliness attribute*), 6

do\_not\_clean (*loot.PluginCleanliness attribute*), 6

## E

error (*loot.LogLevel attribute*), 5

error (*loot.MessageType attribute*), 6

## F

fatal (*loot.LogLevel attribute*), 5

fo3 (*loot.GameType attribute*), 5

fo4 (*loot.GameType attribute*), 5

fonv (*loot.GameType attribute*), 5

## G

get\_masterlist\_revision()  
(*loot.DatabaseInterface method*), 7

get\_plugin\_cleanliness()  
(*loot.DatabaseInterface method*), 7

get\_plugin\_metadata() (*loot.DatabaseInterface method*), 7

get\_plugin\_tags() (*loot.DatabaseInterface method*), 7

get\_simple\_messages() (*loot.PluginMetadata method*), 8

## I

info (*loot.LogLevel attribute*), 5

is\_modified (*loot.MasterlistInfo attribute*), 6

## L

language (*loot.SimpleMessage attribute*), 6

load\_lists() (*loot.DatabaseInterface method*), 7

loot.create\_game\_handle() (*built-in function*), 7

loot.DatabaseInterface (*built-in class*), 7

loot.GameInterface (*built-in class*), 7

loot.GameInterface.loot.get\_database()  
(*built-in function*), 7

loot.GameInterface.loot.load\_current\_load\_order\_status()  
(*built-in function*), 7

loot.GameType (*built-in class*), 5

loot.is\_compatible() (*built-in function*), 7

loot.LogLevel (*built-in class*), 5

loot.MasterlistInfo (*built-in class*), 6

loot.MessageType (*built-in class*), 5

loot.PluginCleanliness (*built-in class*), 6

loot.PluginMetadata (*built-in class*), 8

loot.PluginTags (*built-in class*), 6

loot.set\_logging\_callback() (*built-in function*), 7

loot.SimpleMessage (*built-in class*), 6

loot.Version (*built-in class*), 8

loot.WrapperVersion (*built-in class*), 8

## M

major (*loot.Version attribute*), 8

major (*loot.WrapperVersion attribute*), 8

minor (*loot.Version attribute*), 8

minor (*loot.WrapperVersion attribute*), 8

## P

patch (*loot.Version attribute*), 8

patch (*loot.WrapperVersion attribute*), 8

## R

removed (*loot.PluginTags attribute*), 7

revision (*loot.Version attribute*), 8

revision (*loot.WrapperVersion attribute*), 8

revision\_date (*loot.MasterlistInfo* attribute), 6  
revision\_id (*loot.MasterlistInfo* attribute), 6

## S

say (*loot.MessageType* attribute), 6  
string() (*loot.Version* static method), 8  
string() (*loot.WrapperVersion* static method), 8

## T

tes4 (*loot.GameType* attribute), 5  
tes5 (*loot.GameType* attribute), 5  
tes5se (*loot.GameType* attribute), 5  
text (*loot.SimpleMessage* attribute), 6  
trace (*loot.LogLevel* attribute), 5  
type (*loot.SimpleMessage* attribute), 6

## U

unknown (*loot.PluginCleanliness* attribute), 6  
update\_masterlist() (*loot.DatabaseInterface*  
method), 7  
userlist\_modified (*loot.PluginTags* attribute), 7

## W

warn (*loot.MessageType* attribute), 6  
warning (*loot.LogLevel* attribute), 5  
write\_minimal\_list() (*loot.DatabaseInterface*  
method), 7