Skip to content

Extensions Command Line Arguments

Command for managing Blender extensions.

options:

`-h, --help`
: show this help message and exit

subcommands:

Package Management

list:
: List all packages.

sync:
: Synchronize with remote repositories.

update:
: Upgrade any outdated packages.

install:
: Install packages.

install-file:
: Install package from file.

remove:
: Remove packages.

Repository Management

repo-list:
: List repositories.

repo-add:
: Add repository.

repo-remove:
: Remove repository.

Extension Creation

build:
: Build a package.

validate:
: Validate a package.

server-generate:
: Create a listing from all packages.

Package Management

Subcommand: `list`

usage:

```
blender --command extension list [-h] [-s]
```

List packages from all enabled repositories.

options:

-h, --help

show this help message and exit

-s, --sync

Sync the remote directory before performing the action.

Subcommand: `sync`

usage:

```
blender --command extension sync [-h]
```

Download package information for remote repositories.

options:

-h, --help

show this help message and exit

Subcommand: `update`

usage:

```
blender --command extension update [-h] [-s]
```

Download and update any outdated packages.

options:

-h, --help

show this help message and exit

-s, --sync

Sync the remote directory before performing the action.

Subcommand: `install`

usage:

```
blender --command extension install [-h] [-s] [-e] [--no-prefs]
                                    PACKAGES
```

positional arguments:

PACKAGES:

The packages to operate on (separated by `,` without spaces).

options:

-h, --help

show this help message and exit

-s, --sync

Sync the remote directory before performing the action.

-e, --enable

Enable the extension after installation.

--no-prefs

Treat the user-preferences as read-only, preventing updates for operations that would otherwise modify them. This means removing extensions or repositories for example, wont update the user-preferences.

Subcommand: `install-file`

usage:

```
blender --command extension install-file [-h] -r REPO [-e] [--no-prefs]
                                         FILE
```

Install a package file into a user repository.

positional arguments:

FILE:
The packages file.

options:

`-h, --help`

show this help message and exit

`-r *REPO*, --repo *REPO*`

The repository identifier.

`-e, --enable`

Enable the extension after installation.

`--no-prefs`

Treat the user-preferences as read-only, preventing updates for operations that would otherwise modify them. This means removing extensions or repositories for example, wont update the user-preferences.

Subcommand: `remove`

usage:

```
blender --command extension remove [-h] [--no-prefs] PACKAGES
```

Disable & remove package(s).

positional arguments:

PACKAGES:
The packages to operate on (separated by `,` without spaces).

options:

`-h, --help`

show this help message and exit

`--no-prefs`

Treat the user-preferences as read-only, preventing updates for operations that would otherwise modify them. This means removing extensions or repositories for example, wont update the user-preferences.

Repository Management

Subcommand: `repo-list`

usage:

```
blender --command extension repo-list [-h]
```

List all repositories stored in Blender's preferences.

options:

-h, --help
: show this help message and exit

Subcommand: `repo-add`

usage:

```
blender --command extension repo-add [-h] [--name NAME]
                                     [--directory DIRECTORY]
                                     [--url URL]
                                     [--access-token ACCESS_TOKEN]
                                     [--source SOURCE]
                                     [--cache BOOLEAN] [--clear-all]
                                     [--no-prefs]
                                     ID
```

Add a new local or remote repository.

positional arguments:

ID:
: The repository identifier.

options:

-h, --help
: show this help message and exit

--name *NAME*
: The name to display in the interface (optional).

--directory *DIRECTORY*
: The directory where the repository stores local files (optional). When omitted a directory in the users directory is automatically selected.

--url *URL*
: The URL, for remote repositories (optional). When omitted the repository is considered "local" as it is not connected to an external repository, where packages may be installed by file or managed manually.

--access-token *ACCESS_TOKEN*
: The access token to use for remote repositories which require a token.

--source *SOURCE*
: The type of source in ('USER', 'SYSTEM'). System repositories are managed outside of Blender and are considered read-only.

--cache *BOOLEAN*
: Use package cache (default=1).

--clear-all
: Clear all repositories before adding, simplifies test setup.

--no-prefs
: Treat the user-preferences as read-only, preventing updates for operations that would otherwise modify them. This means removing extensions or repositories for example, wont update the user-preferences.

Subcommand: `repo-remove`

usage:

```
blender --command extension repo-remove [-h] [--no-prefs] ID
```

Remove a repository.

positional arguments:

ID:
: The repository identifier.

options:

`-h, --help`
: show this help message and exit

`--no-prefs`
: Treat the user-preferences as read-only, preventing updates for operations that would otherwise modify them. This means removing extensions or repositories for example, wont update the user-preferences.

Extension Creation

Subcommand: `build`

usage:

```
blender --command extension build [-h] [--source-dir SOURCE_DIR]
                                  [--output-dir OUTPUT_DIR]
                                  [--output-filepath OUTPUT_FILEPATH]
                                  [--valid-tags VALID_TAGS_JSON]
                                  [--split-platforms] [--verbose]
```

Build a package in the current directory.

options:

`-h, --help`
: show this help message and exit

`--source-dir` *`SOURCE_DIR`*
: The package source directory containing a `blender_manifest.toml` manifest.

  Default's to the current directory.

`--output-dir` *`OUTPUT_DIR`*
: The package output directory.

  Default's to the current directory.

`--output-filepath` *`OUTPUT_FILEPATH`*
: The package output filepath (should include a `.zip` extension).

  Defaults to `{id}-{version}.zip` using values from the manifest.

`--valid-tags` *`VALID_TAGS_JSON`*
: Reference a file path containing valid tags lists.

  If you wish to reference custom tags a `.json` file can be used. The contents must be a dictionary of lists where the `key` matches the

If you wish to reference custom tags a `.json` file can be used. The contents must be a dictionary of lists where the `key` matches the extension type.

For example:

`{"add-ons": ["Example", "Another"], "theme": ["Other", "Tags"]}`

To disable validating tags, pass in an empty path `--valid-tags=""`.

`--split-platforms`

Build a separate package for each platform. Adding the platform as a file name suffix (before the extension).

This can be useful to reduce the upload size of packages that bundle large platform-specific modules (`*.whl` files).

`--verbose`

Include verbose output.

Subcommand: `validate`

usage:

```
blender --command extension validate [-h]
                                     [--valid-tags VALID_TAGS_JSON]
                                     [SOURCE_PATH]
```

Validate the package meta-data in the current directory.

positional arguments:

SOURCE_PATH:

The package source path (either directory containing package files or the package archive). This path must containing a `blender_manifest.toml` manifest.

Defaults to the current directory.

options:

`-h, --help`

show this help message and exit

`--valid-tags` *`VALID_TAGS_JSON`*

Reference a file path containing valid tags lists.

If you wish to reference custom tags a `.json` file can be used. The contents must be a dictionary of lists where the `key` matches the extension type.

For example:

`{"add-ons": ["Example", "Another"], "theme": ["Other", "Tags"]}`

To disable validating tags, pass in an empty path `--valid-tags=""`.

Subcommand: `server-generate`

usage:

```
blender --command extension server-generate [-h] --repo-dir REPO_DIR
                                            [--repo-config REPO_CONFIG]
                                            [--html]
                                            [--html-template HTML_TEMPLATE_FILE]
```

Generate a listing of all packages stored in a directory. This can be used to host packages which only requires static-file hosting.

options:

options:

-h, --help

show this help message and exit

--repo-dir *REPO_DIR*

The remote repository directory.

--repo-config *REPO_CONFIG*

An optional server configuration to include information which can't be detected. Defaults to `blender_repo.toml` (in the repository directory).

This can be used to defined blocked extensions, for example

```
schema_version = "1.0.0"

[[blocklist]]
id = "my_example_package"
reason = "Explanation for why this extension was blocked"
[[blocklist]]
id = "other_extenison"
reason = "Another reason for why this is blocked"
```

--html

Create a HTML file (`index.html`) as well as the repository JSON to support browsing extensions online with static-hosting.

--html-template *HTML_TEMPLATE_FILE*

An optional HTML file path to override the default HTML template with your own.

The following keys will be replaced with generated contents:

- `${body}` is replaced the extensions contents.
- `${date}` is replaced the creation date.