deluge Documentation

Release 2.1.1.dev115

Deluge Team

August 26, 2024

CONTENTS

CONTENTS

1.1 Getting started with Deluge

This is a starting point if you are new to Deluge where we will walk you through getting up and running with our BitTorrent client.

1.1.1 Installing Deluge

Instructions for installing Deluge.

Linux

Ubuntu

One-click **Install**

```
sudo apt install deluge
```

Package Details

Fedora

```
sudo dnf install deluge
```

Package Details

Arch

```
pacman -S deluge-gtk
```

Arch Wiki

OpenSUSE

[1 Click Install](#)

[Package Details](#)

Gentoo

[Package Details](#)

Flatpak

One-click [Install](#)

[Package Details](#)

Windows

Download [installer](#)

Availble for Windows 7, 8 & 10 for both 32-bit and 64-bit OSes.

macOS

Unfortunately no official installer package currently available.

See *[Alternative Installs](#)*

FreeBSD

```
pkg add deluge
```

[Package details](#)

PyPi

Install with pip:

```
pip install deluge
```

Install with all *[optional dependencies](#)*:

```
pip install deluge[all]
```

Will require system installed packages such as libtorent and GTK3. See *[DEPENDS](#)*

e.g. on Ubuntu/Debian install these packages:

```
sudo apt install python3-pip python3-libtorrent python3-gi python3-gi-cairo gir1.2-gtk-3.
→0 gir1.2-ayatanaappindicator3-0.1
```

Alternative Installs

Ubuntu PPA

The stable PPA contains the latest releases.

```
sudo add-apt-repository -u ppa:deluge-team/stable
sudo apt install deluge
```

The development PPA contains daily builds from the `develop` branch.

```
sudo add-apt-repository -u ppa:deluge-team/develop
sudo apt install deluge
```

macOS Community

Unofficial `.app` packages

Check sticky topics in MacOS Forum

Macports

```
sudo port install deluge
```

Package Details

Homebrew

1. Install Homebrew
2. Open a terminal to install required packages:

   ```
   brew install pygobject3 gtk+3 adwaita-icon-theme
   brew install libtorrent-rasterbar
   ```

3. To fix translations:

   ```
   brew link gettext --force
   ```

4. Install Deluge:

   ```
   pip install deluge
   ```

1.2 How-to guides

A collection of guides covering common issues that might be encountered using Deluge.

1.2.1 GTK UI

How to set Deluge as default torrent application

Check registered MIME types

```
gio mime application/x-bittorrent
gio mime x-scheme-handler/magnet
```

Set Deluge as default for MIME types

```
gio mime x-scheme-handler/magnet deluge.desktop
gio mime application/x-bittorrent deluge.desktop
```

Troubleshooting

```
update-mime-database ~/.local/share/mime
update-desktop-database ~/.local/share/applications
```

XDG Check

```
xdg-mime query default x-scheme-handler/magnet
```

References

https://help.gnome.org/admin/system-admin-guide/stable/mime-types-custom-user.html.en

1.2.2 Deluge as a service

Services are used to start applications on system boot and leave them running in the background. They will also stop the application nicely on system shutdown and automatically restart them if they crash.

The Deluge daemon deluged and Web UI deluge-web can both be run as services.

How to create systemd services for Linux

This guide walks you through setting up Deluge systemd services on Linux.

Ensure Deluge daemon `deluged` and Web UI `deluge-web` are installed. Use `which` to check installation paths and if necessary modify the service file `ExecStart` lines to point to alternative paths.

Create a service specific user

For security it is best to run a service with a specific user and group. You can create one using the following command:

```
sudo adduser --system  --gecos "Deluge Service" --disabled-password --group --home /var/
→lib/deluge deluge
```

This creates a new system user and group named `deluge` with no login access and home directory `/var/lib/deluge` which will be the default location for the config files.

In addition you can add to the `deluge` group any users you wish to be able to easily manage or access files downloaded by Deluge, for example:

```
sudo adduser <username> deluge
```

Daemon (deluged) service

Create the file `/etc/systemd/system/deluged.service` containing the following:

```
[Unit]
Description=Deluge Bittorrent Client Daemon
Documentation=man:deluged
After=network-online.target

[Service]
Type=simple
UMask=007

ExecStart=/usr/bin/deluged -d

Restart=on-failure

# Time to wait before forcefully stopped.
TimeoutStopSec=300

[Install]
WantedBy=multi-user.target
```

User configuration

To run the service using the previously created user e.g. `deluge`, first create the service configuration directory:

```
sudo mkdir /etc/systemd/system/deluged.service.d/
```

Then create a user file `/etc/systemd/system/deluged.service.d/user.conf` with the following contents:

```
# Override service user
[Service]
User=deluge
Group=deluge
```

Start deluged service

Now enable it to start up on boot, start the service and verify it is running:

```
sudo systemctl enable /etc/systemd/system/deluged.service
sudo systemctl start deluged
sudo systemctl status deluged
```

Umask for deluged downloaded files

The `umask` in the service file can be modified to determine access to files downloaded by deluged (also applies to logging files). Some useful access values are detailed as follows:

- `000` - full access for all users and groups.
- `007` - only user and group that `deluged` is running as (e.g. `deluge`) with no access from any other accounts.
- `002` - user and group `deluged` is running as with read-only for all other accounts.
- `022` - user `deluged` is running as with read-only for all other accounts.

The service for `deluged` must be stopped and started instead of just restarted after changes.

Web UI (deluge-web) service

Create the file `/etc/systemd/system/deluge-web.service` containing the following:

```
[Unit]
Description=Deluge Bittorrent Client Web Interface
Documentation=man:deluge-web
After=deluged.service
Wants=deluged.service

[Service]
Type=simple
UMask=027

ExecStart=/usr/bin/deluge-web -d
```

```
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

User configuration

To run the service using the previously created user e.g. `deluge`, first create the service configuration directory:

```
sudo mkdir /etc/systemd/system/deluge-web.service.d/
```

Then create a user file `/etc/systemd/system/deluge-web.service.d/user.conf` with the following contents:

```
# Override service user
[Service]
User=deluge
Group=deluge
```

Start deluge-web service

Now enable it to start up on boot, start the service and verify it is running:

```
sudo systemctl enable /etc/systemd/system/deluge-web.service
sudo systemctl start deluge-web
sudo systemctl status deluge-web
```

Service logging

Create a log directory for Deluge and give the service user (e.g. `deluge`), full access:

```
sudo mkdir -p /var/log/deluge
sudo chown -R deluge:deluge /var/log/deluge
sudo chmod -R 750 /var/log/deluge
```

The deluge log directory is now configured so that user `deluge` has full access, group `deluge` read only and everyone else denied access. The `umask` specified in the services sets the permission of new log files.

Enable logging in the service files by editing the `ExecStart` line, appending `-l` and `-L` options:

```
ExecStart=/usr/bin/deluged -d -l /var/log/deluge/daemon.log -L warning
```

```
ExecStart=/usr/bin/deluge-web -d -l /var/log/deluge/web.log -L warning
```

See `deluged -h` for all available log-levels.

Restart the services:

```
sudo systemctl daemon-reload
sudo systemctl restart deluged
sudo systemctl restart deluge-web
```

Log rotation

To enable log rotation append `--logrotate` to the above `ExecStart` lines.

Conditionally start deluged for mount points

If you have a USB disk drive or network drive that may not be immediately available on boot or disconnected at random then you may want the `deluged` service to wait for mount point to be ready before starting. If they are unmounted or disconnected then `deluged` is stopped. When they become available again `deluged` is started.

Ensure you have added the correct drive details to `fstab` or equivalent so they are mounted at boot.

List the available drive mounts:

```
sudo systemctl -t mount
```

Look for your mount point in the `Description` column. Mounts are formatted similar to the mount point with `-s` replacing /s in the path. e.g.: `media-xyz.mount`

Modify the `[Unit]` section of the `deluged.service` script by adding the details below, substituting `xyz.mount` for the mount you want the service to depend on:

```
[Unit]
Description=Deluge Bittorrent Client Daemon
# Start after network and specified mounts are available.
After=network-online.target xyz.mount
Requires=xyz.mount
# Stops deluged if mount points disconnect
BindsTo=xyz.mount
```

For multiple mount points add a space between additional entries. e.g.:

```
After=network-online.target xyz.mount abc.mount def.mount
```

Modify the `[Install]` section to ensure the deluged service is started when the mount point comes back online:

```
[Install]
WantedBy=multi-user.target xyz.mount
```

Reference: systemd.unit

How to create launchd services for macOS

The following launchd script uses a separate user deluge, this is optional but recommended for security. To create a new deluge user follow the Apple help steps.

The paths to `deluged` and `deluge-web` assumes installation using Homebrew and will need modified if using other installation methods e.g. `Deluge.app`.

Daemon (deluged) service

Create the file /Library/LaunchDaemons/org.deluge-torrent.deluged.plist containing the following:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
↪PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>org.deluge-torrent.deluged</string>
    <key>ProgramArguments</key>
    <array>
        <string>/usr/local/bin/deluged</string>
        <string>-d</string>
        <string>-L</string>
        <string>error</string>
        <string>-l</string>
        <string>/var/log/deluged.log</string>
    </array>
    <key>StandardOutPath</key>
    <string>/tmp/deluged.stdout</string>
    <key>StandardErrorPath</key>
    <string>/tmp/deluged.stderr</string>
    <!-- To enable running as 'deluge' user remove comments.
    <key>UserName</key>
    <string>deluge</string>
    -->
    <key>RunAtLoad</key>
    <true/>
    <key>KeepAlive</key>
    <true/>
</dict>
</plist>
```

Set the service to load on startup and then start it:

```
sudo launchctl load -w /Library/LaunchDaemons/org.deluge-torrent.deluged.plist
sudo launchctl start org.deluge-torrent.deluged
```

Web UI (deluge-web) service

Create the file /Library/LaunchDaemons/org.deluge-torrent.deluge-web.plist containing the following:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
↪PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>org.deluge-torrent.deluge-web</string>
    <key>ProgramArguments</key>
```

(continues on next page)

```
    <array>
        <string>/usr/local/bin/deluge-web</string>
        <string>-d</string>
        <string>-L</string>
        <string>error</string>
        <string>-l</string>
        <string>/var/log/deluge-web.log</string>
    </array>
    <key>StandardOutPath</key>
    <string>/tmp/deluge-web.stdout</string>
    <key>StandardErrorPath</key>
    <string>/tmp/deluge-web.stderr</string>
    <!-- To enable running as 'deluge' user remove comments.
    <key>UserName</key>
    <string>deluge</string>
    -->
    <key>RunAtLoad</key>
    <true/>
    <key>KeepAlive</key>
    <true/>
</dict>
</plist>
```

Set the service to load on startup and then start it:

```
sudo launchctl load -w /Library/LaunchDaemons/org.deluge-torrent.deluge-web.plist
sudo launchctl start org.deluge-torrent.deluge-web
```

Logging

Logging is enabled by default in the above script, error level, and can be modified as required.

How to create NSSM services for Windows

Download NSSM and read their usage page about installing.

In order for the services to be stopped properly, use the debug application versions (ending in `-debug.exe`).

Daemon (deluged) service

Create a `deluged` service:

```
nssm install deluged
```

The following are minimum UI changes required for the service to work:

```
Path: C:\Program Files\Deluge\deluged-debug.exe
Arguments: -c C:\config_location
```

Web UI (deluge-web) service

```
nssm install deluge-web
```

```
Path: C:\Program Files\Deluge\deluge-web-debug.exe
Arguments: -c C:\config_location
```

If Web UI is not accessible outside your machine (if you're running Deluge from a home server), you have to whitelist Deluge in your Windows Firewall for the `deluge-web` and `deluge-web-debug` executable.

1.3 Release notes

A summary of the important changes in major releases of Deluge. For more details see the [changelog] or the git commit log.

1.3.1 Changelog

2.2.x (TBA)

Breaking changes

- Python 3.6 support removed (Python >= 3.7)

2.1.1 (2022-07-10)

Core

- Fix missing trackers added via magnet
- Fix handling magnets with tracker tiers

2.1.0 (2022-06-28)

Breaking changes

- Python 2 support removed (Python >= 3.6)
- libtorrent minimum requirement increased (>= 1.2).

Core

- Add support for SVG tracker icons.
- Fix tracker icon error handling.
- Fix cleaning-up tracker icon temp files.
- Fix Plugin manager to handle new metadata 2.1.
- Hide passwords in config logs.
- Fix cleaning-up temp files in add_torrent_url.
- Fix KeyError in sessionproxy after torrent delete.
- Remove libtorrent deprecated functions.
- Fix file_completed_alert handling.
- Add plugin keys to get_torrents_status.
- Add support for pygeoip dependency.
- Fix crash logging to Windows protected folder.
- Add is_interface and is_interface_name to validate network interfaces.
- Fix is_url and is_infohash error with None value.
- Fix load_libintl error.
- Add support for IPv6 in host lists.
- Add systemd user services.
- Fix refresh and expire the torrent status cache.
- Fix crash when logging errors initializing gettext.

Web UI

- Fix ETA column sorting in correct order (#3413).
- Fix defining foreground and background colors.
- Accept charset in content-type for json messages.
- Fix 'Complete Seen' and 'Completed' sorting.
- Fix encoding HTML entities for torrent attributes to prevent XSS.

Gtk UI

- Fix download location textbox width.
- Fix obscured port number in Connection Manager.
- Increase connection manager default height.
- Fix bug with setting move completed in Options tab.
- Fix adding daemon accounts.
- Add workaround for crash on Windows with ico or gif icons.

- Hide account password length in log.
- Added a torrent menu option for magnet copy.
- Fix unable to prefetch magnet in thinclient mode.
- Use GtkSpinner when testing open port.
- Update About Dialog year.
- Fix Edit Torrents dialogs close issues.
- Fix ETA being copied to neighboring empty cells.
- Disable GTK CSD by default on Windows.

Console UI

- Fix curses.init_pair raise ValueError on Py3.10.
- Swap j and k key's behavior to fit vim mode.
- Fix torrent details status error.
- Fix incorrect test for when a host is online.
- Add the torrent label to info command.

AutoAdd

- Fix handling torrent decode errors.
- Fix error dialog not being shown on error.

Blocklist

- Add frequency unit to interval label.

Notifications

- Fix UnicodeEncodeError upon non-ascii torrent name.

2.0.5 (2021-12-15)

WebUI

- Fix js minifying error resulting in WebUI blank screen.
- Silence erronous missing translations warning.

2.0.4 (2021-12-12)

Packaging

- Fix python optional setup.py requirements

Gtk UI

- Add detection of torrent URL on GTK UI focus
- Fix piecesbar crashing when enabled
- Remove num_blocks_cache_hits in stats
- Fix unhandled error with empty clipboard
- Add torrentdetails tabs position menu (#3441)
- Hide pygame community banner in console
- Fix cmp function for None types (#3309)
- Fix loading config with double-quotes in string
- Fix Status tab download speed and uploaded

Web UI

- Handle torrent add failures
- Add menu option to copy magnet URI
- Fix md5sums in torrent files breaking file listing (#3388)
- Add country flag alt/title for accessibility

Console UI

- Fix allowing use of windows-curses on Windows
- Fix hostlist status lookup errors
- Fix AttributeError setting config values
- Fix setting 'Skip' priority

Core

- Add workaround libtorrent 2.0 file_progress error
- Fix allow enabling any plugin Python version
- Export torrent get_magnet_uri method
- Fix loading magnet with resume_data and no metadata (#3478)
- Fix httpdownloader reencoding torrent file downloads (#3440)
- Fix lt listen_interfaces not comma-separated (#3337)

- Fix unable to remove magnet with delete_copies enabled (#3325)
- Fix Python 3.8 compatibility
- Fix loading config with double-quotes in string
- Fix pickle loading non-ascii state error (#3298)
- Fix creation of pidfile via command option
- Fix for peer.client UnicodeDecodeError
- Fix show_file unhandled dbus error

Documentation

- Add How-to guides about services.

Stats plugin

- Fix constant session status key warnings
- Fix cairo error

Notifications plugin

- Fix email KeyError with status name
- Fix unhandled TypeErrors on Python 3

Autoadd plugin

- Fix magnet missing applied labels

Execute plugin

- Fix failing to run on Windows (#3439)

2.0.3 (2019-06-12)

Gtk UI

- Fix errors running on Wayland (#3265).
- Fix Peers Tab tooltip and context menu errors (#3266).

Web UI

- Fix TypeError in Peers Tab setting country flag.
- Fix reverse proxy header TypeError (#3260).
- Fix request.base 'idna' codec error (#3261).
- Fix unable to change password (#3262).

Extractor plugin

- Fix potential error starting plugin.

Documentation

- Fix macOS install typo.
- Fix Windows install instructions.

2.0.2 (2019-06-08)

Packaging

- Add systemd deluged and deluge-web service files to package tarball (#2034)

Core

- Fix Python 2 compatibility issue with SimpleNamespace.

2.0.1 (2019-06-07)

Packaging

- Fix `setup.py` build error without git installed.

2.0.0 (2019-06-06)

Codebase

- Ported to Python 3

Core

- Improved Logging
- Removed the AutoAdd feature on the core. It's now handled with the AutoAdd plugin, which is also shipped with Deluge, and it does a better job and now, it even supports multiple users perfectly.
- Authentication/Permission exceptions are now sent to clients and recreated there to allow acting upon them.
- Updated SSL/TLS Protocol parameters for better security.
- Make the distinction between adding to the session new unmanaged torrents and torrents loaded from state. This will break backwards compatibility.
- Pass a copy of an event instead of passing the event arguments to the event handlers. This will break backwards compatibility.
- Allow changing ownership of torrents.
- File modifications on the auth file are now detected and when they happen, the file is reloaded. Upon finding an old auth file with an old format, an upgrade to the new format is made, file saved, and reloaded.
- Authentication no longer requires a username/password. If one or both of these is missing, an authentication error will be sent to the client which should then ask the username/password to the user.
- Implemented sequential downloads.
- Provide information about a torrent's pieces states
- Add Option To Specify Outgoing Connection Interface.
- Fix potential for host_id collision when creating hostlist entries.

Gtk UI

- Ported to GTK3 (3rd-party plugins will need updated).
- Allow changing ownership of torrents.
- Host entries in the Connection Manager UI are now editable.
- Implemented sequential downloads UI handling.
- Add optional pieces bar instead of a regular progress bar in torrent status tab.
- Make torrent opening compatible with all Unicode paths.
- Fix magnet association button on Windows.
- Add keyboard shortcuts for changing queue position:
    - Up: `Ctrl+Alt+Up`
    - Down: `Ctrl+Alt+Down`
    - Top: `Ctrl+Alt+Shift+Up`
    - Bottom: `Ctrl+Alt+Shift+Down`

Web UI

- Server (deluge-web) now daemonizes by default, use '-d' or '–do-not-daemonize' to disable.

- Fixed the '–base' option to work for regular use, not just with reverse proxies.

Blocklist Plugin

- Implemented whitelist support to both core and GTK UI.

- Implemented IP filter cleaning before each update. Restarting the deluge daemon is no longer needed.

- If "check_after_days" is 0(zero), the timer is not started anymore. It would keep updating one call after the other. If the value changed, the timer is now stopped and restarted using the new value.

1.3.2 Deluge 2.0 release notes

Welcome to the latest release of Deluge, a long time in the making!

What's new

Some of the highlights since the last major release.

- Migrated to Python 3 with minimal support retained for Python 2.7.

- Shiny new logo.

- Multi-user support.

- Performance updates to handle thousands of torrents with faster loading times.

- A New Console UI which emulates GTK/Web UIs.

- GTK UI migrated to GTK3 with UI improvements and additions.

- Magnet pre-fetching to allow file selection when adding torrent.

- Fully support libtorrent 1.2 release.

- Language switching support.

- Improved documentation hosted on ReadTheDocs.

- AutoAdd plugin replaces built-in functionality.

- Web UI now daemonizes by default so service scripts will require -d option.

Packaging

PyPi

As well as the usual source tarball available for download we now have published Deluge on the PyPi software repository.

- https://pypi.org/project/deluge/

Windows and MacOS

Unfortunately there are no packages yet for Windows or MacOS but they are being worked on. For now alternative install methods are available for testing.

Upgrade considerations

Deluge 2.0 is not compatible with Deluge 1.x clients or daemons so these will require upgrading too. Also third-party Python scripts may not be compatible if they directly connect to the Deluge client and will need migrating.

Always make a backup of your config before a major version upgrade to guard against data loss.

Translations may not be as up-to date so please help out, see *translations* page.

Plugins written for Deluge 1.3 will need upgrading for Deluge 2.0, due to the requirement of Python 3 and GTK3 UI. There is a *update plugin* document to help Plugin authors update their plugins.

1.4 Development & community

Deluge is an open-source project, and relies on its community of users to keep getting better.

1.4.1 Contributing code

Basic requirements and standards

- A new ticket is required for bugs or features. Search the ticket system first, to avoid filing a duplicate.
- Ensure code follows the *syntax and conventions*.
- Code must pass tests. See *testing* document for information on how to run and write unit tests.
- Commit messages are informative.

Pull request process:

- Fork us on GitHub.
- Clone your repository.
- Create a feature branch for your issue.
- Apply your changes:
    - Add them, and then commit them to your branch.
    - Run the tests until they pass.
    - When you feel you are finished, rebase your commits to ensure a simple and informative commit log.
- Create a pull request on GitHub from your forked repository.
    - Verify that the tests run by Travis-ci are passing.

Syntax and conventions

Code formatting

We use two applications to automatically format the code to save development time. They are both run with pre-commit.

Black

- Python

Prettier

- JavaScript
- CSS
- YAML
- Markdown

Common

- Line length: 79 chars.
- Indent: `4 spaces`, no tabs.
- All code should use `'single quotes'`.

Python

We follow PEP8 and Python Code Style which is adhered to with Black.

- Code '"must"' pass Black, flake8 and isort source code checkers. (Optionally Pylint)

```
flake8 deluge
isort -rc -df deluge
pylint deluge
pylint deluge/plugins/\*/deluge/
```

- Using the pre-commit application can aid in identifying issues while creating git commits.

Strings and bytes

To prevent bugs or errors in the code byte strings (`str`) must be decoded to strings (Unicode text strings, `unicode`) on input and then encoded on output.

Notes:

- PyGTK/GTK+ will accept `str` (UTF-8 encoded) or `unicode` but will only return `str`. See GTK3 Unicode docs.
- There is a `bytearray` type which enables in-place modification of a string. See Python Bytearrays
- Python 3 renames `unicode` to `str` type and byte strings become `bytes` type.

JavaScript

- Classes should follow the Ext coding style.

- Class names should be in !CamelCase

- Instances of classes should use camelCase.

Path separators

- All relative path separators used within code should be converted to posix format /, so should not contain \ or \\. This is to prevent confusion when dealing with cross-platform clients and servers.

Docstrings

All new docstrings must use Napoleon Google Style with old docstrings eventually converted over.

Google Style example:

```
def func(arg):
    """Function purpose.

    Args:
        arg (type): Description.

    Returns:
        type: Description. If the line is too, long indent next
            line with three spaces.
    """
    return
```

See complete list of supported headers.

Verify that the documentation parses correctly with:

```
python setup.py build_docs
```

1.4.2 Running tests

Testing uses PyTest framework and PyTest-Twisted to handle Twisted framework.

Testing

The tests are located in the source folder under `deluge/tests`. The tests are run from the project root directory. View the unit test coverage at: deluge-torrent.github.io

Pytest

```
pytest deluge/tests
pytest deluge/tests/test_client.py
pytest deluge/tests/test_client.py -k test_connect_localclient
```

Plugin

Running the tests for a specific plugin (requires pytest):

```
pytest deluge/plugins/<name-of-plugin>
```

Tox

All the tests for Deluge can be run using Tox

See available targets:

```
tox -l
py3
lint
docs
```

Run specific test:

```
tox -e py3
```

Verify code with pre-commit:

```
tox -e lint
```

CI

Deluge develop branch is tested automatically by GitHub actions.

When creating a pull request (PR) on github, units tests will be automatically be run.

1.4.3 Documentation contributions

Build

We use Sphinx to create the documentation from source files and docstrings in code.

```
pip install -r docs/requirements.txt
python setup.py build_docs
```

The resulting html files are in `docs/build/html`.

man pages

Located in `docs/man`

1.4.4 Translation contributions

Translators

For translators we have a [Launchpad translations](#) account where you can translate the `.po` files.

Marking text for translation

To mark text for translation in Python and ExtJS wrap the string with the function `_()` like this:

```
torrent.set_tracker_status(_("Announce OK"))
```

For GTK the text can also be marked translatable in the `glade/*.ui` files:

```
<property name="label" translatable="yes">Max Upload Speed:</property>
```

For more details see: [Python Gettext](#)

Translation process

These are the overall stages in gettext translation:

`Portable Object Template -> Portable Object -> Machine Object`

- The `deluge.pot` is created using `generate_pot.py`.
- Upload `deluge/i18n/deluge.pot` to [Launchpad translations](#).
- Give the translators time to translate the text.
- Download the updated `.po` files from translation site.
- Extract to `deluge/i18n/` and strip the `deluge-` prefix:

  ```
  rename -f 's/^deluge-//' deluge-*.po
  ```

- The binary `MO` files for each language are generated by `setup.py` using the `msgfmt.py` script.

To enable Web UI to use translations update `gettext.js` by running `gen_gettext.py` script.

Useful applications

- podiff - Compare textual information in two PO files
- gtranslator - GUI PO file editor
- Poedit - GUI PO file editor

Testing translation

Testing that translations are working correctly can be performed by running Deluge as follows.

Create an `MO` for a single language in the correct sub-directory:

```
mkdir -p deluge/i18n/fr/LC_MESSAGES
python msgfmt.py -o deluge/i18n/fr/LC_MESSAGES/deluge.mo deluge/i18n/fr.po
```

Run Deluge using an alternative language:

```
LANGUAGE=fr deluge
LANGUAGE=ru_RU.UTF-8 deluge
```

Note: If you do not have a particular language installed on your system it will only translate based on the `MO` files for Deluge so some GTK text/button strings will remain in English.

1.5 Development guide

This is a guide to help with developing Deluge.

1.5.1 Developer tutorials

A list of articles to help developers get started with Deluge.

Setup tutorial for Deluge development

The aim of this tutorial is to download the source code and setup an environment to enable development work on Deluge.

Pre-requisites

To build and run the Deluge applications they depends on tools and libraries as listed in DEPENDS.md.

Almost all of the Python packages dependencies will be installed using pip but there are some packages or libraries that are required to be installed to the system.

Ubuntu

Build tools

```
sudo apt install git intltool closure-compiler python3-pip
pip3 install --user tox
```

You might need to add ~/.local/bin to your PATH.

Runtime libraries and tools

```
sudo apt install python3-libtorrent python3-geoip python3-dbus  python3-gi \
python3-gi-cairo gir1.2-gtk-3.0 gir1.2-ayatanaappindicator3-0.1 python3-pygame⏎
↪libnotify4 \
librsvg2-common xdg-utils
```

Setup development environment

Clone Deluge git repository

Download the latest git code to local folder.

```
git clone git://deluge-torrent.org/deluge.git
cd deluge
```

Create Python virtual environment

Creation of a [Python virtual environment] keeps the development isolated and easier to maintain and Tox has an option to make this process easier:

```
tox -e denv
```

Activate virtual environment:

```
source .venv/bin/activate
```

Deluge will be installed by Tox in *develop* mode which creates links back to source code so that changes will be reflected immediately without repeated installation. Check it is installed with:

```
(.venv) $ deluge --version
deluge-gtk 2.0.0b2.dev149
libtorrent: 1.1.9.0
Python: 2.7.12
OS: Linux Ubuntu 16.04 xenial
```

Setup pre-commit hook

Using pre-commit ensures submitted code is checked for quality when creating git commits.

```
(.venv) $ pre-commit install
```

You are now ready to start playing with the source code.

Reference

- Contributing
- [Key requirements concepts]

1.5.2 How-to guides

A collection of guides for specific issues or to cover more detail than the tutorials.

Web JSON-RPC

How to connect to JSON-RPC with curl

Before continuing make sure `deluge-web` or Web UI plugin is running.

Create a curl configuration file

To save a lot of typing and to keep the curl command short we shall create a `curl.cfg` files and put the following contents in it:

```
request = "POST"
compressed
cookie = "cookie_deluge.txt"
cookie-jar = "cookie_deluge.txt"
header = "Content-Type: application/json"
header = "Accept: application/json"
url = "http://localhost:8112/json"
write-out = "\n"
```

To pretty-print the JSON result see: https://stackoverflow.com/q/352098/175584

Log in to Web UI

Log in to the Web UI and get session cookie:

```
curl -d '{"method": "auth.login", "params": ["deluge"], "id": 1}' -K curl.cfg
```

Result is `true` to signify that login was successful:

```
{
    "error": null,
    "id": 1,
    "result": true
}
```

Check the contents of the cookie file to verify session ID created.

```
cat cookie_deluge.txt
# Netscape HTTP Cookie File
# http://curl.haxx.se/docs/http-cookies.html
# This file was generated by libcurl! Edit at your own risk.


localhost       FALSE       /json       FALSE       1540061203       _session_
→id       <session_id>
```

Check connected to deluged

Use the `web.connected` method to get a boolean response if the Web UI is connected to a deluged host:

```
curl -d '{"method": "web.connected", "params": [], "id": 1}' -K curl.cfg
```

Result is `false` because Web UI is not yet connected to the daemon:

```
{
    "error": null,
    "id": 1,
    "result": false
}
```

Get list of deluged hosts

Use the `web.get_hosts` method:

```
curl -d '{"method": "web.get_hosts", "params": [], "id": 1}' -K curl.cfg
```

The result contains the <hostID> for using in request `params` field.

```
{
    "error": null,
    "id": 1,
    "result": [
        [
            "<hostID>",
            "127.0.0.1",
            58846,
            "localclient"
        ]
    ]
}
```

Get the deluged host status

```
curl -d '{"method": "web.get_host_status", \
"params": ["<hostID>"], "id": 1}' -K curl.cfg
```

The result shows the version and status; *online*, *offline* or *connected*.

```
{
    "error": null,
    "id": 1,
    "result": [
        "<hostID>",
        "Online",
        "2.0.0"
    ]
}
```

Connect to deluged host

To connect to deluged with `<hostID>`:

```
curl -d '{"method": "web.connect", \
"params": ["<hostID>"], "id": 1}' -K curl.cfg
```

The result contains the full list of available host methods:

```
{
    "error": null,
    "id": 1,
    "result": [
        "core.add_torrent_url",
        ...
        "core.upload_plugin"
    ]
}
```

Disconnect from host

```
curl -d '{"method": "web.disconnect", "params": [], "id": 1}' -K curl.cfg
```

A successful result:

```
{
    "error": null,
    "id": 1,
    "result": "Connection was closed cleanly."
}
```

Add a torrent

```
curl -d '{"method": "web.add_torrents", "params": \
[[{"path":"/tmp/ubuntu-12.04.1-desktop-amd64.iso.torrent", \
"options":null}]], "id": 1}' -K curl.cfg
```

Add a magnet URI

```
curl-d '{"method": "core.add_torrent_magnet", \
"params": ["<magnet_uri>", {}], "id": 1}' -K curl.cfg
```

Get list of files for a torrent

```
curl -d '{"method": "web.get_torrent_files", \
"params": ["<torrentid>"], "id": 1}' -K curl.cfg
```

Set a core config option

```
curl -d '{"method": "core.set_config", \
"params":[{"max_upload_slots_global":"200"}], "id": 1}' -K curl.cfg

{"error": null, "result": null, "id": 1}
```

Useful curl configuration options

For full list of options see man page `man curl` or help `curl --help`:

```
--cookie (-b) # Load cookie file with session id
--cookie-jar (-c) # Save cookie file with session id
--compressed # responses are gzipped
--include (-i) # Include the HTTP header in output (optional)
--header (-H) # HTTP header
--request (-X) # custom request method
--data (-d) # data to send in POST request '{"method": "", "params": [], "id": ""}'
--insecure (-k) # use with self-signed certs https
```

Plugins

How to update a Deluge 1.3 plugin for 2.0

With the new code in Deluge 2.0 there are changes that require authors of existing plugins to update their plugins to work on Deluge 2.0.

The main changes are with Python 3 support and the new GTK3 user interface with the dropping of GTK2. However it is still possible for a 1.3 plugin to be made compatible with 2.0 and this guide aims to helps with that process.

Python

Python version matching

Ensure your code is Python >=3.6 compatible.

In `1.3-stable` the plugins that were built with a specific version of Python would only be loaded if the system Python also matched.

This has change in Deluge 2.0 and it will load any Python version of plugin eggs so compatibility is essential for end-users not to encounter issues.

GTK 3 addition

In order to support both Deluge 1.3 and 2.0 all existing plugin GTK UI files must be copied and then converted to contain only GTK3 code with the old files still using PyGTK e.g.:

```
cp gtkui.py gtk3ui.py
```

Convert from libglade to GtkBuilder

With PyGTK there were two library options for creating the user interface from XML files by default Deluge plugins used libglade but that has been deprecated and removed in GTK3. So the libglade `.glade` files will need converted to GtkBuilder `.ui` files and the Python code updated.

See the official Migrating to GtkBuilder document for more details.

GtkBuilder conversion script

Install the `gtk-builder-convert` converter on Ubuntu with:

```
sudo apt install libgtk2.0-dev
```

To convert your GTK run it like so:

```
gtk-builder-convert data/config.glade data/config.ui
```

Glade UI designer for GTK2

The above conversion can also be done in Glade UI designer (version <=3.8).

In the preferences select `GtkBuilder` as the project file format. Ensure that the minimum Gtk version is set to 2.24 and fix any deprecated widgets.

The updated file should be saved with file extension `.ui`.

Python code changes

The code needs to replace `gtk.glade` references with `gtk.Builder` and the first step is updating how the files are loaded:

```
- glade = gtk.glade.XML(get_resource("config.glade"))
+ builder = Gtk.Builder.new_from_file(get_resource("config.ui"))
```

Replace signals method:

```
- glade.signal_autoconnect(self)
+ builder.connect_signals(self)
```

Replace `get_widget` with `get_object`:

```
- glade.get_widget
+ builder.get_object
```

Check for any remaining `glade` methods and replace with the `builder` equivalents.

Migrate XML files to GTK3

If you open and save the file it will update with the new requirement header:

```
<!-- Generated with glade 3.18.3 -->
<interface>
  <requires lib="gtk+" version="3.10"/>
```

You can fix deprecated widgets but keep the minimum GTK version to <= 3.10 for desktop compatibility.

An example of migrating a Deluge plugin to GtkBuilder: AutoAdd GtkBuilder

Gtk import rename

Move from PyGTK to GTK3 using Python bindings.

https://pygobject.readthedocs.io/en/latest/guide/porting.html

```
wget https://gitlab.gnome.org/GNOME/pygobject/raw/master/tools/pygi-convert.sh
cp gtkui.py gtk3ui.py
sh pygi-convert.sh gtk3ui.py
```

```
-import gtk
+from gi.repository import Gtk
```

```
- self.builder = gtk.Builder()
+ self.builder = Gtk.Builder()
```

Deluge GTK3

Imports

Imports will need renamed from `deluge.ui.gtkui` to `deluge.ui.gtk3`.

There is also a new PluginBase for Gtk3 UI:

```
-from deluge.plugins.pluginbase import GtkPluginBase
+from deluge.plugins.pluginbase import Gtk3PluginBase
-class GtkUI(GtkPluginBase):
+class Gtk3UI(Gtk3PluginBase):
```

Entry points

To enable the GTK3 UI to find the plugin the entry points requires updating too.

In the plugin `__init__.py` file add a new `Gtk3UIPlugin` class:

```
class Gtk3UIPlugin(PluginInitBase):
    def __init__(self, plugin_name):
        from .gtk3ui import Gtk3UI as _plugin_cls
        self._plugin_cls = _plugin_cls
        super(Gtk3UIPlugin, self).__init__(plugin_name)
```

A new entry for GTK3 UI can then be added to `setup.py`:

```
     [deluge.plugin.gtkui]
     %s = %s:GtkUIPlugin
+    [deluge.plugin.gtk3ui]
+    %s = deluge_%s:Gtk3UIPlugin
     [deluge.plugin.webui]
     %s = %s:WebUIPlugin
-    """ % ((__plugin_name__, __plugin_name__.lower())*3)
+    """ % ((__plugin_name__, __plugin_name__.lower())*4)
```

1.5.3 Packaging documentation

Release Checklist

Pre-release

- Update *translation* po files from Launchpad account.
- Changelog is updated with relevant commits and release date is added.
- Docs *release notes* are updated.
- Tag release in git and push upstream e.g.

  ```
  git tag -a deluge-2.0.0 -m "Deluge 2.0.0 Release"
  ```

Release

- Create source and wheel distributions:

```
python setup.py sdist bdist_wheel
```

- Upload to PyPi (only accepts `tar.gz`):

```
twine upload dist/deluge-2.0.0.tar.gz dist/deluge-2.0.0-py3-none-any.whl
```

- Calculate `sha256sum` for each file e.g.

```
cd dist; sha256sum deluge-2.0.0.tar.xz > deluge-2.0.0.tar.xz.sha256
```

- Upload source tarballs and packages to `ftp-osl.osuosl.org`.

    - Ensure file permissions are global readable: `0644`

    - Sub-directories correspond to *major.minor* version e.g. all `2.0.x` patch releases are stored in `source/2.0`.

    - Change release version in `version*` files, create a new version file for any major releases.

    - SSH into OSUOSL FTP site and run `trigger-deluge` to sync files.

- Create packages (Ubuntu, Windows, OSX).

    - Ubuntu: https://code.launchpad.net/~deluge-team/+recipe/stable-releases

        * Ensure launchpad git repo has sync'd to get latest version

        * Update version in recipe (reset any dash number to 0)

        * Check for new distribution series needing selected.

        * Request build for selected series.

Post-release

- Update with version, hashes and release notes:

    - Publish new docs version on ReadTheDocs.

    - Wikipedia

- Close Trac milestone and add new milestone version for future tickets.

- Ensure all stable branch commits are also applied to development branch.

Launchpad recipe

The launchpad build recipes are for build from source automatically to provide Ubuntu packages. They are used to create daily builds of a Deluge git branch.

Note these don't have the same control as a creating a publishing to PPA.

Main reference: https://help.launchpad.net/Packaging/SourceBuilds/Recipes

Deluge Launchpad build recipes

Recipe configuration: https://code.launchpad.net/~deluge-team/+recipes

An example for building the develop branch:

```
# git-build-recipe format 0.4 deb-version 2.0.0.dev{revno}+{git-commit}+{time}
lp:deluge develop
nest-part packaging lp:~calumlind/+git/lp_deluge_deb debian debian develop
```

There are two parts, first to get the source code branch and then the `debian` files for building the package.

Testing and building locally

Create a `deluge.recipe` file with the contents from launchpad and create the build files with `git-build-recipe`:

```
git-build-recipe --allow-fallback-to-native deluge.recipe lp_build
```

Setup pbuilder and build the deluge package:

```
sudo pbuilder build lp_build/deluge*.dsc
```

Alternatively to build using local files with pdebuild:

```
cd lp_build/deluge/deluge
pdebuild
```

This will allow modifying the `debian` files to test changes to `rules` or `control`.

Packaging for Windows

Currently there is no working package for Deluge 2.0. The previous Python freezing application `bbfreeze` is not compatible with Python 3 and the project is no longer maintained.

There are two alternatives `cxfreeze` and `pyinstaller` but neither is trivial with the GTKUI application.

See #3201

1.5.4 Deluge dependencies

The following are required to install and run Deluge. They are separated into sections to distinguish the precise requirements for each module.

All modules will require the common section dependencies.

Prerequisite

- Python >= *3.6*

Build

- setuptools
- intltool - Optional: Desktop file translation for *nix.
- closure-compiler - Minify javascript (alternative is rjsmin)

Common

- Twisted >= *17.1* - Use TLS extras for `service_identity` and `idna`.
- OpenSSL >= *1.0.1*
- pyOpenSSL
- rencode >= *1.0.2* - Encoding library.
- PyXDG - Access freedesktop.org standards for *nix.
- xdg-utils - Provides xdg-open for *nix.
- zope.interface
- chardet - Optional: Encoding detection.
- setproctitle - Optional: Renaming processes.
- Pillow - Optional: Support for resizing tracker icons.
- dbus-python - Optional: Show item location in filemanager.
- ifaddr - Optional: Verify network interfaces.

Linux and BSD

- distro - Optional: OS platform information.

Windows OS

- pywin32
- certifi

Core (deluged daemon)

- libtorrent *>= 1.2.0*
- GeoIP or [pygeoip] - Optional: IP address country lookup. (*Debian:* `python-geoip`)

GTK UI

- GTK+ >= 3.10
- PyGObject
- Pycairo
- librsvg *>= 2*
- ayatanaappindicator3 w/GIR - Optional: Ubuntu system tray icon.

MacOS

- GtkOSXApplication

Web UI

- mako

Plugins

Notifications

- pygame - Optional: Play sounds
- libnotify w/GIR - Optional: Desktop popups.

1.6 Reference

Technical reference material.

1.6.1 Deluge Web UI

The Deluge web interface is a full featured interface built using the ExtJS framework, running on top of a Twisted web server.

SSL Configuration

By default the web interface will use the same private key and certificate as the Deluge daemon. You can use a different certificate/key and specify it in the Web UI config, see below for details.

Create SSL Certificate Examples

Sample guide: How to Create a SSL Certificate

Linux

```
openssl req -new -x509 -nodes -out deluge.cert.pem -keyout deluge.key.pem
```

Windows

```
C:\OpenSSL\bin\openssl.exe req -config C:\OpenSSL\bin\openssl.cnf -x509 -days 365 -
↪newkey rsa:1024 -keyout hostkey.pem -nodes -out hostcert.pem
```

Enable Web UI SSL

There are two ways to enable SSL encryption in the web server:

- Specify in your config (accessible via the Preferences window).
- Use `--ssl` when running the web server, overriding the configuration value to enable SSL.

Enable Development mode

Append `?dev=true` to the Web UI URL to enable development mode, uses the source JavaScript files (if available) rather than compressed versions:

```
http://127.0.0.1:8112/?dev=true
```

1.6.2 Deluge RPC

Message Formats

DelugeRPC is a protocol used for daemon/client communication. There are four types of messages involved in the protocol: RPC Request, RPC Response, RPC Error and Event. All messages are zlib compressed with rencode encoded strings and their data formats are detailed below.

RPC Request

This message is created and sent by the client to the server requesting that a remote method be called. Multiple requests can be bundled in a list.

[[request_id, method, [args], {kwargs}], . . .]

request_id (int)
> An integer determined by the client that is used in replies from the server. This is used to ensure the client knows which request the data is in response to. Another alternative would be to respond in the same order the requests come in, but this could cause lag if an earlier request takes longer to process.

method (str)
> The name of the remote method to call. This name can be in dotted format to call other objects or plugins methods.

args (list)
> The arguments to call the method with.

kwargs (dict)
> The keyword arguments to call the method with.

RPC Response

This message is created and sent in response to a RPC Request from a client. It will hold the return value of the requested method call. In the case of an error, a RPC Error message will be sent instead.

[message_type, request_id, [return_value]]

message_type (int)
> This will be a RPC_RESPONSE type id. This is used on the client side to determine what kind of message is being received from the daemon.

request_id (int)
> The request_id is the same as the one sent by the client in the initial request. It used on the client side to determine what message this is in response to.

return_value (list)
> The return value of the method call.

RPC Error

This message is created in response to an error generated while processing a RPC Request and will serve as a replacement for a RPC Response message.

[message_type, request_id, exception_type, exception_msg, traceback]

message_type (int)
> This will be a RPC_ERROR type id.

request_id (int)
> The request_id is the same as the one sent by the client in the initial request.

exception_type (str)
> The type of exception raised.

exception_msg (str)
> The message as to why the exception was raised.

traceback (str)
  The traceback of the generated exception.

Event

This message is created by the daemon and sent to the clients without being in response to a RPC Request. Events are generally sent for changes in the daemon's state that the clients need to be made aware of.

[message_type, event_name, data]

message_type (int)
  This will be a RPC_EVENT type id.

event_name (str)
  This is the name of the event being emitted by the daemon.

data (list)
  Additional data to be sent with the event. This is dependent upon the event being emitted.

1.6.3 Deluge RPC API

  - *Deluge RPC*

1.6.4 Deluge Web JSON-RPC API

  - Spec: JSON-RPC v1

  - URL: `/json`

  - *Deluge RPC API*

class deluge.ui.web.json_api.**WebApi**
  The component that implements all the methods required for managing the web interface. The complete web json interface also exposes all the methods available from the core RPC.

  add_host(*host*, *port*, *username=''*, *password=''*)
    Adds a host to the list.

      Parameters

        - **host** (*str*) – The IP or hostname of the deluge daemon.

        - **port** (*int*) – The port of the deluge daemon.

        - **username** (*str*) – The username to login to the daemon with.

        - **password** (*str*) – The password to login to the daemon with.

      Returns

        tuple –

        A tuple of (bool, str). If True will contain the host_id, otherwise
          if False will contain the error message.

  add_torrents(*torrents*)
    Add torrents by file

> **Parameters**
> **torrents** (`list`) – A list of dictionaries containing the torrent path and torrent options to add with.

```
json_api.web.add_torrents([{
    "path": "/tmp/deluge-web/some-torrent-file.torrent",
    "options": {"download_location": "/home/deluge/"}
}])
```

connect(*host_id*)

Connect the web client to a daemon.

> **Parameters**
> **host_id** (`str`) – The id of the daemon in the host list.
>
> **Returns**
> *Deferred* – List of methods the daemon supports.

connected()

The current connection state.

> **Returns**
> True if the client is connected
>
> **Return type**
> boolean

deregister_event_listener(*event*)

Remove an event listener from the event queue.

> **Parameters**
> **event** (`string`) – The event name

disconnect()

Disconnect the web interface from the connected daemon.

download_torrent_from_url(*url*, *cookie=None*)

Download a torrent file from a URL to a temporary directory.

> **Parameters**
> **url** (`string`) – the URL of the torrent
>
> **Returns**
> the temporary file name of the torrent file
>
> **Return type**
> string

edit_host(*host_id*, *host*, *port*, *username=''*, *password=''*)

Edit host details in the hostlist.

> **Parameters**
>
> - **host_id** (`str`) – The host identifying hash.
>
> - **host** (`str`) – The IP or hostname of the deluge daemon.
>
> - **port** (`int`) – The port of the deluge daemon.
>
> - **username** (`str`) – The username to login to the daemon with.
>
> - **password** (`str`) – The password to login to the daemon with.

> > **Returns**
> > > *bool* – True if successful, False otherwise.

get_config()

> Get the configuration dictionary for the web interface.

> > **Return type**
> > > dictionary

> > **Returns**
> > > the configuration

get_events()

> Retrieve the pending events for the session.

get_host_status(*host_id*)

> Returns the current status for the specified host.

> > **Parameters**
> > > **host_id** (*string*) – the hash id of the host

get_hosts()

> Return the hosts in the hostlist.

get_magnet_info(*uri*)

> Parse a magnet URI for hash and name.

get_plugin_info(*name*)

> Get the details for a plugin.

get_plugin_resources(*name*)

> Get the resource data files for a plugin.

get_plugins()

> All available and enabled plugins within WebUI.

> **Note:** This does not represent all plugins from deluge.client.core.

> > **Returns**
> > > *dict* – A dict containing 'available_plugins' and 'enabled_plugins' lists.

get_torrent_files(*torrent_id*)

> Gets the files for a torrent in tree format

> > **Parameters**
> > > **torrent_id** (*string*) – the id of the torrent to retrieve.

> > **Returns**
> > > The torrents files in a tree

> > **Return type**
> > > dictionary

get_torrent_info(*filename*)

> Return information about a torrent on the filesystem.

> > **Parameters**
> > > **filename** (*string*) – the path to the torrent

> **Returns**
> information about the torrent:

```
{
    "name": the torrent name,
    "files_tree": the files the torrent contains,
    "info_hash" the torrents info_hash
}
```

> **Return type**
> dictionary

get_torrent_status(*torrent_id*, *keys*)

> Get the status for a torrent, filtered by status keys.

register_event_listener(*event*)

> Add a listener to the event queue.
>
> > **Parameters**
> > **event** (*string*) – The event name

remove_host(*host_id*)

> Removes a host from the hostlist.
>
> > **Parameters**
> > **host_id** (*str*) – The host identifying hash.
> >
> > **Returns**
> > *bool* – True if successful, False otherwise.

set_config(*config*)

> Sets the configuration dictionary for the web interface.
>
> > **Parameters**
> > **config** (*dictionary*) – The configuration options to update

set_theme(*theme*)

> Sets a new Theme to the WebUI
>
> > **Parameters**
> > **theme** (*str*) – the theme to apply

start_daemon(*port*)

> Starts a local daemon.

update_ui(*keys*, *filter_dict*)

> Gather the information required for updating the web interface.
>
> > **Parameters**
> > - **keys** (*list*) – the information about the torrents to gather
> > - **filter_dict** (*dictionary*) – the filters to apply when selecting torrents.
> >
> > **Returns**
> > The torrent and UI information.
> >
> > **Return type**
> > dictionary

> **upload_plugin**(*filename*, *path*)
>
>> Upload a plugin to config.

class `deluge.ui.web.json_api.`**WebUtils**

> Utility functions for the Web UI that do not fit in the WebApi.
>
>> **get_languages**()
>>
>>> Get the available translated languages
>>>
>>>> **Returns**
>>>>
>>>>> *list* – of tuples [`(lang-id, language-name), ...`]
>>
>> **get_themes**()
>>
>>> Get the available themes
>>>
>>>> **Returns**
>>>>
>>>>> *list* – of themes [`theme1, theme2, ...`]

- genindex

- modindex