Ultros Documentation

Release 1.0.0

Gareth Coles, Sean Gordon

April 14, 2015

Contents

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

There’s not a lot here right now, but keep an eye out, this will be fleshed out very soon!

Are you a developer? Maybe you’re looking for the api

CHAPTER 1

Installation

Ultros requires the following for its basic functions:

- **Python 2.6 or 2.7 32-bit (NOT 64-bit as some libraries don't support it yet)**
  - **NOTE: Do not install Python from the website if you're on a Mac! See the Mac instructions below!**
  - Twisted
  - Zope.Interface
  - Kitchen
  - Yapsy 1.10.2 (Specifically, as they keep changing their API)
  - PyYAML
- **Optionally, the following are required for some core features**
  - **To support SSL in protocols:**
    - PyOpenSSL
  - **For the Mumble protocol:**
    - Google Protobuf
  - **For the URLs plugin:**
    - BeautifulSoup 4
    - Netaddr
- **On linux, you'll need the following**
  - libffi
  - libffi-dev
  - build-essential

1.1 Downloading Ultros

We highly recommend you use Git to download Ultros, as it allows you to easily keep it up to date, without worrying about patching or moving files around.

- **For Windows, you can install MSysGit here (Allow it to install to System32): https://code.google.com/p/msysgit/download**

- Where I mention a terminal below, you can use "Git Bash" from your start menu.

- On Linux, install Git from your package manager.

Next, open a terminal and run the following commands:

```
cd <path>  # Replace <path> with the directory you want to store Ultros in
git clone https://github.com/UltrosBot/Ultros.git
cd Ultros
```

You will now have a full copy of Ultros, just waiting to be set up!

To update Ultros in future, simply do the following:

```
cd <path>/Ultros  # Replace <path> with the directory from above
git pull --rebase
```

If you're thick-skulled, paranoid about wasting space or just don't like Git, you can download a zipball from the site, but you will have to keep it up-to-date manually.

Please see below for OS-specific installation instructions. For configuration, please see the *Configuration* page.

1.2 Quick-start

If you're going to use one of the start scripts, you'll need to install both **pip** and **virtualenv**. Install python, and then run the following commands to set both of those up:

```
cd <path>/Ultros  # Replace <path> with the directory you stored Ultros in
python packages.py
pip install virtualenv
```

On linux, you'll also want to run the following:

```
chmod a+x start.sh
```

To start Ultros, run **start.bat** on Windows, and **start.sh** on everything else. Always use CTRL+C to kill Ultros gracefully.

On linux, you may need to install libffi-dev for the installer to work. On Debian-based systems (such as Ubuntu), you can do something like this:

```
apt-get install libffi-dev
```

1.3 Without virtualenv

If you don't want to use a virtualenv (which you really should), you can set up and run Ultros in the following way.

On all systems, you should simply be able to open a terminal, **cd** to your copy of Ultros and run the package manager to install dependencies.

```
cd <path>/Ultros  # Replace <path> with the directory you stored Ultros in
python packages.py setup
```

The packages in this installer only support Python 2.7.

On linux, you may need to install **libffi-dev** for the installer to work. On Debian-based systems (such as Ubuntu), you can do something like this:

```
apt-get install libffi-dev
```

Presuming all is well, the following will start Ultros:

```
python run.py
```

If this doesn't work for you, then you can try the OS-specific methods below.

1.4 Windows

- Download and install Python 2.7.6: https://www.python.org/ftp/python/2.7.6/python-2.7.6.msi
- Add Python to your PATH: http://www.anthonydebarros.com/2011/10/15/setting-up-python-in-windows-7/
- Install pip by downloading and running this script (Just copy it into a file ending in .py and run it): https://raw.github.com/pypa/pip/master/contrib/get-pip.py
- Download and install Twisted: http://twistedmatrix.com/Releases/Twisted/13.2/Twisted-13.2.0.win32-py2.7.msi
- **If you require SSL support:**
  - Download and install OpenSSL for Windows: http://slproweb.com/download/Win32OpenSSL-1_0_1g.exe
  - Download and install PyOpenSSL: https://pypi.python.org/packages/2.7/p/pyOpenSSL/pyOpenSSL-0.13.1.win32-py2.7.exe
  - You'll see some errors in the next step if you don't do this, but Ultros should still work just fine for things that don't need SSL.

Now, open a command prompt, and run the following (replacing **<path>** with the *path to wherever you downloaded Ultros*):

```
cd <path>
pip install -r requirements.txt
```

Once this is done, you can start Ultros. On Windows, you should never do this by double-clicking run.py, it's always much safer to run it in a command prompt, so that you'll be able to shut Ultros down properly.

You may create a batch script using either of the below methods for starting Ultros.

To start Ultros normally:

```
@ECHO off
echo Starting Ultros..
python run.py
PAUSE
```

To start Ultros in debug mode:

```
@ECHO off
echo Starting Ultros in debug mode..
python run.py --debug
PAUSE
```

When you want to stop Ultros, instead of closing the window, **click on it and press CTRL+C to stop it gracefully**, and *then* close the window. Due to some annoying quirks in Windows, if you don't do this, then Ultros may not have time to save all its data. If you do this and lose some data, then it's not a bug, and we would appreciate if you would use the above method for stopping Ultros, instead of reporting it as one.

1.5 Linux

As the superior operating system for hosting practically anything, we highly recommend you use Linux to host your bot if you plan to keep it online for long periods of time. Linux also has a much easier setup, as follows.

- **Install Python from your package manager.**
  - Most package managers will install the latest version of Python 2, but some versions of Linux will install Python 3. Remember to check which version it installs!
- Install libffi and libffi-dev from your package manager.
- **If you need SSL, remember to install the standard OpenSSL package from your package manger, as well as a compiler (su**
  - You'll see some errors in the next step if you don't do this, but Ultros should still work just fine for things that don't need SSL.
- **Use pip to install all of the required modules.**
  - On some distros, you may also need to install python-pip

If you're on a recent version of Ubuntu or Debian, you should be able to do all of this in a method similar to the following, replacing <path> with the path to your copy of Ultros.

```
sudo apt-get install python python-dev openssl build-essential libffi libffi-dev
cd <path>
pip install -r requirements.txt
```

Naturally, you should replace the call to apt-get above with a call to your distro's package manager if you're not using Ubuntu or Debian.

Once you've done this, you can start Ultros using one of the following methods.

To start Ultros normally:

```
cd <path>
python run.py
```

To start Ultros in debug mode:

```
cd <path>
python run.py --debug
```

1.6 Mac OSX

- First of all, you should install Homebrew, if you haven't already: http://brew.sh/
- Open Terminal.app and run the following:

```
sudo xcode-select --install
brew install python
cd <path>  # Replace <path> with the directory you downloaded Ultros to
pip install -r requirements.txt
```

This could take a little while to complete - The first part may require you to update xcode as well.

To start Ultros normally:

```
cd <path>
python run.py
```

To start Ultros in debug mode:

```
cd <path>
python run.py --debug
```

Configuration

Ultros uses YAML for all its configuration files by default. While at least one of us feels that YAML is the best format for the job, it's worth noting that the exceptions produced by the YAML parser can be cryptic at best, so you should take note of the following guidelines.

- **You cannot use tabs in YAML files** - only spaces. It doesn't matter how many spaces you use, as long as you use the same number of spaces for all indentation in the same file.
- Take note of the layout of the example configuration files. For example, if you're adding elements to a list, you'll notice the example probably has "- " before the list entry, so you should probably do the same thing with your list entries.
- Everyone makes mistakes, and we're happy to help anyone having issues, but please take the time to check over your YAML files before asking us about bugs in the configuration handling. You can always use a linter to check your file over.

2.1 Main configuration

The main configuration deals with the plugins and protocols you want to use with your bot, as well as reconnection settings. As mentioned, Ultros supports connecting to multiple protocols, multiple instances of the same protocol, and even multiple connections to the same server.

The exerpts below are taken from the **config/settings.yml.example** file. If you need to redownload that file, you can find it here: `settings.yml.example`

```yaml
# Specify the names of the protocols to use for connections here.
protocols:
  - irc-esper
```

In this section, you simply list off the names of your protocols. These names are what you will refer to when you modify other configuration files. We recommend a descriptive naming system, such as **<protocol>-<network>**. The names you specify here must also exist in **config/protocols/<name>.yml**, which is where you configure the protocol itself. See the sections below for more information on that.

```yaml
plugins:
  - Auth  # Configurable permissions and authentication system that ships with Ultros
  - Bridge  # Configurable message relaying between protocols or within a protocol
  - Factoids  # Create and retrieve factoids
  - URLs  # Tools for working with URLs, URL titles and URL shorteners
```

In this section, you can list off the names of all of the plugins you want Ultros to load. The names here are specified per-plugin, and you can usually find the names of the plugin in their documentation. If you can't, then you can also edit the **plugins/<plugin>.plug** files; the name is also stored there.

The plugins that ship with Ultros are as follows:

- Auth
- Bridge
- Dialectizer
- Factoids
- URLs

```
reconnections: # Settings for reconnecting on connection failures.
               # Reconnection counters are not shared between protocols.
  delay: 10 # How long to wait between reconnection attempts, in seconds.
  attempts: 5 # How many times to attempt reconnecting
  on-drop: yes # Whether to reconnect if we lose connection
  on-failure: yes # Whether to reconnect if we fail to connect
  reset-on-success: yes # Whether to reset the counter if we successfully reconnect
```

In this section, you specify how Ultros should behave when it loses connection or flat-out fails to connect. The options are explained in the snippet above. You should note that the reconnection counters are not shared between protocols, which means that they will reconnect independently of each other.

```
# Simple metrics, for http://ultros.io/metrics

# Set this to "on" to enable the sending of some basic, anonymous metrics to the site.
#     This will assign this instance of the bot with a UUID if it doesn't have one already.

# Set this to "off" to alert the server that your UUID is not to have metrics collected, and
# disable metrics reporting.
#     If you have no UUID, this will prevent the bot from ever connecting to the server.

# Set this to "destroy" to have the server delete its records of your UUID, and delete it from
# this bot's instanec.
#     If you have no UUID, this will prevent the bot from connecting to the server. You don't have
#     to change this to "off" when the UUID is destroyed, for that reason.
metrics: on

# This allows you to disable the sending of exceptions to the Ultros metrics server, without disabli
# metrics entirely. Simply set it to "no" to do that.
send-exceptions: yes
```

This final option is for configuring Ultros' basic metrics. If you don't like metrics and you haven't yet started your bot, then set this to **off** and it will never contact the server.

- **on** - The bot will connect to the server, be assigned a UUID (if it doesn't have one already) and submit metrics every ten minutes. The only metrics we send to the server is a list of the types of protocols that are loaded, the currently enabled plugins, and which packages are installed.
- **off** - If the bot has already been assigned a UUID, it will contact the server once to clear its protocols, plugins and packages lists and to let it know that metrics will be disabled. After this is done, the bot won't contact the server again.
  - If the bot doesn't have a UUID, it won't contact the server.
- **destroy** - If you want your UUID to be unassigned and all of the bot's information to be removed from the server, you can set your metrics setting to **destroy**. This will tell the server to delete all its stored data on the current UUID, and will also delete the UUID that's stored locally.
  - If the bot doesn't have a UUID, it won't contact the server. Therefore, you don't need to change this to **off** when you've had the data removed.

If you want to look at the metrics, you can find them here: http://ultros.io/metrics

2.2 Protocols » All

As mentioned earlier, each protocol gets its own configuration file in **config/protocols/<name>.yml**. Each type of protocol has its own configuration, and they can be fairly different, but there is one section that is common across all protocols.

Example from an IRC protocol:

```
main:
  protocol-type: irc
  can-flood: no
```

Example from a Mumble protocol:

```
main:
  protocol-type: mumble
  can-flood: no
```

This section must be present in all protocol configs.

- **protocol-type** is the type of protocol you're defining here. As of this document, the types are **irc** and **mumble**.
- **can-flood** lets you turn off flood-limiting for your protocol. Some plugins may also use this to limit their output.
  - On IRC, you will almost always want this set to **no**, unless your bot has operator status on the network.

2.3 Protocols » IRC

The IRC protocol configuration is split up into several sections. You'll need the main section above, with the type set to **irc**.

```
network: # Network connection settings
  address: irc.esper.net # Address of the server
  port: 6697 # Port to connect to
  ssl: yes # Use SSL?
  password: ""
```

```
#  bindaddr: 127.0.0.1
```

The **network** section above is where you define your connection information.

- **address** - The address of the server to connect to.
- **port** - The port to connect to. The default port for IRC is **6667** (**6697** if you're using SSL).
- **ssl** - Whether to use SSL or not. This requires that you installed OpenSSL, in the installation instructions.
- **password** - Server password. You won't need this unless you're connecting through a bouncer or a special server.
- **bindaddr** - If you have more than one networking interface, the one to use for the connection. If you don't know what this is, then simply leave it commented out, it's not that important.

```
identity: # Settings relating to identification and authentication
  nick: Ultros # The bot's nickname
  authentication: None # This can be None, NickServ, NS-old Auth or Password.
```

```
                          # Use Auth if you're connecting to QuakeNet
                          # Use NS-old to omit the username when identifying with NickServ. If in doubt,
                          # Note that using NS-old means that you must sign on as the user you want to 
  auth_name: Ultros # The username to use for authentication (If applicable)
  auth_pass: password # The password to use for authentication (If applicable)
  auth_target: NickServ # Only used for NickServ auth, change this if the network has renamed their 
```

The **identity** section is where you set up the bot's identification and any authentication it needs.

- **nick** - The nickname to connect with.
- **authentication** - The type of authentication to use, if any. It can be one of the following:
  - **None** - No authentication.
  - **NickServ** - Use modern NickServ authentication, with a username and password.
  - **NS-old** - Use old-style NickServ authentication, with just a password. Your nick needs to be the username you're logging in with.
  - **Auth** - The authentication method used by QuakeNet.
  - **Password** - Standard password authentication.
- **auth_name** - The username to use for authentication, where applicable.
- **auth_pass** - The password to use for authentication, where applicable.
- **auth_target** - When you're using a form of NickServ auth above, this is the nickname NickServ is set to use. It's usually just **NickServ**.

```
channels: # Initial channels to join.
          # Remember, channel names must be surrounded in "quotes"
  - name: "#Ultros"
    key:
    kick_rejoin: no  # Set this to yes to have the bot rejoin automatically when kicked, if the glob
```

The **channels** section defines what channels to join when the bot has connected. It's a list of channels of the following format..

- **name** - The name of the channel to join. Be sure to wrap it in **"**quotes**", otherwise the leading **#** will comment it out.
- **key** - The channel key, if there is one. You can leave this blank if there isn't one.
- **kick_rejoin** - Whether to rejoin when kicked. This is ignored if you set **kick_rejoin** to **yes** below.

```
control_chars: "." # What messages must be prefixed with to count as a command.
```

The **control_chars** option defines what needs to be at the start of a message for it to be used as a command.

- It defaults to **"."**.
- This doesn't have to just be one character - Additionally, you can use **{NICK}** if you want the bot's nick to be there.

```
rate_limiting: # Limit the speed of sending messages
  enabled: yes
  line_delay: 0.1 # Delay (in seconds) between each line being sent
```

The **rate_limiting** section is for limiting the speed messages are sent at. This is important as most IRC networks will kill the bot if it sends messages too fast.

- **enabled** - Unless your bot has operator status on the network, you'll want this set to **yes**.

- **line_delay** - How long (in seconds) to wait between messages. Defaults to **0.1** seconds.

```
kick_rejoin: no
```

The **kick_rejoin** option here overrides the channel-specific ones if you set it to **yes**. Set it to **no** if you want to configure this for individual channels.

```
rejoin_delay: 2
```

The **rejoin_delay** option simply specifies how long to wait before rejoining a channel the bot was kicked from, in seconds.

```
perform:  # Raw lines to send to the server after we've identified but before we join channels
- "PRIVMSG ChanServ :INVITE #staff"
```

For **advanced users**, the **perform** section is a list of raw IRC messages to send to the server - after identifying, but before joining channels. For example, to have the bot be invited to an invite-only channel, you could do something like:

```
perform:
- "PRIVMSG ChanServ :INVITE #staff"
- "PRIVMSG ChanServ :INVITE #HERPDERP"
```

```
invite_join: no  # Whether to automatically join channels on invite
```

This section allows the bot to join channels automatically when someone on the network invites it. Note that turning this on will mean that anyone can have your bot join any channels they can use /invite in, so be aware of that.

2.4 Protocols » Mumble

Undocumented right now, please read the config file.

```
main:
  protocol-type: mumble
  can-flood: no

network:  # Network settings
  address: localhost  # The address to connect to
  port: 64738  # The port to connect on

identity:  # Identity settings. Who am I?
  username: Ultros  # Username to connect with
  password:   # Server password. You probably don't need this.
  tokens: []  # Auth tokens for locked-down channels.
  certificate: "config/protocols/mumblecert.p12" # Path to certificate file, relative to the base di

channel: # Channel to join on connect, specified either by name or id
  name:
  id:

control_chars: "." # What messages must be prefixed with to count as a command.
                   # This doesn't have to be just one character!
                   # You can also use {NICK} in place of the bot's current nick.
```

Plugins

This is the documentation for each known plugin (That is, those that are shipped with Ultros and those in the official contrib repository).

As of right now, only the **auth** plugin has been documented.

3.1 Core » Auth

Obtaining	Full name	Description
Included with Ultros	**Auth**	Adds a full permissions and authentication system. **This is required for most plugins to function.**

The Auth plugin provides two very important things.

- A user registration system with securely salted and hashed passwords
- A pretty granular permissions system for use with other plugins

The Auth plugin is **required** for most plugins to function. If you don't use it, you'll see lots of errors. The only reason we decided to make this functionality a plugin is so that other developers can create alternative systems for handling permissions and user authentication.

3.1.1 Setting up

The Auth plugin comes as standard with Ultros. Simply add **Auth** to your **settings.yml** file, under the **plugins** section.

```
plugins:
  - Auth
```

Once you've done that, copy **config/plugins/auth.yml.example** to **config/plugins/auth.yml** and edit to configure the plugin.

```
# Example settings of the basic password-based authentication plugin
# If you're not using the auth or permissions from this, you may as well just
#  disable this plugin.

use-superuser: yes # Allow users to be set as superadmin?
use-auth: yes # Use the authentication provider?
use-permissions: yes # Use the permissions provider?

```

```
# Permissions themselves are not defined in this file;
#  see data/plugins/auth/permissions.yml for that.
```

3.1.2 Permissions?

You may be asking yourself: “What are permissions and why do I need them?”, “Can’t it just use IRC/Mumble ranks?”, “Where are my pants?” - While some people feel like a full permissions system is a bit overkill (and we can understand why they might), we feel like it provides a much greater level of control over who can do what.

Still with us? Great, let’s get stuck in! Your permission file is named **permissions.yml** and can be found in **data/plugins/auth/**.

Here’s an example permissions file, which uses all the features of the permissions system.

```
groups:
    default:
        permissions:
            - auth.login
            - auth.logout
            - auth.register
            - auth.passwd
            - bridge.relay
            - urls.shorten
            - urls.title
            - drunkoctopus.drink
            - drunkoctopus.drunkenness
            - lastfm.nowplaying
            - lastfm.lastfmnick
            - money.main
            - geoip.command
            - urbandictionary.definition
            - wordnik.dict
            - wordnik.wotd
            - urls.shorten
            - items.give
            - items.get
            - wolfram.wolfram
            - money.main
            - factoids.get.*
            - aoshelper.playercount
            - aoshelper.aostoip
            - aoshelper.iptoaos
            - russianroulette.rroulette
            - jargon.jargon
            - 8ball.8ball
        protocols:
            irc-fraction:
                permissions:
                    - hb.hb
            mumble-fraction:
                permissions:
                    - hb.hb
    untrusted:
        inherit: default
        permissions:
            - ^aoshelper.*
            - ^factoids.*
            - ^auth.register
            - ^auth.passwd
            - ^drunkoctopus.*
    trusted:
        inherit: default
        permissions:
            - brainfuck.exec
            - minecraft.query
            - drunkoctopus.drink
            - drunkoctopus.drunkenness
            - factoids.add.channel
            - factoids.set.channel
            - factoids.delete.channel
    trusted-plus:
        inherit: trusted
        permissions:
            - factoids.add.protocol
            - factoids.set.protocol
            - factoids.delete.protocol
            - dialectizer.set
            - hb.hb
    admin:
        inherit: trusted-plus
        permissions:
            - web.admin
            - urls.manage
            - control.join
            - control.leave
            - control.say
            - control.action
            - factoids.set.*
            - factoids.add.*
            - factoids.delete.*
    super-admin:
        inherit: admin
        permissions:
            - control.raw
users:
    g:
        group: trusted-plus
        options:
            superadmin: false
        permissions:
            - control.*
    rakiru:
        group: super-admin
        options:
            superadmin: true
        permissions: []
```

At first glance, this seems like quite a lot, so let's break it down into two sections - **groups** and **users**.

Permissions groups

Permissions groups are a convenient way to give a set of permissions to multiple users at once. They also allow you to define specific permissions based on which protocol or channel a user is in when their permissions are checked.

The **default** group is what unregistered and newly registered users get checked against by default. Let me repeat that:

every new user will have the permissions in this group. Don't add any permissions you don't want **everyone** to have, to this group.

With that in mind, let's look at the **default group** that is generated when you first run Ultros with this plugin enabled.

```
groups:
    default:
        options: {}
        permissions:
            - auth.login
            - auth.logout
            - auth.register
            - auth.passwd
            - bridge.relay
            - factoids.get.*
            - urls.shorten
            - urls.title
```

The first thing you'll notice in this file is the start of the **groups** section. We define every permissions group in this section. Within this section, we have a section named **default** - this is the name of our group, and can be pretty much anything textual, but you must always have a group named **default**.

Within the **default** group, there are a couple more sections.

- An **options** section. This is currently unused, but plugins are allowed to use this for whatever they like.
- A **permissions** section. This is where we list off each permission we'd like to give to this group.
  - Permissions support **unix shell pattern matching**. The following matchers are as follows:
    - Wildcard: `"*"` - This matches any number of characters, and can match any character.
      - For example: `"factoids.get.*"`
    - Singular wildcard: `"?"` - This matches any character, but only one at a time.
      - For example: `"factoids.get.???"`
    - Character groups: `"[abc]"` - This matches any character that is within the brackets.
      - For example: `"factoids.get.[abcdefghijklmnopqrstuvwxyz_-]"`
    - Negative character groups: `"[!abc]"` - This matches any character that is **not** within the brackets.
      - For example: `"factoids.get.[!1234567890]"`
    - Character literals: `"[*]"` - For when you need to use special characters in your permissions.
      - For example: `"factoids.get.[*?[]]"`
  - Permissions also support **regular expressions**.
    - Definition: `"/pattern/flags"`
    - The following flags are supported:
      - **d**: Debug output
      - **i**: Ignore case
      - **l**: Depend on the locale for certain escapes
      - **m**: Multi-line start and end of string chars (**^** and **$**)
      - **s**: Make the dot (**.**) match newlines too
      - **u**: Depend on the unicode character table for certain escapes

      · **x**: "Verbose" or pretty regex support (with comments!)

- You can also specify **negative permissions nodes**, which will **deny** specific permissions already granted. They are always prefixed with the circumflex character, ^.
  - For example, if we specify `"factoids.get.*"` but we don't want users to see a factoid named **admin**, we could then give them `"^factoids.get.admin"`.
  - Negative permissions always take priority, even when talking about group inheritance. For that reason, you should be careful when designing your groups.

Groups also support a few more sections. Let's take a look at a more complicated default group.

```yaml
groups:
    default:
        permissions: [...] # Permissions from above
        protocols:
            irc-fraction:
                permissions:
                    - hb.hb
                sources:
                    "#fraction":
                        - brainfuck.exec
                    "#noheartbeat":
                        - ^hb.hb
            mumble-fraction:
                permissions:
                - hb.hb
```

The **protocols** section is where we begin to define protocols and channels for specific permissions. This lets us be a lot more granular when specifying permissions.

- Within the **protocols** section should be sections named after your protocols. In this example, we have a section for the **irc-fraction** and **mumble-fraction** protocols. Within this section, we can have..
  - A **permissions** section, which works the same as the one described above.
  - A **sources** section, where we describe each channel or other source. This is a list of permissions, as described above.

So, for example..

- If I'm in the default group and in **#fraction** on **irc-fraction**, then I get both the `hb.hb` and `brainfuck.exec` permissions.
- If I'm not in **#fraction** but I'm on **irc-fraction** or **mumble-fraction**, then I'll only be given `hb.hb`.
- If I'm in **#noheartbeat** on **irc-fraction**, then I won't get `hb.hb`, as it's been denied to users in that channel.
- If I'm not on either of those protocols, then I won't get either of the aforementioned permissions - even if I'm in **#fraction** on **irc-esper**.

Note: These permissions depend on the source they're checked against. For example, if I'm in **#noheartbeat** on **irc-fraction** and I attempt to use the **hb** command in **#hb**, then I'll be allowed to use it - but I won't if I use it in **#noheartbeat**.

This method of defining permissions gives us an awful lot of control over who gets to do what, and where. You should always take the time to go over this properly.

Warning: You should **never** give the `"*"` node to someone directly. Only use it to specify sub-permissions, never give users every permission available in this way. If you really need this, you should use the **superadmin** option detailed below.

Permissions users

So, you've made all your groups and assigned permissions to them, and everything's looking pretty spiffy - but wait, how do you give admin to a user? The answer is using the **users** section. Let's take a look at the default users section generated when you first run the bot with this plugin enabled.

```
users:
    superadmin:
        group: default
        options:
            superadmin: true
        permissions: []
```

In this section, we can list off our registered users, assign them groups and even individual permissions as detailed in the groups section. But wait, what's this **superadmin** user?

When you first ran the bot with this plugin enabled, you'll have seen a message that looks like this:

```
11 May 2014 - 14:12:12 |                  Auth |     INFO | Generating a default auth account and pass
11 May 2014 - 14:12:12 |                  Auth |     INFO | You will need to either use this to add pe
11 May 2014 - 14:12:12 |                  Auth |     INFO | Remember to delete this account when you'v
11 May 2014 - 14:12:12 |                  Auth |     INFO | ========================================
11 May 2014 - 14:12:12 |                  Auth |     INFO | Super admin username: superadmin
11 May 2014 - 14:12:12 |                  Auth |     INFO | Super admin password: 0rE8CpW37ihTA07xH5VU
11 May 2014 - 14:12:12 |                  Auth |     INFO | ========================================
```

The **superadmin** account, by default, has access to everything and is granted every permission available, regardless of groups and other settings. The first thing you should do is login to this account and change the randomly-generated password.

Warning: We do **not** recommend using a superadmin account, or giving any account the **superadmin** option. It exists for those rare cases that you may really need it, but do not use it as an excuse to not fill out your permissions file properly. If you do give someone else a user or group like this, you **will** mess up and give them a permission you **did not want them to have**.
You have been warned.

3.1.3 Other files

The auth plugin uses a couple other files for storage of various things.

- **blacklist.yml** contains a set of blacklisted passwords. This is updated if a user attempts to register an account in a public place, such as a channel.
- **passwords.yml** contains securely salted and hashed passwords, one for each user that has registered themselves. You cannot reset passwords by editing this file, so don't even try.
  - You should delete the entry for **superadmin** from this file when you've finished setting up your own account and permissions.

3.1.4 Commands and permissions

Commmand	Params	Permission	Description
.login	**Username Password**	auth.login	Allows you to login with your registered account.
.logout		auth.logout	Allows you to logout when you've logged in.
.passwd	**Old pass New pass**	auth.passwd	Lets you change your account password.
.register	**Username Password**	auth.register	Lets you register an account with the bot.

3.2 Core » Bridge

This page hasn't been written yet!

3.3 Core » Control

This page hasn't been written yet!

3.4 Core » Debug

Obtaining	Full name	Description
Included with Ultros	**Debug**	Injects a REPL, which is usable via the **debug** command.

The debug plugin provides a very useful and powerful tool for debugging your bot. It injects a REPL into the bot, on the main thread, which can be accessed via the **debug** command. Please note that this plugin is intended for use by developers; if you're just running a bot then you should not play with this.

3.4.1 Using

To run some code, use the **debug** or **dbg** command. Because code is run using **eval**, the bot is unable to relay output generated as a return value of whatever you're doing - it will be printed to the console instead. To get around this, we've provided an output function - `output(object)`. Pass it anything you want to be output, and it shall be so.

Note: Making the output function actually work properly was difficult. If you're running things in threads or otherwise asynchronously, then this plugin cannot guarantee that it will output to the correct location if you run another command while the last one is processing.

Code is run in the scope of the `reload()` function, which is a member of `DebugPlugin`. A useful member for working with the bot is `self.factory_manager`, which manages most of the bot.

Warning: Remember to be careful about who you give access to use this plugin. It can grant users *full control* over the machine the bot is running on, depending on the user the bot is running as.
In short: **Don't let anyone else use this plugin, and don't use it in production!**

3.4.2 Commands and permissions

Commmand	Params	Permission	Description
.debug .dbg	**Code**	debug.debug	Run some code.

3.5 Core » Dialectizer

This page hasn't been written yet!

3.6 Core » Factoids

This page hasn't been written yet!

3.7 Core » URLs

This page hasn't been written yet!

3.8 8-ball » 8-Ball

This page hasn't been written yet!

3.9 Anti-Mibbit » Anti-Mibbit

This page hasn't been written yet!

3.10 AoS-Helper » AoS-Helper

This page hasn't been written yet!

3.11 DrunkOctopus » DrunkOctopus

This page hasn't been written yet!

3.12 Feeds » Feeds

This page hasn't been written yet!

3.13 Heartbleed » Heartbleed

This page hasn't been written yet!

3.14 Jargon » Jargon

This page hasn't been written yet!

3.15 LastFM » LastFM

This page hasn't been written yet!

3.16 Minecraft » Minecraft

This page hasn't been written yet!

3.17 Money » Money

This page hasn't been written yet!

3.18 Old-plugins » Ass

This page hasn't been written yet!

3.19 Old-plugins » Brainfuck

This page hasn't been written yet!

3.20 Old-plugins » GeoIP

This page hasn't been written yet!

3.21 Old-plugins » Items

This page hasn't been written yet!

3.22 Old-plugins » Lastseen

This page hasn't been written yet!

3.23 Old-plugins » Memos

This page hasn't been written yet!

3.24 Old-plugins » Russian-roulette

This page hasn't been written yet!

3.25 Twilio » Twilio

This page hasn't been written yet!

3.26 URL-tools » URL-tools

This page hasn't been written yet!

3.27 UrbanDictionary » UrbanDictionary

This page hasn't been written yet!

3.28 Web » Web

This page hasn't been written yet!

3.29 Wolfram » Wolfram

This page hasn't been written yet!

3.30 Wordnik » Wordnik

This page hasn't been written yet!

3.31 xkcd » xkcd

This page hasn't been written yet!

Big ol' list of commands and permissions

This page is simply a big ol' list of commands, the plugin they're from and their permissions nodes. You can use this as a guide when configuring your instance of Ultros.

Remember, a **.** is the default command prefix, you can change it in your protocol-specific configuration file. Commands that don't start with a **.** here do not use the command prefix.

Additionally, where there are multiple commands, those that are in **bold** are the "real" commands (any others are aliases). Any params that are in **bold** are required for that command.

4.1 Core » Auth

Commmand	Params	Permission	Description
.login	**Username Password**	auth.login	Allows you to login with your registered account.
.logout		auth.logout	Allows you to logout when you've logged in.
.passwd	**Old pass New pass**	auth.passwd	Lets you change your account password.
.register	**Username Password**	auth.register	Lets you register an account with the bot.

4.2 Core » Bridge

This plugin has **no commands**.

4.3 Core » Control

Commmand	Params	Permission	Description
.action	**Target Message**	control.action	Send an action on the current protocol.
.func	**Function Data**	control.func	Lets you directly call a function on the current protocol.
.join	**Channel**	control.join	Tell the current protocol to join a channel.
.leave	**Channel**	control.leave	Tell the current protocol to leave a channel.
.raw	**Data**	control.raw	Send raw data to the current protocol.
.say	**Target Message**	control.say	Send a message on the current protocol.

4.4 Core » Dialectizer

Commmand	Params	Permission	Description
.dialectizer .dialectiser	**Name**	dialectizer.set	Allows you to set a dialectizer for the current channel.

4.5 Core » Factoids

Commmand	Params	Permission	Description
Standard commands			
.addfactoid	**Location Key**	factoids.add.[location]	Used to create a new factoid or add a line to an existing one.
.delfactoid	**Location Key**	factoids.delete.[location]	Used to delete an existing factoid.
.getfactoid	**Location Key Data**	factoids.get.[location]	Used to get the contents of a factoid.
.setfactoid	**Location Key Data**	factoids.set.[location]	Used to create a new factoid or replace an existing one.
Short commands for getting factoids			
??	**Key**	factoids.get.channel	Retrieve a factoid in the current channel.
??<	**Key**	factoids.get.channel	Retrieve a factoid privately.
??>	**Key Username**	factoids.get.channel	Retrieve a factoid and have it sent to another user privately.
Short commands for adding to factoids			
??+	**Key Data**	factoids.add.channel	Create or add to a factoid for the current channel.
@?+	**Key Data**	factoids.add.protocol	Create or add to a factoid for the current protocol.
!?+	**Key Data**	factoids.add.global	Create or add to a factoid in the global scope.
Short commands for deleting factoids			
??-	**Key**	factoids.delete.channel	Delete a factoid from the current channel.
@?-	**Key**	factoids.delete.protocol	Delete a factoid from the current protocol.
!?-	**Key**	factoids.delete.global	Delete a factoid from the global scope.
Short commands for setting factoids			
??~	**Key Data**	factoids.set.channel	Create or replace a factoid from the current channel.
@?~	**Key Data**	factoids.set.protocol	Create or replace a factoid for the current protocol.
!?~	**Key Data**	factoids.set.global	Create or replace a factoid from the global scope.
Other permissions			
factoids.get.web	Allow listing the factoids from the web interface, if installed.		

4.6 Core » URLs

Command	Params	Permission	Description
.urls	**Setting Value**	urls.manage	Change various URL handling settings.
.shorten	URL	urls.shorten	Shorten a specified URL, or the last URL sent to the current channel.
Other permissions			
N/A		urls.title	Allows a user to have their links parsed by the bot, and the title sent to the current channel.

4.7 DrunkOctopus » DrunkOctopus

Command	Params	Permission	Description
.drunkenness	Amount	drunkoctopus.drunkenness	Shows you the bot's drunkenness level, and allows you to change it.
.drink	**Drink**	drunkoctopus.drink	Give the bot a drink. Drinks are specified in the configuration file.

4.8 Feeds » Feeds

This plugin has **no commands**.

4.9 LastFM » LastFM

Commmand	Params	Permission	Description
.nowplaying .np	Username	lastfm.nowplaying	Shows what track you (or someone else) are currently playing.
.lastfmnick	Username	lastfm.lastfmnick	Set your Last.FM nickname, or check what you specified for it.

4.10 Minecraft » Minecraft

Command	Params	Permission	Description
.mcquery	**Address** Port	minecraft.query	Retrieves information on a Minecraft server. Port defaults to 25565.

4.11 Money » Money

Command	Params	Permission	Description
.money	**Amount** **Currencies**	money.main	Perform a currency conversion. You need to specify at least one currency code.

4.12 Old-plugins » Ass

This plugin has **no commands**.

4.13 Old-plugins » Brainfuck

>-12.4pt>-12.4pt>-12.4pt>-12.4pt 4.13. OLD-PLUGINS » BRAINFUCK

4.14 Old-plugins » GeoIP

	Commmand Description	Params	Permission
	.geoip Perform a geoip lookup on a web address or IP address.	**Address**	geoip.command

4.15 Old-plugins » Items

	Commmand Description	Params	Permission
.give Give an item to the bot.	.get Receive a random item that someone has given to the bot. **Item**	items.give	items.get

4.16 Old-plugins » Lastseen

Commmand	Params	Permission	Description
.seen	**Username**	seen.seen	Check when a user was last seen being active.

4.17 Old-plugins » Memos

This plugin has **no commands**. It's also **not written yet**.

4.18 Old-plugins » Russian-roulette

Commmand	Params	Permission	Description
.rroulette .roulette		russianroulette.rroulette	Play some Russian Roulette!

4.19 URL-tools » URL-tools

This plugin has **no commands**.

4.20 UrbanDictionary » UrbanDictionary

Commmand	Params	Permission	Description
.urbandictonary .ud	**Term**	urbandictionary.definition	Look up a term on Urban Dictionary.

4.21 Web » Web

Commmand	Params	Permission	Description
N/A		web.admin	Gives a user access to the admin interface.

4.22 Wolfram » Wolfram

Commmand	Params	Permission	Description
.wolfram	**Query**	wolfram.wolfram	Send a query to Wolfram|Alpha.

4.23 Wordnik » Wordnik

Commmand	Params	Permission	Description
.dict	**Word**	wordnik.dict	Check the definition of a word on Wiktionary.
.wotd		wordnik.wotd	Check the Wordnik word of the day.

4.24 xkcd » xkcd

Commmand	Params	Permission	Description
.xkcd	**Comic**	xkcd.xkcd	Search XKCD for a certain comic.

4.25 All permissions

Permission	Command	Aliases
auth.login	.login	
auth.logout	.logout	
auth.passwd	.passwd	
auth.register	.register	
brainfuck.exec	.bf	
control.action	.action	
control.func	.func	
control.join	.join	
control.leave	.leave	
control.raw	.raw	
control.say	.say	
dialectizer.set	.dialectizer	.dialectiser
drunkoctopus.drunkenness	.drunkenness	
drunkoctopus.drink	.drink	
factoids.add.[location]	.addfactoid	??+, @?+, !?+
factoids.delete.[location]	.delfactoid	??-. @?-, !?-
factoids.get.[location]	.getfactoid	??, ??<, ??>
factoids.get.web	For listing factoids on the web interface	
factoids.set.[location]	.setfactoid	??~, @?~, !?~
geoip.command	.geoip	

Continued on next page

Table 4.1 – continued from previous page

Permission	Command	Aliases
items.get	.get	
items.give	.give	
lastfm.nowplaying	.nowplaying	.np
lastfm.lastfmnick	.lastfmnick	
minecraft.query	.mcquery	
money.main	.money	
russianroulette.rroulette	.rroulette	.roulette
urbandictionary.definition	.urbandictionary	.ud
urls.manage	.urls	
urls.shorten	.shorten	
urls.title	**N/A**	
web.admin	For access to the admin area of the Web interface	
wolfram.wolfram	.wolfram	
wordnik.dict	.dict	
wordnik.wotd	.wotd	
xkcd.xkcd	.xkcd	

CHAPTER 5

Indexes

- *genindex*
- *modindex*
- *search*