Ultros Documentation

Release 1.1.0

2015 Gareth Coles, Sean Gordon

July 22, 2018

Contents

Ultros is a multi-protocol IRC bot designed from the ground up with extensibility and ease-of-use in mind.

This site will serve as your usage documentation. We'll try to keep this up to date, but please bear in mind that writing documentation is time-consuming and that nobody on the team enjoys doing it.

Are you a developer? Maybe you're looking for the api.

This documentation is **under heavy development** right now. If you get stuck, please feel free to join us on IRC (**irc://irc.esper.net/Ultros**). Likewise, if you use Ultros a lot and are good at writing documentation, come talk to us or submit some pull requests. We'd appreciate it.

CHAPTER 1

Installation

Over time, Ultros' installation steps have changed significantly. Ultros attempts to be as cross-platform as possible, but it would also be true that Ultros does target Linux as its primary platform. Special instructions have been included for other platforms below.

If you're just looking to set up your bot, please refer to the configuration page.

1.1 Requirements

Ultros' optimal environment requires the following.

- Git
- Python 2.6 or 2.7 (Mac users, see below for installation instructions)
- Pip and Virtualenv
- The latest OpenSSL distribution for your system

Once you have all of the above installed, you'll want to set up a Virtualenv. Please read the Virtualenv docs for more info. While setting one up is an optional step, we feel that it's a good idea to keep sets of packages separate from other system applications, so we do recommend using it.

1.2 Windows

To install on Windows, you'll need to set up a few extra things.

- Download and install the latest version of Python 2.x (**Not Python 3.x**) from the Python website.
- Follow these instructions to add Python to your PATH.
- Install Twisted from the Twisted site, making sure you pick the 32-bit or 64-bit version as appropriate.
- Download and install OpenSSL for Windows from this site, making sure to pick the latest full release - **not** the "Light" version.

- Download and install PyOpenSSL from here, making sure you pick the 32-bit or 64-bit version as appropriate.
- Install the latest version of MSysGit, either allowing it to install to System32, or adding the install location to your PATH like you did for Python.

Once you've done the above, open a command prompt (**cmd.exe** if you're using the run box), and do the following.

1. **cd** to the location you're installing Ultros to.
2. **Clone the respository** from https://github.com/UltrosBot/Ultros.git
3. **cd** to the newly-created **Ultros/** folder.
4. **python -m ensurepip** to ensure that pip is installed.
5. If you're using Virtualenv, this is a good time to set it up, according to the Virtualenv docs.
6. Run **pip install -r requirements.txt**.

Assuming all of the above completed successfully, you should now be ready to configure Ultros - see the configuration page for more information on this. The above steps may be summarized within this simple batch script

```
@ECHO off
:: Download Ultros
cd C:\path\to\install\ultros\to
git clone https://github.com/UltrosBot/Ultros.git
cd Ultros

:: Install pip
python -m ensurepip

:: If you're not using Virtualenv, you can skip this section
pip install virtualenv
virtualenv venv
venv\Scripts\activate

:: Finally, set up requirements
pip install -r requirements.txt
```

1.3 Linux

Linux is our preferred operating system, and we highly recommend that you use it to host your bot. To do so, you'll need the following.

- Git
- **Python 2.7**
  - Most distributions will install Python 2 by default when you specify **python**, however some distributions (such as Arch) will install Python 3. Ultros does not support Python 3 and will not be able to until Twisted does, so be careful of this.
  - You'll also need the development headers, usually from the corresponding **-dev** package, as well as pip, often from the corresponding **-pip** package.
- **libffi** and **libffi-dev**
- The latest version of OpenSSL
- Your distro's equivalent of build-essential (A C compiler and headers)

Once you have all of the above installed, you may proceed to download and set up Ultros as follows:

```
# Download Ultros
cd /path/to/install/ultros/to
git clone https://github.com/UltrosBot/Ultros.git
cd Ultros

# If you're not using Virtualenv, you can skip this section
pip install virtualenv
virtualenv venv
source venv/bin/activate

# Finally, set up requirements
pip install -r requirements.txt
```

Assuming all of the above completed successfully, you should now be ready to configure Ultros - see the configuration page for more information on this.

Warning: We highly recommend that you **do not run Ultros as root**. It does not require administrator privileges, and you should not grant it access to them. You may like to create a separate user for Ultros, which will also provide you with a convenient location to store it.

1.4 Mac OSX

Note: These instructions are for Mavericks (10.9), and may differ slightly for different versions of OSX.

You'll need to do a few things before you can set up Ultros.

1. Install Homebrew, if you haven't already.
2. Set up your environment as shown here.
3. Open Terminal.app and run the following

   ```
   brew install git
   brew install python
   ```

   This may take a while to complete, and may also require you to update Xcode. However, you should install Python this way instead of downloading it from the Python website.

Now you're able to set up Ultros.

```
# Download Ultros
cd /path/to/install/ultros/to
git clone https://github.com/UltrosBot/Ultros.git
cd Ultros

# If you're not using Virtualenv, you can skip this section
pip install virtualenv
virtualenv venv
source venv/bin/activate

# Finally, set up requirements
pip install -r requirements.txt
```

Assuming all of the above completed successfully, you should now be ready to configure Ultros - see the configuration page for more information on this.

CHAPTER 2

Plugins

This section of the documentation will serve as your information on how to set up and use the plugins that are included with Ultros. This does not include the packages in the contrib repo - Go there for information on those, as each package has its documentation in the repo for the time being.

2.1 URLs

One of Ultros' most-used plugins is the URLs plugin. This plugin is in charge of handling URLs (aka links) that are shared within any channels Ultros is a part of. By default, all this plugin will do is attempt to find the page title of a standard http or https URL, but other plugins (such as URL-tools) may add special handling for specific sites.

This plugin requires that the **Auth** plugin be loaded and available, and that a permissions manager be set up.

2.1.1 Getting started

The first thing you'll want to do is head into **config/plugins** and copy **urls.yml.example** to **urls.yml**. The configuration has sensible defaults, so you're free to stop here if you just want to plug-and-go.

If not, open the file and configure it to your liking, following the guidelines below.

```
spoofing:  # Sites to spoof differently. Normally we spoof Firefox, you can set an alternative string for user-agent spoofing, or disable it entirely with False.
  soundcloud.com: False
```

This section is all about user-agent spoofing. Spoofing is necessary so that websites respond to us as if we're a real web browser - Firefox by default. In this section, you can set a different user-agent string for specific domains, or disable spoofing by setting this to **False** - This is necessary for sites like Soundcloud, which use Javascript to set the page title when a real browser is detected, but simply places it in the HTML otherwise.

```
# The default user-agent to use for all domains that don't have custom spoofing
default_user_agent: "Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.0
↪"
```

Further to this, you may set the default user-agent string to use for all websites that haven't been placed in the spoofing setting above. The default is Firefox's user-agent string.

```
redirects:
  max: 15

  domains:  # Only these domains will be allowed for pre-handler redirects
  - "5z8.info"  # shadyurl.com
  - "bit.ly"
  - "cli.gs"
  - "db.tt"
  - "deck.ly"
  - "dickensurl.com"
  - "fb.me"
  - "fur.ly"
  - "gg.gg"
  - "git.io"
  - "goo.gl"
  - "is.gd"
  - "mcaf.ee"
  - "nazr.in"  # Pssh
  - "owl.ly"
  - "redd.it"
  - "su.pr"
  - "t.co"
  - "tinyurl.com"
  - "turl.ca"
  - "vurl.com"
  - "waa.ai"
  - "youtu.be"
```

This section is all about pre-handler redirects. To understand what this means, you should understand that the URLs plugin works by registering handlers for different sites based on various different criteria. Before those handlers are run, however, we can attempt to resolve any redirects presented by URL shortening services and other sites.

- **max**: The maximum number of redirects to follow before giving up. Set this to a reasonably low amount.
- **domains**: A whitelist of domains to follow redirects for before handlers are run. Regular expressions are not used.

```
max_title_length: 150  # Truncate titles that are longer than this - note that this␣
↪only applies
                         # to the title itself, not the message containing it
```

For the default website handler only: The maximum length of a title to be sent to a chat network. As the configuration states, note that this is the length of the title as shown on the page - the actual message sent to the chat network will be slightly larger to accommodate the domain info.

```yaml
blacklist: []  # List of patterns to match against URLs; if matched then the URL will␣
↪be ignored.
```

The blacklist is used to ignore different URls based on regular expressions. The full URL will be tested against each regular expression in this list, and if one of them matches, will be completely ignored by the plugin. For example, you may want to ignore **git.io** URLs if the bot is in channels where those URLs are pasted a lot - such as by another bot.

If you don't understand regular expressions, we recommend the excellent Learn Regex the Hard Way by Zed. A. Shaw - Although it only goes up to exercise 16, it's a great place to start.

Please also note that when you're writing regular expressions in YAML, you should surround them with 'single quotes', so that YAML will not try to directly handle any regex escapes you use.

```yaml
default_shortener: tinyurl  # The default shortener for channels without one set
                              # This will revert to tinyurl if the shortener doesn't␣
↪exist
```

For the URL shortening part of the plugin, this is the default handler to use. This plugin only provides a TinyURL shortener by default, which is used whenever the shortener you set here can't be found, but plugins such as URL-tools may add other shorteners, which you may use here instead.

Note that this doesn't override the per-channel shorteners (which will be covered later) unless they're missing.

```yaml
accept_language:
  # This section is entirely optional
  # default: "en"  # Sent for any site not in the list below
  domains:  # Leave out the starting "www."
    "example.com": "en-GB,en;q=0.9"
```

This section is all about languages. Some websites will look out for a header named **Accept-Language** and try to serve their website using the specified language. You may set the default language requested here, as well as specifying specific languages for separate domains, if you so wish. This may be of particular interest to users that aren't native English speakers.

```yaml
sessions:  # Sessions allow cookies to be stored during requests and retrieved later␣
↪on
            # All matching is done using regular expressions - https://docs.python.
↪org/2/library/re.html
            # This sessions config doesn't apply to extra URL handlers - they're in␣
↪charge of their own.
  enable: True  # The global switch - Set to False to disable session support entirely

  cookies:  # What to do with cookies in each session type
             # session | Accept all cookies, but don't save anything to file
             # save    | Save any cookies set by websites to the cookie jar
             # update  | Discard any new cookies, but update any old ones that␣
↪already exist
             # discard | Discard all new cookies, don't save anything
    group: save
    global: discard

  never: []  # Domains that should never store their sessions
```

(continues on next page)

(continued from previous page)

```
              # These are checked first, before the rest
#   - 'facebook\.com'
#   - '.*\.facebook\.com'

  group:  # Groups of domains that should share session stores and never use the global store
            # These are checked after grouped domains - Any not matched after this stage use the global store
    example_group:
    - 'google\.com'
    - '.*\.google\.com'
    - 'youtube\.com'
    - '.*\.youtube\.com'
```

This rather large section is all about how we handle cookies. It's a little complicated, but provides quite a lot of flexibility. This part of the plugin uses regular expressions.

- **enable**: Set this to False to completely disable cookie support for this plugin. Note that plugins that add handlers are free to ignore this setting.
- **cookies**: What to do with cookies, depending on how they're categorised.
  - **Categories**:
    - **group**: Domains that you've grouped together, as shown below.
    - **global**: Any domains that you haven't included in a group.
  - **Settings**:
    - **session**: Accept all cookies and hold onto them until the plugin is reloaded. Never save them to file.
    - **save**: Save all cookies to file.
    - **update**: Discard any new cookies, but update any others that already exist.
    - **discard**: Discard all cookies, don't save anything.
- **never**: Regular expressions for domains that should never have cookies stored for them
- **group**: Groups of domains that should share their cookies, but keep them separate from all other domains.
  - **example_group**: Set this to a name that you'll remember, as it's used as the name of the cookie jar.
    - All domains in this list should be proper regular expressions to match.

If you don't understand regular expressions, we recommend the excellent Learn Regex the Hard Way by Zed. A. Shaw - Although it only goes up to exercise 16, it's a great place to start.

Please also note that when you're writing regular expressions in YAML, you should surround them with 'single quotes', so that YAML will not try to directly handle any regex escapes you use.

```
connection:
  max_read_size: 16384
```

This section is about advanced connection settings. Right now it only contains one setting - **max_read_size**. This setting is used when finding titles on pages - It will only read the specified number of bytes before attempting to find the title. This prevents excessively large pages or maliciously-crafted URLs from taking too long to parse or using up all of Ultros' memory.

We recommend you keep the default of **16384 bytes** (that's **16 KiB**). You may change it if required, but setting it too high may make the plugin sluggish, and setting it too low may miss some titles.

```
proxies:  # For proxying requests through http proxies
```

The last line in the file is the version of your configuration. Do not change this or you will likely break your configuration as we add newer versions.

Once you're all set up and ready to go, don't forget to open **config/settings.yml** and add **URLs** to your list of plugins!

2.1.2 Permissions and commands

Command: urls

- **Permission**: *urls.manage*
- **Usage**: *urls <setting> <value>*
  - **Setting**: *set <on/off>* - Enable or disable handling URLs for the current channel
  - **Setting**: *shortener <name>* - Set which URL shortener to use for the current channel
  - Run this command without arguments for help text and a list of shorteners

---

Command: shorten

- **Permission**: *urls.shorten*
- **Usage**: *shorten [url]*
  - You may specify a URL to shorten, or omit it to use the last URL that was sent to the channel
  - This will use the channel's configured shortener, or the default shortener when that isn't configured, the shortener is missing, or the command is used in a private message

---

Permission: *urls.trigger*

- This is used to determine whether a user is allowed to trigger the URLs plugin with a URL
  - This is a default permission, but you may use it to deny access to specific users, channels or protocols if needed

2.1.3 Known extension plugins

- URL-tools

CHAPTER 3

Indexes

- genindex
- modindex
- search