



Axmud Guide

Version 1.1.050

31 Jul 2018

Copyright © 2014-18 A S Lewis. Permission is granted to copy, distribute an/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover-Texts.

<http://axmud.sourceforge.net/>

Contents

1 Introduction.....	5
2 Installing Axmud.....	6
2.1 Installation on MS Windows.....	6
2.2 Installation on Linux.....	8
2.3 Installation problems.....	9
3 Starting Axmud.....	10
3.1 Setup wizard.....	10
3.2 Connections window.....	10
4 Axmud basics.....	14
4.1 The main window.....	14
4.2 Client commands and world commands.....	15
4.3 Important client commands.....	17
4.4 Other kinds of instruction.....	18
4.5 Main window buttons.....	20
4.6 The object viewer window.....	21
4.7 Edit windows.....	22
4.8 Preference windows.....	23
4.9 The client preference window.....	23
4.10 The session preference window.....	24
4.11 The quick preference window.....	24
5 Interfaces.....	25
5.1 Profiles.....	26
5.2 Guild and race profiles.....	27
5.3 Custom profiles.....	28
5.4 The profile priority list.....	28
5.5 Interfaces in practice.....	29
5.6 Changing priorities.....	30
6 Patterns.....	31
6.1 Regular expressions.....	31
7 Interfaces.....	35
7.1 Triggers.....	35
7.2 Aliases.....	43
7.3 Macros.....	46
7.4 Timers.....	47
7.5 Hooks.....	49
7.6 Active and inactive interfaces.....	53
8 Axmud scripting.....	54
8.1 Types of script.....	54
9 Recordings and missions.....	55
9.1 Starting a recording.....	55
9.2 Creating a mission.....	55
9.3 Starting the mission.....	56
9.4 Adding breaks.....	56
9.5 Editing missions.....	57
9.6 Locator breaks.....	58
10 Axbasic.....	59
10.1 Axbasic script example: Hello world!.....	59
10.2 Axbasic script example: Test script.....	59
10.3 Axbasic script example: Hunt The Wumpus!.....	60

10.4	Testing scripts.....	60
10.5	Running scripts as a task.....	60
10.6	Axbasic help.....	62
10.7	Retrieving Axmud data.....	62
11	Tasks.....	63
11.1	Built-in tasks.....	63
11.2	Task dependencies.....	65
11.3	Task commands.....	65
11.4	The Status task.....	66
11.5	The Locator task.....	69
11.6	The Divert task.....	71
11.7	The Chat task.....	72
11.8	The Compass task.....	76
12	Initial and custom tasks.....	77
12.1	Initial tasks.....	77
12.2	Editing tasks.....	78
12.3	Custom tasks.....	79
13	Axmud plugins.....	81
13.1	The 'plugins' directory.....	81
13.2	Loading plugins.....	81
13.3	Plugin headers.....	82
13.4	Disabling plugins.....	83
13.5	Writing plugins.....	83
14	The desktop.....	86
14.1	Zonemaps.....	86
14.2	Workspaces.....	86
14.3	Zones.....	87
14.4	Standard zonemaps.....	87
14.5	Setting zonemaps.....	88
14.6	Layers.....	89
14.7	External windows.....	90
15	The Automapper window.....	91
15.1	Introduction.....	91
15.2	Automapper buttons.....	92
15.3	Creating regions.....	92
15.4	Creating rooms.....	93
15.5	Automapper modes.....	94
15.6	Selecting rooms.....	95
15.7	Selecting exits.....	96
15.8	Simple and complex exits.....	96
15.9	One-way exits.....	98
15.10	Connecting rooms.....	98
15.11	Moving rooms.....	99
15.12	Bent exits.....	100
15.13	Up and down.....	101
15.14	Non-primary directions.....	101
15.15	Labels.....	102
15.16	Exit ornaments.....	102
15.17	Assisted moves.....	104
15.18	Pathfinding.....	105

15.19 Room tags.....	106
15.20 Room guilds.....	106
15.21 Room flags.....	107
15.22 The painter.....	110
15.23 Room interiors.....	111
15.24 Updating maps.....	111
15.25 Backing up maps.....	113
15.26 Miscellaneous features.....	114
16 Axmud for visually-impaired users.....	122
16.1 Compatible speech engines.....	122
16.2 Connecting to a world.....	122
16.3 Playing the game.....	122
16.4 Enabling/disabling text-to-speech.....	123
16.5 Text-to-speech configurations.....	125
16.6 Configuring text-to-speech for tasks.....	127
17 Peek/Poke Operations.....	132
18 GNU Free Documentation License.....	157
0. PREAMBLE.....	157
1. APPLICABILITY AND DEFINITIONS.....	157
2. VERBATIM COPYING.....	158
3. COPYING IN QUANTITY.....	158
4. MODIFICATIONS.....	159
5. COMBINING DOCUMENTS.....	160
6. COLLECTIONS OF DOCUMENTS.....	161
7. AGGREGATION WITH INDEPENDENT WORKS.....	161
8. TRANSLATION.....	161
9. TERMINATION.....	161
10. FUTURE REVISIONS OF THIS LICENSE.....	162
11. RELICENSING.....	162

1 Introduction

Axmud is a modern client for MUDs (Multi-User Dungeons), MUCKs, MOOs, MUSEs, MUSHes, MUXs and other multiplayer text-based games (collectively called *worlds*).

Axmud's features include:

- Telnet, SSH and SLL connections
- ANSI, xterm, OSC and RGB colours
- Full support for all major MUD protocols, including MXP and GMCP (with partial Pueblo support)
- Class-based triggers, aliases, macros, timers and hooks
- A powerful graphical automapper
- 70 pre-configured worlds
- Multiple approaches to scripting
- Fully customisable from top to bottom, using the command line or the extensive GUI interface
- Native support for visually-impaired users

Axmud is still in the beta-testing phase and, as such, it lacks a proper manual. This guide introduces the most important features. Many of Axmud's features are fully documented; these help files can be read using the `‘;help’` command.

If you want only the bare essentials, click the blue question mark button in Axmud's 'main' window to read the quick help file.

When you install Axmud, you'll notice that there are two programmes, one of which uses settings that have been optimised for users with a visual impairment.

Visually-impaired users should skip most of this Guide, and read Section 16 instead.

2 Installing Axmud

Axmud is known to work on MS Windows and Linux. It might be possible to install Axmud on other systems, such as macOS and *BSD, but the authors have not been able to confirm this yet.

2.1 Installation on MS Windows

The easiest way to use Axmud on Windows is to download and run the Windows installer from the Axmud website.

The installer contains everything you need to run Axmud, including a copy of Strawberry Perl, a text-to-speech engine and all the required modules and libraries.

Users who already have Strawberry Perl and/or ActivePerl installed on their system can install Axmud manually using of the following methods.

2.1.1 Installation using Strawberry Perl

Axmud requires the 32-bit edition of Strawberry Perl, even on a 64-bit computer. If you have the 64-bit edition of Strawberry Perl installed on your system, either replace it with the 32-bit edition, or use the installer described above (which contains a copy of the portable 32-bit edition).

First, open a command prompt and get some modules from CPAN:

```
cpan Archive::Extract IO::Socket::INET6 IPC::Run Net::OpenSSH Regexp::IPv6
```

Then get the following module:

```
cpan File::ShareDir::Install
```

If the line above generates an error, try this line instead (it should be typed as single line):

```
cpanm --force --build-args SHELL=cmd.exe --install-args SHELL=cmd.exe  
File::ShareDir::Install
```

Now we need some modules from the Sisyphusion repo.

(If you reading this document some time in the distant future and find the repo is no longer available, you can either search around for a replacement, or you can use the installer.)

```
ppm set repository sisyphusion http://sisyphusion.tk/ppm  
ppm set save  
ppm install Cairo Glib Pango Gtk2 Gnome2::Canvas
```

Now install Axmud. It might possible to install it directly from CPAN but, if not, download the source code from the Axmud website and extract the *.tar.gz* file.

Open a command prompt and navigate into the extracted folder. From there, installation is standard.

```
perl Makefile.PL
gmake
gmake install
axmud.pl
```

2.1.2 Installation using ActivePerl

Axmud is known to work with both 32-bit and 64-bit editions of ActivePerl.

First, open a command prompt and get some modules from CPAN:

```
ppm install dmake
ppm install Archive::Extract File::ShareDir::Install IO::Socket::INET6
ppm install IPC::Run Net::OpenSSH Regexp::IPv6
```

Now we need some modules from the Sisypheusion repo.

(If you reading this document some time in the distant future and find the repo is no longer available, you can either search around for a replacement, or you can use the installer.)

```
ppm repo add http://www.sisypheusion.tk/ppm
ppm install Glib Gtk2 Cairo Pango Gnome2::Canvas
```

Now install Axmud. It might possible to install it directly from CPAN but, if not, download the source code from the Axmud website and extract the *.tar.gz* file.

Open a command prompt and navigate into the extracted folder. From there, installation is standard.

```
perl Makefile.PL
dmake
dmake install
axmud.pl
```

2.1.3 Creating your own MS Windows installer

The installer described above was created using a script you can find in:

```
.../nsis/axmud_install.nsi
```

The script contains instructions for building your own installer.

2.2 Installation on Linux

There are four methods of installation on Linux - install using the *.deb* package, install from CPAN, install using *git*, or install from source.

2.2.1 Install using the *.deb* package

The quickest method, if your system supports it, is to download the *.deb* package from the Axmud website.

On many modern systems, a *.deb* package will install itself if you double-click on it.

2.2.2 Install from CPAN

Axmud can be downloaded and installed directly from the Perl archive site, CPAN.

First, make sure you have the right dependencies. Open a terminal window and type:

```
sudo apt-get update
sudo apt-get install build-essential libcanberra-gtk-module libgnomecanvas2-dev
sudo apt-get install libwnck-dev
```

If you want to use sound effects and/or text-to-speech, you should also type:

```
sudo apt-get install libsox-fmt-mp3 timidity
```

Then you can download and install Axmud itself:

```
sudo cpan install Games::Axmud
```

When installation is complete, start Axmud by typing :

```
/usr/local/bin/axmud.pl
```

2.2.3 Install using *git*

Unstable (development) versions of Axmud are available from GitHub.

First, make sure you have the right dependencies. Open a terminal window and type:

```
sudo apt-get update
sudo apt-get install build-essential libcanberra-gtk-module
sudo apt-get install libgnomecanvas2-dev libwnck-dev
sudo cpan install Archive::Extract File::HomeDir File::ShareDir
sudo cpan install File::ShareDir::Install Gnome2::Canvas Gnome2::Wnck
sudo cpan install JSON Net::OpenSSH
```

If you want to use sound effects and/or text-to-speech, you should also type:

```
sudo apt-get install libsox-fmt-mp3 timidity
```


Then you can download and install Axmud itself:

```
git clone git://github.com/axcore/axmud
cd axmud
perl Makefile.PL
make
sudo make install
```

When installation is complete, start Axmud by typing:

```
axmud.pl
```

2.2.4 Install from source

Manual installation is quite simple. First, download the source code from the Axmud website (the most recent file ending *.tar.gz*).

Open a terminal window and navigate to the directory containing the downloaded file, for example:

```
cd Downloads
```

Then make sure you have the right dependencies:

```
sudo apt-get update
sudo apt-get install build-essential libcanberra-gtk-module
sudo apt-get install libgnomecanvas2-dev libwnck-dev
sudo cpan install Archive::Extract File::HomeDir File::ShareDir
sudo cpan install File::ShareDir::Install Gnome2::Canvas Gnome2::Wnck JSON
sudo cpan install Net::OpenSSH
```

If you want to use sound effects and/or text-to-speech, you should also type:

```
sudo apt-get install libsox-fmt-mp3 timidity
```

Then install Axmud itself:

```
perl Makefile.PL
make
sudo make install
```

When installation is complete, start Axmud by typing:

```
axmud.pl
```

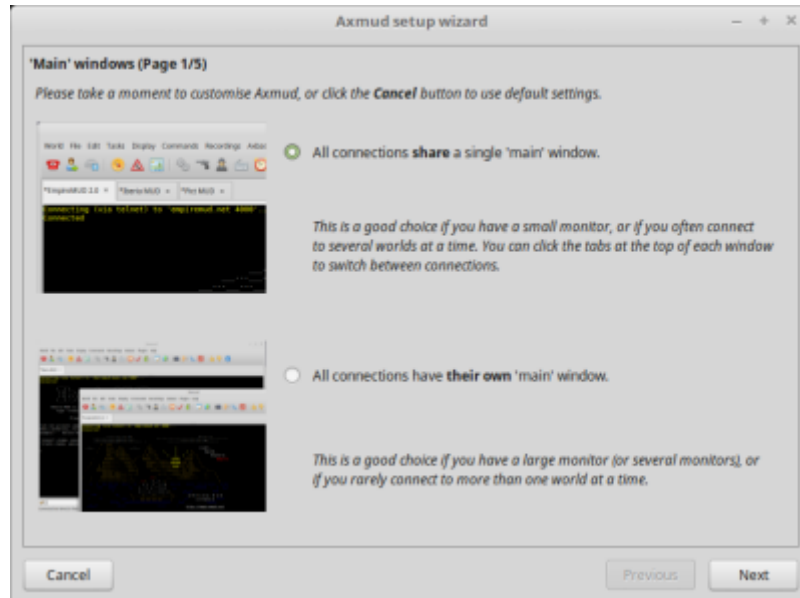
2.3 Installation problems

If you experience any problems getting Axmud to run, please contact us at the website above. We'll be falling over ourselves to help.

3 Starting Axmud

3.1 Setup wizard

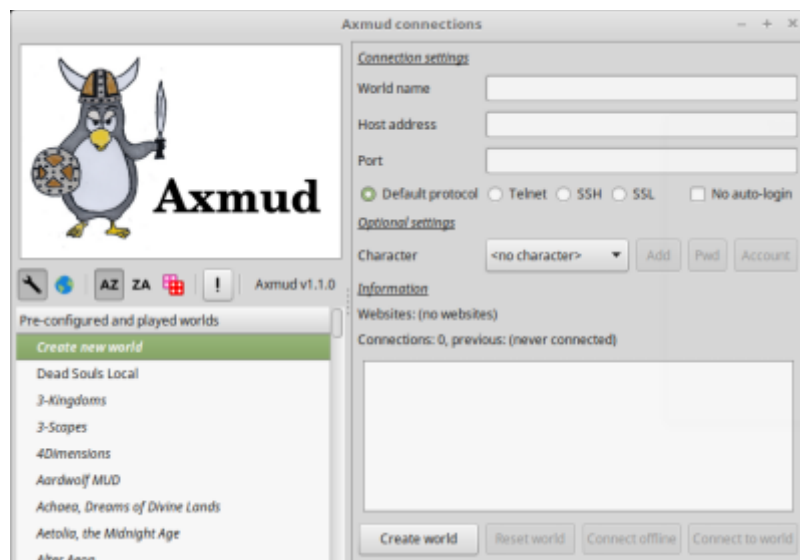
When you run Axmud for the first time you'll see the setup wizard.



This window is self-explanatory. If you're happy with standard settings, you can click the 'Cancel' button at any time.

3.2 Connections window

The Connections window appears whenever you start Axmud.



3.2.1 Pre-configured worlds

On the left you'll see a list of pre-configured worlds.

Pre-configured means that Axmud already knows a lot about the world, such as how to draw maps in the automapper and how to read the current character's health points.

If you're very lucky, your favourite world will be one of those listed. Scroll through the list and click on the world to select it.

Note that 'pre-configured' does not mean *this world requires no configuration at all*. The authors spent only a few hours, not hundreds of hours, at each world; there are invariably many features we didn't notice during that time.

3.2.2 Other worlds



Axmud also provides a much longer list of worlds – over six hundred of them, in the current version – from which you can choose. Click on the world button to see this list.

Axmud knows nothing about these worlds, besides how to connect to them. Features such as the automapper must be configured before they can be used.



Click on the spanner (wrench) button to return to the pre-configured world list.

AZ ZA



Click on these buttons to sort the list alphabetically or randomly.



If you see this orange button, Axmud has some error messages to display. Click the button to read them, if you want to.



Some worlds, which contain explicit sexual themes, are marked as being suitable only for adults. The authors make no guarantees that this information is complete and correct.

3.2.3 Connection Settings

You can select a world from the list, or you can click ‘Create new world’ to create a new world profile.

- The *world name* should begin with a letter, contain no spaces or punctuation, and be no longer than 16 characters
 - Some reserved words such as 'automapper' and 'dictionary' can't be used
 - You can create a temporary world profile by leaving the *world name* box empty. Temporary world profiles can't be saved, so they won't be available the next time you start Axmud
- The *host address* should be a DNS address like *deathmud.com* or an IP address like *101.102.103.104*
 - Both IPv4 and IPv6 addresses are recognised
- The *port* is a number between 0-65535, usually something 5000 or 6666
 - If you don't specify a port number, Axmud will use the default value of 23

A world's connection details are usually displayed prominently on the its own website, if it has one. You can also make use of referrer websites like mudstats.com or mudlistings.com.

3.2.4 Protocols

A *protocol* defines the ‘language’ that computers use to talk to each other.

Axmud supports all three protocols currently in common use. The oldest protocol, telnet, offers no encryption at all. SSH and SSL communications are encrypted.

Unfortunately, most worlds accept only telnet connections. Some worlds accept SSL connections and a very small number of them accept SSH. If you're using a pre-configured world, the *default protocol* is either SSH or SSL, if the world supports them, or telnet, if not.

Switching between one protocol and the other usually means specifying a different port number. It's not enough to simply click on the SSH or SSL buttons. Ask administrators at your favourite worlds if they support SSH/SSL and, if not, why not.

3.2.5 Optional settings

If you already have an account at the world, you can enter your username and password by clicking the ‘Add’ button.

If you're using a pre-configured world, Axmud will automatically login if you add your username and password here.

Even if you're not using a pre-configured world, it's still a good idea to enter these details now, because Axmud creates character profiles for each character you play.

Some worlds have separate account and character names. If so, you should enter both.

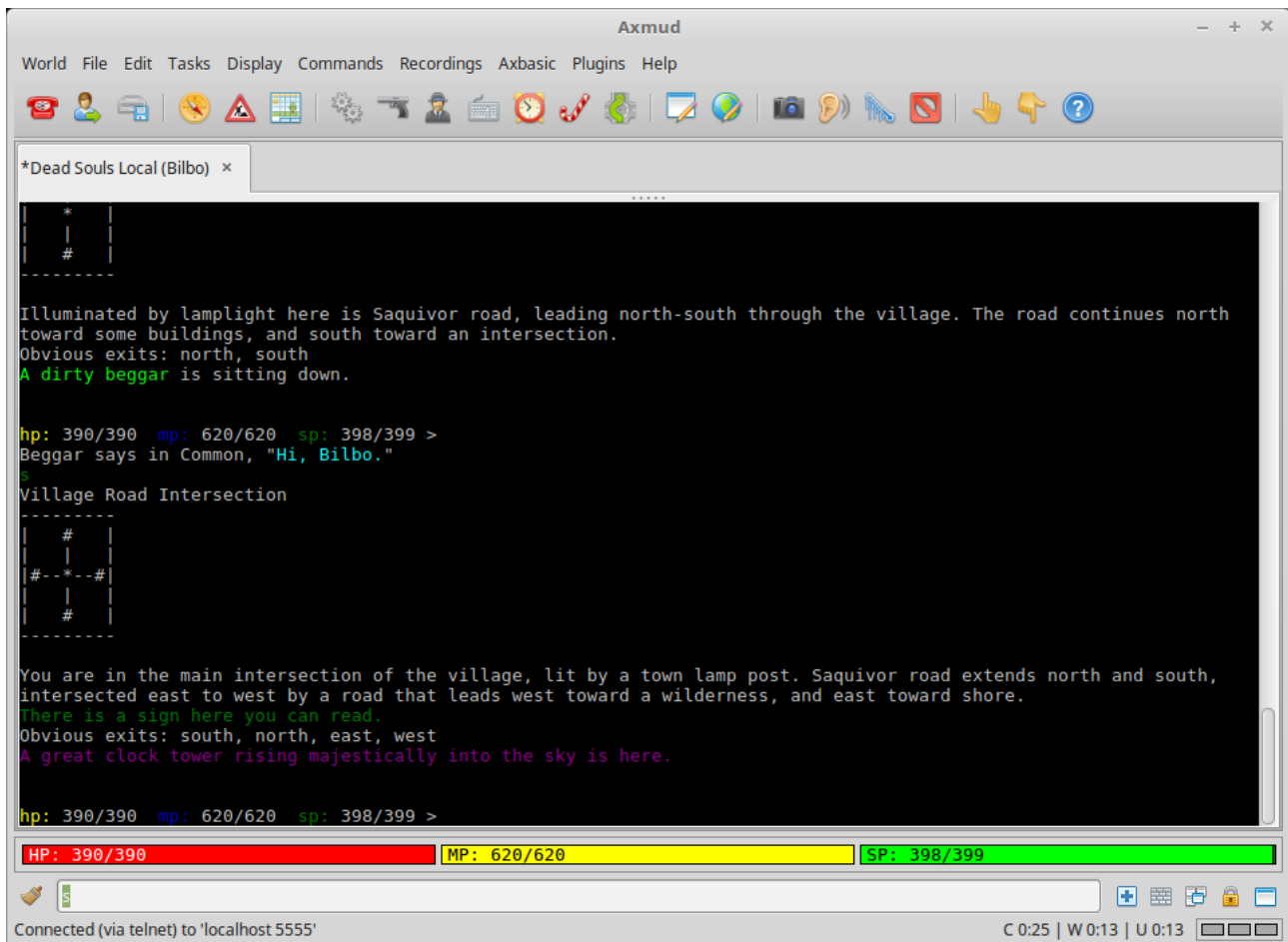
3.2.6 Connect

When you're ready, use one of the four buttons at the bottom of the window.

- *Apply changes* stores any changes you've made without connecting to the world (both world and character profiles are updated)
- *Reset world* resets any changes without connecting to the world
- *Connect to world* stores any changes you've made and connects to the world
- *Connect offline* stores any changes you've made, and opens a simulated connection to the world. This is useful for examining Axmud's stored data, browsing help files, admiring your maps and (in certain circumstances) testing scripts. In *connect offline* mode, absolutely no data is sent to and from the world

4 Axmud basics

4.1 The main window



- The window's title bar shows an asterisk (*) if there is any unsaved data at all
- Below the buttons is a tab label, which displays the current world and the current character
 - The label will change colour after a disconnection
 - The label's asterisk (*) tells you that there is unsaved data *for that world*
- You can connect to several different worlds simultaneously
 - Click the telephone button to re-open the Connections window; from there you can connect to additional worlds
 - Each connection has its own tab (called a 'session')
 - Axmud can optionally create separate main windows for each connection
 - The menus and buttons apply to the whichever session is the visible one

4.2 Client commands and world commands

Instructions can be entered in the command entry box at the bottom of the window. There are several kinds of instruction, some of which are disabled by default.

'World commands' are those which are sent to the world:

```
kill orcs
score
get torch
drop trousers
```

'Client commands' begin with a semicolon character (;) and are *not* sent to the world:

```
;about
;sound on
;openautomapper
;addtrigger -s <kills you> -p <viciously murders you> -rw 1
```

There are over six hundred client commands. One of the most important one is `';help'`, which shows a list of all available client commands:

```
;help
```

You can get more information about each client command and how to use it:

```
;help about
;help sound
;help openautomapper
;help addtrigger
```

You can also get a list of client commands related to a particular topic. For example, to get a list of client commands used with triggers:

```
;searchhelp trigger
```

Almost all client commands can be abbreviated. Here are the possible abbreviations for the `';openautomapper'` command:

```
;openautomapper
;openmap
;map
;oam
```

Abbreviations can even be used with `;'help'`. There are several ways to get help for the `;'addtrigger'` command, including:

```
;'help addtrigger
;'help addtrig
;'help atr
;'h addtrigger
;'h atr
```

Client commands in their abbreviated form are called 'user commands'.

User commands are customisable - you could replace `;'map'` with `;'zyxwv'`, if you really wanted to. For more information:

```
;'help addusercommand
```

The semicolon character (`;`) is also used to send multiple world commands. For example, to travel from one place to another, you might type

```
n;n;ne;ne;n;ne;e;e;ne;n;enter gate;in
```

Axmud will split this line into individual world commands, and send them one at a time. (This default behaviour can be changed, if necessary.)

There is one more type of world command. A 'forced world command' one which begins with two commas (`,,`):

```
,,kill orcs
```

Any command that starts with `,,` is *always* treated as a world command - so, if you need to send something to the world that looks like a client command, just put `,,` before it.

```
,,;help
```

Commands beginning with `,,` are never split before being sent to the world. This is useful for sending commands containing smileys, that would otherwise get split into two commands:

```
,,say hello ;) how are you doing?
```

Besides world commands and client commands, Axmud offers commands called 'echo', 'perl', 'script', 'multi', 'speedwalk' and 'bypass' commands.

These types of command start with a different special character; for example, speedwalk commands start with a full stop/period character (`.`) and script commands begin with an ampersand character (`&`).

All commands except world and client commands are turned off by default (unless you turned them on when the setup wizard window was open). See Section 4.4 for more information about them.

4.3 Important client commands

There are a few important client commands which need to be explained, before we continue.

```
;login
```

This command can be used if the automated login process fails, or if it didn't start in the first place.

Axmud provides a number of built-in tasks - useful scripts which often run in their own task window. Most built-in tasks won't start until the character is marked as 'logged in'. Use this command to tell them it's safe to start.

```
;sound on  
;sound off
```

Some tasks are capable of playing sound effects. The '*sound*' command turns Axmud's sound effects on and off in all sessions.

```
;playsoundeffect snoring
```

If you don't want to wait for a task to play a sound effect, you can use this command to test it's working.

```
;speech on  
;speech off
```

This command turns Axmud's text-to-speech capabilities on and off in all sessions. If you're using settings customised for visually-impaired users, it should already be turned on.

```
;speak Hello world
```

Use the '*speak*' command to test that text-to-speech is working.

```
;quit
```

This command disconnects you from the world. All of the current world's data is saved; it is automatically loaded again the next time you connect to the world.

Data is also saved when the connection closes naturally - for example, when you send the 'quit' command to the world. (This default behaviour can be changed, if necessary.)

```
;qquit
```

This command (which starts with two letter Qs) terminates the session, but does *not* save its data.

```
;panic  
;boss
```

These commands terminate *every* session immediately and *do not save any data*.

```
;save
```

This command saves the world's data without terminating the session.

```
;save -a
```

This command saves data in *every* session, without terminating any of them.

```
;autosave on
```

It would be quite inconvenient to have to type '*save*' every time Axmud's internal data is modified. The '*autosave*' command turns on Axmud's auto-save feature, which saves files every five minutes by default.

4.4 Other kinds of instruction

As mentioned above, client commands are always available, but other kinds of instruction (such as speedwalk commands) are turned off by default, unless you turned them on when the setup wizard window was open.

4.4.1 Echo commands

Use this command to turn echo commands on or off:

```
;togglesigil -e
```

Echo commands begin with a double quote (") character. Everything after this character is displayed in the main window, and in the same colour used for system messages (which is yellow by default). For example:

```
"Hello world!
```

4.4.2 Perl commands

Use this command to turn Perl commands on or off:

```
;togglesigil -p
```

Perl commands begin with a forward slash (/) character. Everything after this character is executed as a mini-Perl programme. The programme's return value is then executed as an instruction.

Axmud users triggers whose behaviour can sometimes be unexpected. For this reason, and for your own security, Perl commands have limited functionality (for example, it's not possible for a wayward trigger to accidentally execute a Perl command that deletes everything on your hard drive).

For those of you familiar with the Perl 5 language, here's an example of a trivial Perl command that sends an arbitrary world command to the current world.

```
/my $string = "kill orcs"; return $string;
```

Echo commands and Perl commands aren't very useful when they are typed in the main window, but they are *very* useful when creating triggers (see Section 7.1).

If you want to run Perl scripts with full functionality, you can use plugins (see Section 13).

The forward slash character is used by some worlds for other purposes, so if you don't need Perl commands, you should probably leave them turned off.

4.4.3 Script commands

Axmud comes with its own scripting language, Axbasic (see Section 10). Script commands are a convenient of starting an Axbasic script.

Use this command to turn script commands on or off:

```
;togglesigil -s
```

Script commands begin with an ampersand (&) character. The character should be followed by the name of an Axbasic script. For example, these two instructions both run the *hello.bas* script:

```
;runscript hello  
&hello
```

4.4.4 Multi commands

Use this command to turn multi commands on or off:

```
;togglesigil -m
```

Multi commands begin with a colon (:) character. Everything after this character is executed as a forced world command in multiple sessions, for example:

```
:shout I'm going for lunch
```

By default, multi commands are applied to all sessions, but they can optionally be applied only in sessions with the same world profile.

4.4.5 Speedwalk commands

Use this command to turn speedwalk commands on or off:

```
;togglesigil -w
```

Speedwalk commands begin with a full stop/period (.) character. Everything after this character is interpreted as a set of movement commands to send to the world. For example, these two instructions have the same effect:

```
north;north;north;west;south;south  
.3nw2s
```

Axmud speedwalk commands are actually quite powerful. The syntax, which is a hybrid of those used by other major MUD clients, goes far beyond merely sending "north" and "south".

For a full description of speedwalk syntax, see the help for the `';speedwalk'` command.

4.4.6 Bypass commands

Use this command to turn bypass commands on or off:

```
;togglesigil -b
```

Bypass commands begin with a greater than (>) character. Everything after this character is sent to the world immediately.

This can be useful if you've placed a limit on how many world commands Axmud can send at a time. This is called 'slowwalking'.

Slowwalking can be configured with the `';slowwalk'` command. It applies a maximum number of instructions that can be executed at a time. Any excess instructions are stored temporarily until they can be sent; a bypass command literally bypasses that queue.

4.4.7 Instruction equivalents

We've mentioned that speedwalk commands can be turned on or off, but actually it's only the initial character which is turned on or off. Speedwalking is always available using the equivalent client command, `';speedwalk'`.

```
.3nw2s  
;speedwalk 3nw2s
```

The two commands above are equivalent, but the first one will only work when speedwalk commands are turned on.

This applies to other kinds of instruction, too. See the help for `';echo'`, `';perl'`, `';multi'` and `';bypass'`.

4.5 Main window buttons

The main window contains a number of buttons for common tasks. You can see what each button does by letting your mouse hover above it.

Here's a summary of the most important buttons.



Opens another Connections window, from which you can connect to another world



Opens the automapper window



Opens a window which displays the quick help file, README



Stops the window from scrolling when text is received from the world



Splits the window into two portions

4.6 The object viewer window

Axmud organises its internal data into collections of data called 'objects'.

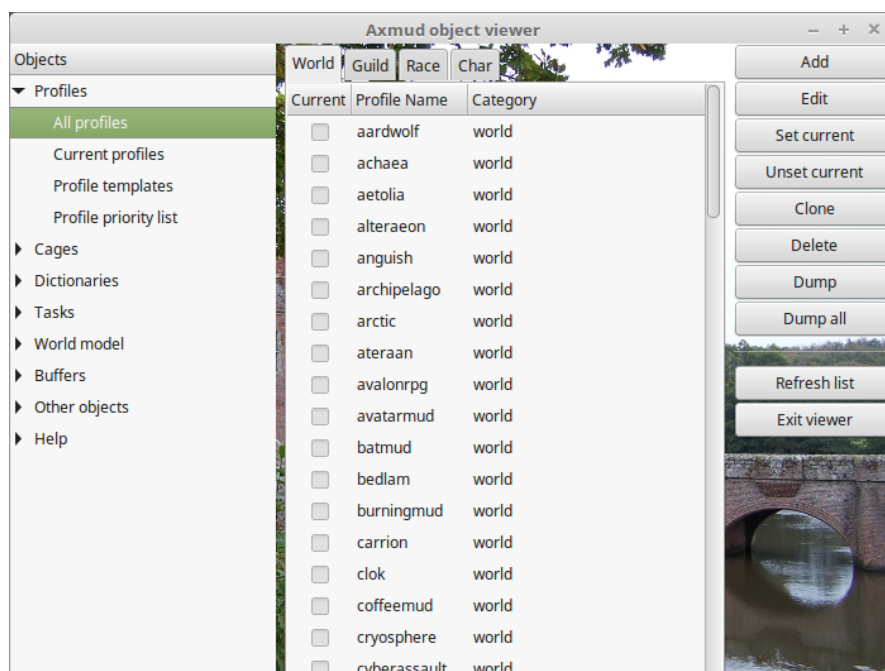
We've already seen two kinds of object - the world profile, which stores data for a particular world, and the character profile, which stores data for a character in that world.

There are many other kinds of object. Axmud lists many of them in its object viewer window.



To open the object viewer window, click on the button, select 'Display > Open object viewer' in the menu or use the client command:

```
;openobjectviewer  
;oov
```

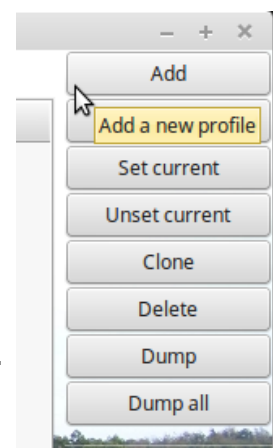


The different types of object are listed on the left. There are far too many to fit inside the window, so they have been divided into groups.

- Double-click on 'Profiles', and then click on 'All profiles'
- There is already a world profile for every pre-configured world. The current world is marked with an X
- If you click on the 'Char' tab, you can see this world's character profiles. The current character is marked with an X

On the right is a strip of buttons; each one is the equivalent of a client command.

- For a slightly longer explanation for each button, let your mouse hover above the button



4.7 Edit windows

Each collection of data - each object - can be viewed and edited using an 'edit window'.

Edit windows can be opened from the object viewer window, or from the main window's menu and buttons, or by using a client command. For example, to edit the 'dslocal' world profile:

```
;editworld dslocal  
;ew dslocal
```

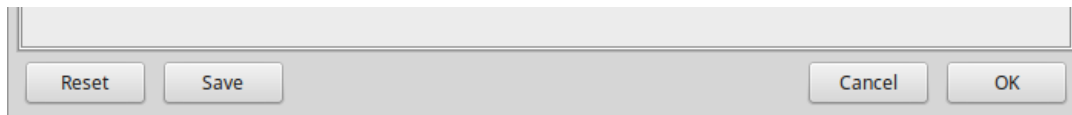
If you don't specify a world profile, the current world profile is edited.

```
;editworld
```

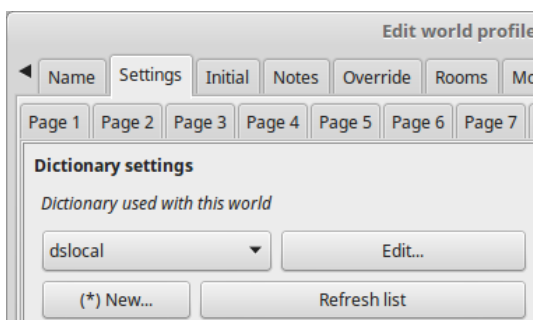
For world profiles, there are more tabs than can be arranged in the single window. Near the top-left and top-right corners of the window you'll see some kind of arrow; click the arrows to scroll through the additional tabs.



Changes you make in this window are not applied until you use one of the buttons at the bottom.



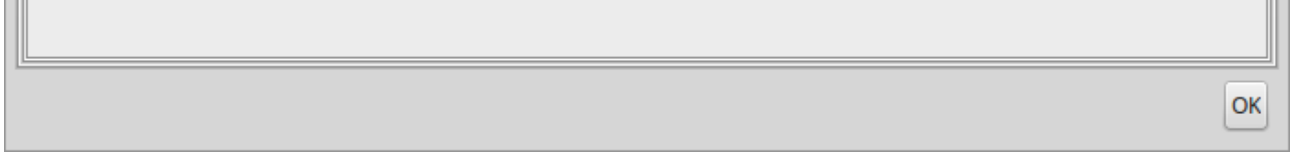
- The 'OK' button applies any changes you've made and closes the window
- The 'Save' button applies any changes you've made without closing the window
- The 'Cancel' button cancels any changes and closes the window
- The 'Reset' button cancels any changes without closing the window



For technical reasons, some changes have to be applied immediately. Buttons marked with an asterisk - such as the 'New' button pictured to the left - apply a change immediately. Using the main 'Cancel' or 'Reset' buttons won't reverse these changes.

4.8 Preference windows

'Preference windows' are a special kind of edit window in which *all* changes are applied immediately. For that reason, the buttons don't use asterisks. Instead, there is a single 'OK' button at the bottom of the window which closes it.



There are two preference windows you'll use often - one for settings that apply to all sessions, and another which applies to a single session.

4.9 The client preference window

The client preference window can be opened from the main window menu (click 'Edit > Axmud preferences...'), or by using a client command:

```
;editclient  
;edc
```

4.9.1 Adding a favourite world

The Connections window lists all of the world profiles you've created, as well as any pre-configured worlds. You can choose which worlds should appear at the top of this list.

- In the client preference window, click on 'Settings > Page 2'
- Add the name of a world profile, e.g. 'valhalla'
- Click the 'Set list' button
- When you next open the Connections window, Valhalla will appear at the top of the list

4.9.2 Enabling/disabling logfiles

Axmud can write a number of different logfiles simultaneously.

Logging is turned off by default. You can use the client preference window to turn it on and to specify which kinds of logfile to write.

- In the client preference window, click on 'Logs > Page 1'
- Click the first checkbox ('Enable logging in general')
- Now click on Page 3
- Use the combo (pull-down) box to select a type of logfile, and then click the 'Toggle' button to enable or disable it

Axmud stores all of its data in a directory (folder) called 'axmud-data'. This directory can be found somewhere in your usual home directory (on MS Windows, it's called something like 'Documents').

These are not the only logfiles Axmud can write; each world profile specifies its own logfiles.

- Open the world profile's edit window using `';editworld'`
- Click on Settings > Page 12
- Use the combo box and the 'toggle' button to enable/disable logfiles

4.9.3 Changing system message colours

Axmud displays its system messages in bright yellow by default. If you find this uncomfortable, the colour is easily changed.

- In the client preference window, click on 'Colours > Page 4'
- Find the line labelled 'System message colour'
- Use the combo (drop-down) box to choose a new colour
 - Bold colours are in CAPITAL LETTERS

4.10 The session preference window

The session preference window can be opened from the main window menu (click 'Edit > Session preferences...'), or by using a client command:

```
;editsession  
;eds
```

In the next Section we'll use this window to create and modify triggers, aliases and so on.

4.11 The quick preference window

A few of the most common configuration options have been grouped together in a single window.

```
;editquick  
;edq
```


5 Interfaces

Axmud offers powerful triggers, aliases, macros, timers and hooks to enhance your playing experience. Collectively, they are called 'interfaces'.

Triggers respond to text received from the world.

- For example, you can create a trigger to look out for lines beginning 'You kill'
- When lines matching this pattern are received, we say the trigger 'fires'
- When the trigger fires, it can send a world command in response
- A good world command to send would be *'get coins from corpse'*

Rewriter triggers are a special kind of trigger that modify the received text before it is displayed. (They don't send a world command, as normal triggers do.)

- For example, you can create a trigger to look out for lines containing the word 'damn'
- The trigger can modify the line to replace the word with '*****'

Splitter triggers are a special kind of trigger that split a line into two or more separate lines. Like rewriter triggers, splitter triggers don't send a world command.

- For example, *Imperian: Sundered Heavens* usually puts a room's verbose description and its exit list on two separate lines, but occasionally they appear on the same line
- The pre-configured world profile for *Imperian* moves the exit list onto a separate line, so that the Automapper doesn't get confused

Aliases respond to world commands, before they are sent to the world.

- You can create an alias to look out for the world command *'gcc'*
- This alias replaces the command with *'get coins from corpse'*, just before it is sent

Macros respond when you press a certain key (or a combination of keys).

- For example, you can create a macro that responds when you press the F1 key
- When you press the key, we say the macro 'fires'
- When the macro fires, it can send a world command in response
- You might create a macro to send *'get coins from corpse'* every time you press F1

Timers do something repeatedly, or wait for a period of time before doing something.

- For example, you can create a timer that fires every 60 seconds
- When the timer fires, it might send the world command 'inventory'
- You could also create a timer which waits ten minutes, fires once, and then disables itself

Hooks respond to certain hook events.

- An example of a hook event is the one called 'login'
- Every time an automated login is completed, we say the hook 'fires'
- The hook might send a world command in response, e.g. 'inventory'

In the next Section we'll talk about how to create new interfaces. Because Axmud interfaces are so powerful, we first need to discuss the way they are stored, and then we need to discuss how Axmud uses special patterns called 'regular expressions' (or regexes).

5.1 Profiles

Axmud interfaces (triggers, aliases, macros, timers and hooks) often 'belong' to a particular profile.

Most of the time, interfaces belong to a world profile, which means that the interfaces are available every time you connect to that world (and are *not* available when you connect to *any other* world.)

It's also possible for interfaces to belong to a character profile, which means that the interfaces are only available when you connect using that character.

The key points to understand are these:

1. If the current world and the current character both have a trigger called 'mytrigger', only one of these triggers can be 'active'
2. The other one is classed as 'inactive', which means it is ignored
3. All of this happens auto-magically. Your only concern is to make sure you have set the correct current character

5.1.1 Adding profiles

An easy way to add a new character profile is by using the '`;addchar`' command. (We've already discussed several other ways.)

```
;addchar bilbo  
;ach bilbo
```

If you don't want to type the password every time you use this character, you should specify it now:

```
;addchar bilbo mypassword
```

You can use the following command to list all character profiles for the current world. (For obvious reasons, it doesn't display passwords.)

```
;listchar  
;lch
```

5.1.2 Setting current profiles

Now, to make this character profile the current one, use the '*setchar*' command:

```
;setchar bilbo  
;sch bilbo
```

If a profile for 'Bilbo' doesn't already exist, it is created. You can also use this command to set (or change) the character's password:

```
;setchar bilbo newpass
```

5.1.3 Deleting profiles

The command to delete a character profile is, unsurprisingly, '*deletechar*'.

```
;deletechar bilbo  
;dch bilbo
```

If Bilbo is the current character, there will no longer be a current character. You'll have to set a new current character with '*setchar*'.

Similar commands can be used with world profiles:

```
;addworld deathmud  
;aw deathmud  
  
;setworld deathmud  
;sw deathmud  
  
;deleteworld deathmud  
;dw deathmud
```

There is one important restriction. Axmud insists that there is *always* a current world. You can't use '*deleteworld*' to delete the current world profile.

5.2 Guild and race profiles

Besides world and character profiles, Axmud provides 'guild' and 'race' profiles as standard.

Guild profiles divide characters into clubs: one guild for warriors, another for wizards, another for thieves, and so on. It doesn't matter if the world calls these divisions 'classes', or 'clans', or anything else - Axmud refers to them as 'guilds'.

At most worlds, the word 'race' is used in the Tolkienesque sense of a species. Race profiles commonly exist for humans, dwarves, trolls, elves and so on.

If you're using a pre-configured world, you probably already have a selection of guild and race profiles to choose from:

```
;listguild
;lg

;listrace
;lr
```

Otherwise, you'll have to use commands like '*setguild*' and '*setrace*'. You only need to set the current guild and race once - after that, every time you log in as Bilbo, the correct guild and race profiles will be set automatically.

5.3 Custom profiles

We've discussed world, guild, race and character profiles, but for some worlds it might be convenient to create other categories of profile. For example, at a MUD which divides players into factions, a character might belong to a guild, a race *and* a faction.

Axmud is able to create new kinds of profile. The blueprint for a new kind of profile is called a 'profile template', and profiles based on it are called 'custom profiles'. If you're interested to find out more (and are feeling brave), read the help for these client commands:

```
;help addtemplate
;help addscalarproperty
;help addlistproperty
;help addhashproperty
;help addcustomprofile
```

5.4 The profile priority list

Assuming you haven't created any custom profiles, your current profiles might now look something like this:

WORLD:	<i>deathmud</i>
CHARACTER:	<i>bilbo</i>
GUILD:	<i>thief</i>
RACE:	<i>halfling</i>

When we create interfaces (triggers, aliases, macros, timers and hooks), they usually 'belong' to a profile. Most of your interfaces will belong to the world profile.

Now, imagine that we have four different triggers, each with the same name ('mytrigger') and each belonging to one of these profiles. As we mentioned above, only one of them can be the 'active' trigger. The others are all treated as 'inactive' triggers.

How does Axmud choose which trigger is the active one? The answer is simple: it's the one which belongs to the character. Axmud specifies that the character profile *takes priority* over other categories of profile.

The profile priority list looks like this:

HIGHEST PRIORITY character > race > guild > world *LOWEST PRIORITY*

When Axmud is deciding which 'mytrigger' is the active one, it first looks at the current character profile. If it has a trigger called 'mytrigger', that's the active trigger.

If not, it looks at the current race profile. If *that* profile has a trigger called 'mytrigger', that's the active trigger. Otherwise it checks the current guild profile, and then the current world profile.

Now we get to the clever bit:

You can create hundreds of different profiles. You can change the current guild as often as you want or change the current character as often as you want (even in the middle of a session). Each of these many profiles can have their own trigger called 'mytrigger'.

And even though there are hundreds of triggers with the same name, only one of them is ever active.

Whenever you add, delete or modify a trigger, whenever you add, delete or modify a profile, whenever you set a new current profile, Axmud automatically marks just one trigger called 'mytrigger' as the active one.

No effort is required by you. It all happens auto-magically.

5.5 Interfaces in practice

In practice, you'll usually have only one trigger called 'mytrigger', and it will belong to the current world.

Suppose this trigger fires when Axmud receives a line containing the text 'You kill'. The trigger's response might be

get coins from corpse

This might be fine for most guilds, races and characters. We can think of this trigger as the default trigger. But, occasionally, we might want to add an exception to the general rule. For example, you might want the trigger to apply to all guilds *except* clerics. In that case, we create a second trigger with the same name, whose response is

bury corpse

When the current character is a cleric, the second trigger will be the active one, because guild profiles take priority over world profiles. For everyone else, there will be only one 'mytrigger' available - the one belonging to the world - and that will be the active one.

5.6 Changing priorities

If, for some reason, you'd prefer guilds to take priority over races, it's possible to change the priority order - but you shouldn't do it unless you really need to. See the help for the following client command:

```
;help setprofilepriority  
;h spp  
  
;help listprofilepriority  
;h lpp
```

6 Patterns

A 'regular expression' - or 'regex' - is just another way of saying 'pattern'. (Axmud uses these terms interchangeably.)

Many of times a second, Axmud will look at a line of text and ask, 'Does it match this pattern?'

Axmud uses regular expressions all the time, so it's important to know something about them.

There are a million and one tutorials on the internet, but here is another one anyway. It's as short as possible and all the example relate to what you might see in a MUD. (Skip to Section 6.2 if you already understand regular expressions.)

6.1 Regular expressions

A pattern can be as simple as the word 'troll'. The pattern 'troll' matches all of these lines:

```
You see a troll, two dwarves and an elf
You kill the troll
There are five trolls here
```

But it doesn't match any of these lines:

```
You see an orc, two dwarves and an elf
You kill the orc
You see the Troll
```

Patterns are usually case-sensitive. The last line above doesn't match the pattern because it contains 'Troll', and we're looking for 'troll'.

Patterns can be longer than a single word. The pattern 'kill the orc' matches the second line (but not the others).

6.1.1 Metacharacters

Sometimes we need to look for lines that *begin* with a certain pattern. The caret character (^) means that this pattern must appear at the *beginning* of the line.

The pattern '^troll' matches both of these lines:

```
troll on the floor, bleeding to death
trolls on the floor, bleeding to death
```

But it doesn't match either of these lines:

```
You see a troll
There are five trolls here
```

At other times we need to look for lines which *end* with a certain pattern. The dollar character (\$) means that this pattern must appear at the *end* of the line.

The pattern 'troll\$' matches both of these lines:

```
You see a troll
You kill the troll
```

Sometimes we will use both special characters together. The pattern '^You kill the troll\$' matches one line, and one line only:

```
You kill the troll
```

Needless to say, the ^ and \$ characters should appear only at the beginning/end of the pattern (and not somewhere in the middle).

6.1.2 Matching any text

Very often we'll need to match a line like this:

```
You are carrying 500 gold coins
```

In a pattern, we can use the full stop (period) character to mean 'any character'. For example, the pattern 'd.g' will match all of the following lines:

```
dig
dog
dug
dagger
degree
```

The character combination '.*' (a full stop/period followed by an asterisk) is very important. It means 'any text at all'.

So, the pattern 'You are carrying .* gold coins' matches all of the following lines:

```
You are carrying 100 gold coins
You are carrying 500 gold coins
You are carrying 1000000000 gold coins
You are carrying no gold coins
```

'.*' actually means 'any text, including no text at all'. So the same pattern will *also* match this line:

```
You are carrying gold coins.
```


6.1.3 Escape sequences

We can use a full stop (period) to mean 'any character', but sometimes you will want to be more specific.

The escape sequence '\w' - a forward slash followed by the lower-case letter w - means 'any letter, number or underline (underscore)'. So, the pattern 'b\wll' matches all of these lines:

```
I see a ball
I see a bell
I see a bill
```

But it won't match this line:

```
I see a b@ll
```

... because the '@' character is not a letter, a number or an underline (underscore).

The combination '\W' means the exact opposite - 'any character *except* a letter, number or underline (underscore)'. So, the pattern 'b\Wll' *does* match this line:

```
I see a b$ll
```

... but it *doesn't* match these lines:

```
I see a ball
I see a bell
I see a bill
```

One more important escape sequence is '\s', which means 'any space character, including tabs'. The opposite is '\S', which means 'any character *except* a space character or a tab'.

6.1.4 Quantifiers

Sometimes we'll need a pattern which can match any of these lines:

```
You kill the cat
You kill the caat
You kill the caaaaaaat
```

The character combination 'a+' means '1 or more letter "a" characters'. So, the pattern 'ca+t' matches all of lines above, but it doesn't match:

```
You kill the kitten
```

You can use the plus sign (+) after any character. For example, 'd' means 'a single number character', but 'd+' means 'one or more number characters'.

The pattern 'You have \d+ gold coins' matches all of the following lines:

```
You have 50 gold coins
You have 1000000 gold coins
```

...but it doesn't match this line:

```
You have no gold coins
```

You can also use a question mark (?) after any character. It means 'zero or one of these characters (but not more)'. And you've already seen that the asterisk (*) means 'zero, one or more of these characters'.

6.1.5 Backreferences

The pre-configured worlds have been set up to look for lines that these:

```
You have 100 gold coins
```

The patterns they use often look like this:

```
You have (.*?) gold coins
```

A pair of brackets (braces) means 'save everything in the middle for later'. In this case, we don't just want to recognise a line matching this pattern - we want to store the number of gold coins for later use.

The '(.)' combination is one example of a 'backreference'. Sometimes we'll need to use several backreferences on the same line.

```
You have (.*?) gold, (.*?) silver and (.*?) brass coins
```

Because we have three backreferences, three different values are stored for later. These backreferences are numbered 1, 2 and 3 (not 0, 1 and 2).

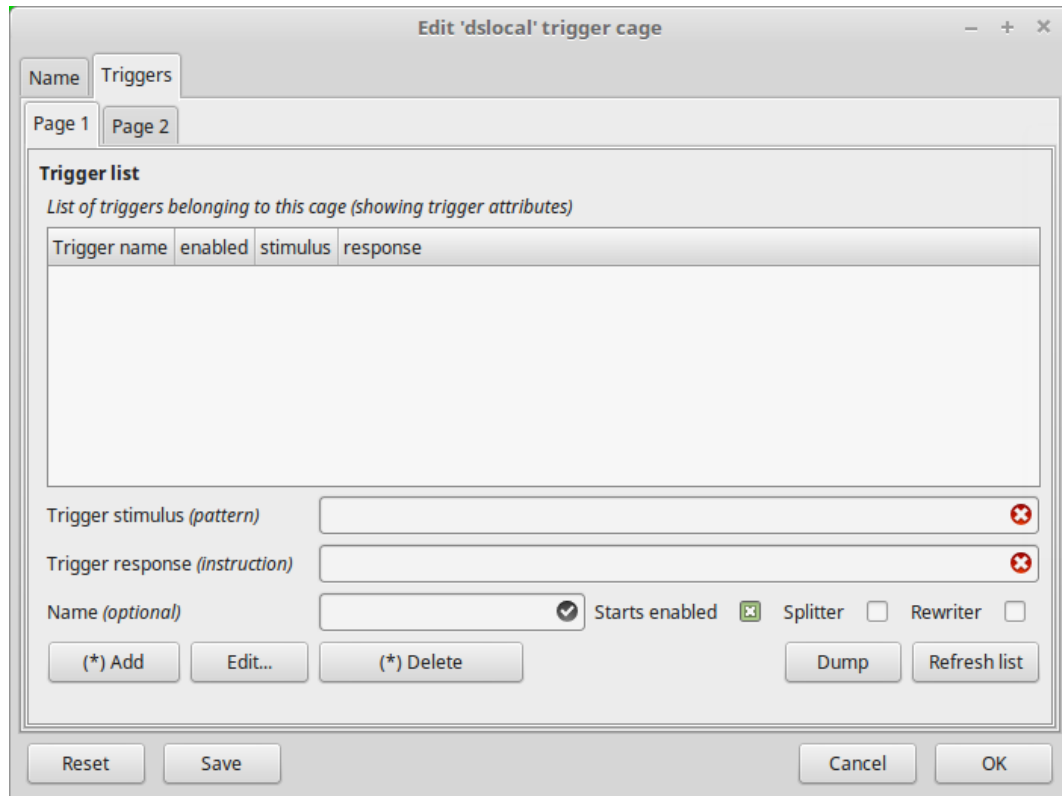
That's the end of the regular expression tutorial. Now we can go ahead and create some interfaces.

7 Interfaces

The quickest way to create a trigger, alias, macro, timer or hook is from the main window menu.

7.1 Triggers

We'll start with triggers. Click on 'Edit > World triggers'.



All interfaces have a 'stimulus' and a 'response'.

A trigger uses a pattern as a stimulus. When Axmud receives a line of text that matches the pattern, we say that the trigger 'fires'. After the trigger fires, it can send a world command in response.

- In the 'Trigger stimulus' box, enter the pattern 'You kill the orc'
- In the 'Trigger response' box, enter the world command 'get coins from corpse'
- In the 'Name' box, enter the name 'mytrigger'
 - A name will be created for you, if you don't enter one
- Click the 'Starts enabled' checkbox to de-select it
 - If the trigger starts disabled, we'll have a chance to finish editing it before Axmud starts checking it against received text
- Make sure the 'Splitter' and 'Rewriter' checkboxes are deselected, too
- Click the 'Add' button to create the trigger

7.1.1 Editing triggers

Triggers are highly customisable. Select 'mytrigger' by clicking on it, and then click the 'Edit' button.

The new edit window contains three tabs. The 'Name' tab allows you to change the trigger's stimulus and response, and also allows you to enable or disable the trigger.

The 'Attributes' tab allows you to customise the trigger - for example, to make matching lines disappear from view, or to disable the trigger automatically after the first match.

When Axmud is testing triggers against a line of text, it checks the triggers in the same order each time. Sometimes the order is important, so the 'Before / after' tab allows you to name the triggers that should be checked before or after this one.

7.1.2 Trigger attributes

Click on the 'Attributes' tab.

The screenshot shows a window titled "Edit trigger interface 'mytrigger'". It has three tabs: "Name", "Attributes" (which is selected), and "Before / after". Below the tabs are two sub-tabs: "Page 1" and "Page 2". The main area is titled "Trigger attributes" and contains a list of settings, each with a checkbox or a text input field. The settings are:

Attribute	Value
Splitter trigger	<input type="checkbox"/>
Split after matching pattern, not before	<input type="checkbox"/>
Split line multiple times, if multiple matches	<input type="checkbox"/>
Rewriter trigger	<input type="checkbox"/>
Rewrite every matching part of line	<input type="checkbox"/>
Ignore case	<input checked="" type="checkbox"/>
Only fire in session's default pane	<input checked="" type="checkbox"/>
Fire in named pane	<input type="text"/>
Require a prompt to fire	<input type="checkbox"/>
Require a login to fire	<input type="checkbox"/>
Omit (gag) from output	<input type="checkbox"/>
Omit (gag) from logfile	<input type="checkbox"/>
Keep checking triggers after a match	<input checked="" type="checkbox"/>
Temporary trigger	<input type="checkbox"/>

At the bottom of the window are four buttons: "Reset", "Save", "Cancel", and "OK".

We'll talk about rewriter triggers in Section 7.1.4 and splitter triggers in Section 7.1.6. First, let's examine the other attributes.

Ignore case

When the checkbox is selected, this trigger doesn't care about capital letters. So, the pattern 'You kill the orc' will match all of the following lines:

```
you kill the orc
You kill the orc
YOU KILL THE ORC
```

From your reading of Section 5 you'll remember that patterns are usually case-sensitive. De-select the button to make the pattern 'You kill the orc' match only the first line above.

Only fire in session's default pane

A pane is an area of an Axmud window containing (multiple lines of) text. Most windows, including the main window, contain only one pane by default. Text received from the world is displayed in the main window's single pane by default.

When the checkbox is selected, the trigger only applies to text received from the world and displayed in that default pane. When it's not selected, the trigger applies to text received from the world and displayed in any pane (including in other windows).

Fire in named pane

One way to create multiple panes is to use scripts, including plugins, tasks and Axbasic scripts. Tasks, in particular, often create their own task window(s) which can contain one or more panes.

Another way is to allow MUD protocols such as MXP and Pueblo to create multiple panes.

In such cases, the pane is often (but not always) given a name. If so, the trigger can be applied only to text received from the world and displayed in that pane.

Require a prompt to fire

Most lines received from the world end with an (invisible) 'newline' character. The character means 'the next piece of text should be displayed on a new line'.

A prompt is a line that doesn't end with a newline character. Typical prompts include

```
What is your name?
Enter your password
```

Axmud waits a short time (typically half a second) before deciding that a line of text is a definitely a prompt. After that, the trigger will fire if the checkbox is selected and if the pattern matches the line.

Require a login to fire

When you first connect to a world, you will often see graphics and other unusual text, which can cause your triggers to behave in unexpected ways. To avoid this problem, you can select this checkbox, after which the trigger won't fire until your character is marked as 'logged in'.

Omit (gag) from output

When the checkbox is selected, any line which matches the trigger's pattern is *not* displayed in the main window.

Omit (gag) from logfile

When the checkbox is selected, any line which matches the trigger's pattern is *not* written to any logfiles.

Keep checking triggers after a match

If the checkbox is not selected, when this trigger's pattern matches a line, Axmud will not check any other triggers. If the checkbox is selected, Axmud will continue checking the line against other triggers.

Temporary trigger

If the checkbox is selected, the trigger only fires once. After that, the active trigger becomes an inactive trigger - it will be available again, the next time you connect to the world, but it will not be available again during this session.

7.1.3 Trigger styles

Triggers are able to change the way matching lines are displayed. This is really useful if you want to change the colour of something like 'The orc drops its treasure', so that it stands out against the surrounding text.

Trigger styles are turned 'off' by default. When you turn them on, you need to decide whether the new style should apply to the whole line, or just the portion of the line that matches the trigger's pattern.

If the trigger's pattern contains backreferences (see Section 6.1.5), you can also apply a style just to that portion of text. For example, if you use the pattern 'You have (.*?) gold coins' which matches the line:

You have 100 gold coins

...you can draw attention to the '100' by clicking on 'Mode n', then selecting the number 1 (meaning, the first backreference in the pattern) in the box just below.

The trigger style can combine colours and other typographical features. 'Foreground colour' is the colour of the text itself. 'Underlay colour' is the colour of the background immediately below the text. (Bold colours are shown in CAPITAL LETTERS). If you don't specify a colour, the line's normal colours are not changed.

You can also apply *italics*, ~~striketrough~~, blinking text and underlines. If you apply the 'link' style, the text will be a clickable link which Axmud will try to open in a web browser.

7.1.4 Rewriter triggers

All interfaces have a stimulus and a response. Most triggers have a pattern (stimulus) and an instruction (response), but there is a special class of trigger called a 'rewriter trigger'.

For rewriter triggers, the text matching the pattern is *substituted* for the text of the response.

Rewriter triggers are often used to filter out coarse and injudicious language, so we'll use that as an example.

- From the main window menu, click 'Edit > World triggers'
- In the 'Trigger stimulus' box, enter a naughty word as a pattern
- In the 'Trigger response' box, enter the text '*****'
- In the 'Name' box, enter the name 'sweartrigger'
- Click the 'Rewriter' checkbox to select it
- Click the 'Add' button to create the trigger

Now, any received text which matches the pattern is modified, so that the scurrilous expletive is replaced by asterisks.

```
Gandalf says 'Oh, ****!'
```

7.1.5 Advanced rewriter triggers

Rewriter triggers can use backreferences, but the substituted text must be enclosed in double-quotes.

For example, suppose you want Axmud to convert a long string, containing lots of useless characters, into a shorter string containing only the important ones:

```
And the winner is ===== Gandalf =====
```

You could use a pattern like this:

```
And the winner is ===== (.*?) =====
```

...which would match the line above, as long as it contained exactly the right number of '=' characters. However, let's assume that the number is variable, and use this pattern instead:

```
And the winner is =+ (.*?) =+
```

(Section 6.1.4 explains that '+' means 'one or more of the previous characters', in this case, one or more '=' characters.)

When the trigger fires, anything inside a pair of brackets (...) is treated as a backreference. In this case, the first backreference contains the winner, Gandalf.

We can use the first backreference in the substituted text by specifying the variable \$1. However, the whole substituted text *must be enclosed in double-quotes*. This allows Axmud to treat the substituted text as something that might contain backreferences.

```
"And the winner is $1"
```

(If there is more than one backreference, you can also use '\$2', '\$3' and so on.)

So now we're ready to create our trigger:

- In the 'Trigger stimulus' box, enter the pattern 'And the winner is =+ (.*) =+'
- In the 'Trigger response' box, enter the text "'And the winner is \$1'"
 - Don't type the single quotes ', but *do* type the double quotes "
- In the 'Name' box, enter the name 'mytrigger'
- Click the 'Add' button to create the trigger
- Select the trigger in the list and click the 'Edit' button
- In the new edit window, click the 'Attributes' tab, and click the 'Rewriter trigger' checkbox

When the trigger fires, the text at the top of this Section will be converted to:

`And the winner is Gandalf`

7.1.6 Splitter triggers

At *Imperian: Sundered Heavens*, a room's verbose description and its exit list are usually on two separate lines, but occasionally they appear on the same line. This is confusing for the Locator task, on which the automapper relies, and can cause the automapper to lose its place.

The solution to the problem is to use a splitter trigger to split the line into two, when necessary.

(You don't need to read this Section if you want to play *Imperian*, because the pre-configured world profile already includes a suitable trigger. However, you should read this Section if you want to learn how to create your own splitter triggers.)

In Axmud terminology, the whole room description - including the title, verbose description, list of exits and the contents of the room - is called a 'room statement'. A typical room statement at *Imperian* looks like this:

`A quiet alcove.
This small alcove is a sanctuary of quiet reflection.
You see a single exit leading north.`

The room statement consists of three lines - a title, a verbose description and an exit list. The verbose description is actually received as a single line - even when it's very long - but, depending on the size of the main window, it might be displayed as several lines.

Alas, some rooms at *Imperian* have a room statement that looks like this:

`A neat storage room.
This small storage room is neatly stacked. You see a single exit
leading south.`

Here, the verbose description and exit list appear on the same line. Because the Locator task is expecting the exit list on a separate line, we need to split the second line into two, with one sentence on each line.

7.1.7 Simple splitter triggers

All triggers have a 'stimulus' and a 'response'. For splitter triggers, both of these are patterns. The simplest splitter triggers use the same pattern for both.

- From the main window menu, click 'Edit > World triggers'
- In the 'Trigger stimulus' box, enter 'You see a single exit leading'
 - Don't type the single quotes '
- In the 'Trigger response' box, enter the same pattern again
- In the 'Name' box, enter the name 'exittrigger'
- Click the 'Splitter' checkbox to select it
- Click the 'Add' button to create the trigger

Now, the next time you type 'look' at *Imperian*, you would see this:

A neat storage room.
This small storage room is neatly stacked.
You see a single exit leading south.

Splitter triggers usually split the line just before the portion that matches the 'stimulus' pattern. Sometimes it's more convenient to split a line just *after* a matching portion of text. Here's an example using the same room statement.

- From the same edit window, click on 'exittrigger' to select it, and then click the 'Edit...' button
- A new edit window appears, with the 'Name' tab already visible
- In the 'Stimulus' box, type 'This small storage room '
 - Don't type the single quotes '
 - Put a space character at the end of the pattern, so that the line is split just before the beginning of the next word
- In the 'Response' box, enter the same pattern
- Now click on 'Attributes > Page 1'
- Find the checkbox 'Split after matching pattern, not before' and click on it to select it
- Click the 'OK' button to close the window

The next time you 'look' at the room, you would see this:

A neat storage room.
This small storage room
is neatly stacked. You see a single exit leading south.

7.1.8 Advanced splitter triggers

The examples in the previous Section use identical patterns for both the stimulus and response. Sometimes it's useful to use two different patterns. Here's an example using the same room statement, in which the verbose description and exit list are initially on the same line.

A neat storage room.

This small storage room is neatly stacked. You see a single exit leading south.

When Axmud tests a splitter trigger against a line of text, it uses the following process:

- First, it checks the stimulus pattern *once*. If the stimulus pattern matches the line of text, Axmud moves onto the next step
 - For example, the pattern 'single exit' matches the second line
- Next, Axmud checks the response pattern *multiple times*. If the response pattern matches the line once, Axmud splits the line into two portions. If the pattern matches the line *twice*, Axmud would split the line into three portions.
 - Axmud will split the line into as many matching portions as necessary

We can use this to our advantage: the stimulus pattern tells Axmud *if* the line should be split, the response pattern tells us *exactly where* it should be split.

Let's modify the existing trigger so that it splits a line containing 'You see a single exit leading', but *only* if it follows another sentence on the same line.

- From the same edit window, click on 'exittrigger' to select it, and then click the 'Edit...' button
- A new edit window appears, with the 'Name' tab already visible
- In the 'Stimulus' box, type '\. This small storage room '
 - Don't type the single quotes '
 - The pattern needs to start with a full stop (period), and that should be followed by a space character. This is how we ensure that 'You see a single exit leading' follows another sentence on the same line.
 - Don't forget that a full stop (period) must be preceded by a \ character in patterns (see Section 6 if you don't know why)
- In the 'Response' box, enter the pattern 'You see a single exit leading'
- Now click on 'Attributes > Page 1'
- Find the checkbox 'Split after matching pattern, not before' and make sure it is *not* selected
- Click the 'OK' button to close the window

The next time you 'look' at the room, you would see this:

```
A neat storage room.  
This small storage room is neatly stacked.  
You see a single exit leading south.
```

This has been a trivial example; in fact, this more complex splitter trigger has exactly the same effect as the first one we created.

However, you can use more complex splitter triggers to split a line in situations that wouldn't be possible using a simpler version.

7.2 Aliases

Aliases act on world commands such as 'kill orc' or 'inventory'. They don't act on client commands (see Section 4.2) or other kinds of instructions such as speedwalk commands (Section 4.4).

Aliases are generally used to make commands shorter. For example, if you frequently have to type a command like this:

```
get coins from corpses
```

...you can use an alias to reduce it to a command like this:

```
gcc
```

You can also reduce *part* of a command. If you frequently have to type a command like this:

```
put amazing red crown on head
```

...you can use an alias to reduce it to a command like this:

```
put arc on head
```

7.2.1 Adding aliases

Like all interfaces, Axmud aliases have a stimulus and a response. The stimulus is a pattern. When you type a world command that matches the pattern, we say that the alias 'fires'.

The response is some text. When the alias fires, the response is *substituted* for part of the original command (or replaces all of it).

- From the main window menu, click 'Edit > World aliases'
- In the 'Alias stimulus' box, enter the pattern 'arc'
- In the 'Alias response' box, enter the text 'amazing red crown'
- In the 'Name' box, enter the name 'myalias'
 - A name will be created for you, if you don't enter one
- Click the 'Add' button to create the alias

You can now type the world command 'put arc on head'; it should appear in the main window as

```
put amazing red crown on head
```

7.2.2 Safer aliases

Aliases like these come with hidden dangers: the pattern 'arc' also matches this command:

```
enter archway
```

If you type 'enter archway', the alias will fire, and the *actual* command sent to the world will be:

```
enter amazing red crownhway
```

...which is probably not what you intended!

For this reason, most of your aliases will use patterns which begin and end with the metacharacters ^ and \$ (see Section 6.1.1 if you don't understand why).

Now, let's create a second alias.

- In the 'Alias stimulus' box, enter the pattern 'gcc'
- Click the button 'Recommended: add ^...\$'
- In the 'Alias response' box, enter the text 'get coins from corpses'
- In the 'Name' box, enter the name 'myalias2'
- Click the 'Add' button to create the alias

This alias will only fire when you type the exact command 'gcc' - it won't fire for any command which *contains* the string 'gcc'.

7.2.3 Editing aliases

Like triggers, aliases are also customisable. Select 'myalias' by clicking on it, and then click the 'Edit' button.

The new edit window contains three tabs. The 'Name' tab allows you to change the stimulus and response, and also allows you to enable or disable the alias.

The 'Attributes' tab allows you to customise the alias - for example, to disable the alias automatically after the first match.

When Axmud is testing aliases against a world command, it checks the aliases in the same order every time. Sometimes the order is important, so the 'Before / after' tab allows you to name the aliases that should be checked before or after this one.

7.2.4 Alias attributes

Click on the 'Attributes' tab. There are three attributes which can be customised.

Ignore case

When the checkbox is selected, this alias doesn't care about capital letters. In the examples above, 'amazing red crown' will be substituted for all of the following world commands:

arc
Arc
ARC

From your reading of Section 5 you'll remember that patterns are usually case-sensitive. De-select the button to make the pattern 'arc' match only the first command above.

(Although it's unlikely that you would want to distinguish between upper-case and lower-case world commands, the capability is available, if you should ever need it.)

Keep checking aliases after a match

If the checkbox is not selected, when this alias's pattern matches a world command, Axmud will not check any other aliases. If the checkbox is selected, Axmud will continue checking the command against other aliases.

Temporary alias

If the checkbox is selected, the alias only fires once. After that, the active alias becomes an inactive alias - it will be available again, the next time you connect to the world, but it will not be available again during this session.

7.2.5 Advanced aliases

Aliases can use backreferences, but the replacement string must be enclosed within double-quotes.

For example, suppose you want Axmud to convert a world command like 'give 50 gc to orc' to 'give 50 gold coins to orc'. This is how to do it, so that the alias will work with any number of coins being given to any creature:

- In the 'Alias stimulus' box, enter the pattern '^give (.*?) gc to (.*?)\$'
- In the 'Alias response' box, enter the text "'give \$1 gold coins to \$2'"
 - Don't type the single quotes ', but *do* type the double quotes ".
- In the 'Name' box, enter the name 'myalias3'
- Click the 'Add' button to create the alias

You can now test the alias by typing the world command 'give 50 gc to orc'.

7.3 Macros

Macros wait for you to press a key like 'F1', and then do something in response.

Like all interfaces, Axmud macros have a stimulus and a response. The stimulus is a keycode that represents one of the keys on your keyboard. When you press the key, we say that the macro 'fires'.

The response can be any kind of instruction. It's usually a world command like 'get coins from corpse', but it could be a client command like `';open automapper'` or any other kind of instruction (see Section 4.4).

7.3.1 Axmud keycodes

A key part of Axmud's design is *platform independence*. If you install Axmud on one computer with Linux and on another computer with MS Windows, the data you save on one can be transferred to the without problems.

For this reason, Axmud uses keycodes - a short string corresponding to every key on the keyboard. The keycode for each key is the same, regardless of which operating system you're using.

Axmud's keycodes - always in lower-case - are:

```
shift alt alt_gr ctrl num_lock
escape pause break insert delete return backspace space
home page_up page_down end
up down left right
f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12
grave tilde exclam at number_sign dollar percent ascii_circum
  ampersand asterisk paren_left parent_right
plus minus equal underline
bracket_left bracket_right brace_left brace_right colon
  semicolon apostrophe quote slash backslash pipe comma
  full_stop less_than greater_than question_mark
```

There are also a set of keycodes for the keypad (the keys, normally on the right side of the keyboard, which are laid out like a calculator):

```
kp_0 kp_1 kp_2 kp_3 kp_4 kp_5 kp_6 kp_7 kp_8 kp_9
kp_up kp_down kp_left kp_right kp_page_up kp_page_down
  kp_home kp_end kp_insert kp_delete kp_full_stop
```

Keycodes can be combined when two or more keys are pressed together. For example, SHIFT + F1 corresponds to the keycode string 'shift f1'.

7.3.2 Adding macros

- From the main window menu, click 'Edit > World macros'
- In the 'Macro stimulus' box, enter the keycode 'f1'
- In the 'Macro response' box, enter the command 'get coins from corpses'
- In the 'Name' box, enter the name 'mymacro'
- Click the 'Add' button to create the macro

7.3.4 Editing macros

Like triggers and aliases, macros are customisable. Select 'mymacro' by clicking on it, and then click the 'Edit' button.

The new edit window contains three tabs. The 'Name' tab allows you to change the stimulus and response, and also allows you to enable or disable the macro.

The 'Attributes' tab allows you to customise the macro - for example, to disable the macro automatically after the first matching keypress.

When Axmud is testing macros against a keypress, it checks the macros in the same order every time. Sometimes the order is important, so the 'Before / after' tab allows you to name the macros that should be checked before or after this one.

7.4 Timers

Timers allow you to execute an instruction repeatedly at regular intervals. They can also be used to execute an instruction once, after a certain period of time.

Like all interfaces, Axmud timers have a stimulus and a response. The stimulus is a time interval, expressed in seconds. When this interval has passed, we say that the timer 'fires'.

The response can be any kind of instruction. It's usually a world command like 'score', but it could be a client command like *';playsoundeffect beep'* or any other kind of instruction (see Section 4.4).

7.4.1 Adding timers

Let's create a timer which will 'fire' every 60 seconds, and which sends the world command 'score' each time it fires.

- From the main window menu, click 'Edit > World timers'
- In the 'Timer stimulus' box, enter the time interval '60'
- In the 'Timer response' box, enter the command 'score'
- In the 'Name' box, enter the name 'mytimer'
- Click the 'Add' button to create the timer

7.4.2 Editing timers

Like triggers, aliases and macros, timers are customisable. Select 'mymacro' by clicking on it, and then click the 'Edit' button.

The new edit window contains three tabs. The 'Name' tab allows you to change the stimulus and response, and also allows you to enable or disable the timer.

The 'Attributes' tab allows you to customise the timer - for example, to disable the timer automatically after it fires for the first time.

When Axmud is testing whether it's time to fire a timer, or not, it checks the timers in the same order every time. Sometimes the order is important, so the 'Before / after' tab allows you to name the timers that should be checked before or after this one.

7.4.3 Timer attributes

Click on the 'Attributes' tab. There are several attributes which can be customised.

Repeat count

Some timers should continue firing indefinitely; others should only fire a certain number of times.

The repeat count sets how many times the timer fires, before disabling itself. You can use any positive integer value; alternatively, use the value -1 to make the timer repeat indefinitely.

Initial delay

Some timers should fire immediately, but usually you'll want the timer to fire for the first time after a delay.

If you use an initial delay of 0, an enabled timer fires as soon as it becomes active. Otherwise, it waits for the specified number of seconds before firing for the first time.

You will often set the repeat count and the initial delay to the same value, so that the timer fires (for example) 10 seconds after becoming active, then every 10 seconds after that.

Random delays

The timer's stimulus is an interval in seconds. If the stimulus is 10, the timer will fire every 10 seconds.

However, it's possible to introduce an element of surprise. When the checkbox is selected, the delay is a random interval between 0 and the stimulus (10 seconds, in this case). Every time the timer fires, a new random interval is chosen - so the timer might fire after approximately 2 seconds, then 5, then 1, then 9.

(There is a minimum system delay, set to a 0.05 seconds by default. No timer can ever fire more frequently than that.)

Minimum random delay

Sometimes you will want to specify a random delay that's between 5 and 10 seconds, rather than being between (almost) 0 and 10 seconds.

When the 'random delay' checkbox is selected, this value sets the minimum random delay. Specify the value 0 to use the system's shortest possible delay.

Start after login

Select this checkbox to prevent the timer firing before the character is marked as logged in. If you don't select the button, the automatic login process might be disrupted.

Temporary timer

If the checkbox is selected, the timer only fires once. After that, the active timer becomes an inactive timer - it will be available again, the next time you connect to the world, but it will not be available again during this session.

7.5 Hooks

Hooks respond to certain events. An example of a hook event is the one called 'login'.

Every time Axmud completes an automated login - or every time the user performs a manual login with the `';login'` command - the hook event is deemed to have happened (and we say that the hook 'fires').

The response can be any kind of instruction. It's usually a world command like 'inventory', but it could be a client command like `';playsoundeffect beep'` or any other kind of instruction (see Section 4.4).

7.5.1 Hook events

This is a complete list of Axmud's hook events.

- 'connect' - happens when the first data is received from the world (which might not be visible to the user)
- 'disconnect' - happens when we disconnect from the world or get disconnected from the world
- 'login' - happens when the character is marked as 'logged in', either after an automatic login or after using the `';login'` command
- 'prompt' - happens when Axmud receives a prompt from the world
- 'receive_text' - happens when Axmud receives a packet of text from the world (which may comprise one line or several)
- 'sending_cmd' - happens when a world command is about to be sent, but *before* it has been checked against aliases
- 'send_cmd' - happens when a world command is about to be sent, *after* it has been checked against aliases, and possibly modified by them
- 'msdp' - happens when some data is received using the MSDP protocol. This data is normally not visible to the user, but is available to your scripts
- 'mssp' - happens when some data is received using the MSSP protocol

- 'atcp' - happens when some data is received using the ATCP protocol
- 'gmcp' - happens when some data is received using the GMCP protocol
- 'current_session' - happens when this session becomes the visible session in the main window that has the system focus
- 'not_current' - happens when some other session becomes the visible session in the main window that has the system focus
- 'change_current' - happens in every session when the visible session changes in the main window that has the system focus
- 'visible_session' - happens when this session becomes the visible session in its own main window
- 'change_visible' - happens when a different session becomes the visible one in this session's own main window
- 'not_visible' - happens in every session when any main window's visible session changes
- 'user_idle' - happens when no world commands have been sent for 60 seconds
- 'world_idle' - happens when no text has been received from the world for 60 seconds
- 'get_focus', 'lose_focus' - happen when this session's main window becomes the active (focused) window, or when some other window becomes the active window
- 'close_disconnect' - happens just before Axmud stops running

7.5.2 Hook data

Some hook events provide more information about whatever it was that caused the event to happen. This information, which is called 'hook data', is available to any code that needs it.

For the 'prompt' and 'receive_text' events, the hook data is the received text, with all the escape sequences (colour codes) removed.

For the 'sending_cmd' event, the hook data is the world command that's about to be checked against aliases. For the 'send_cmd' event, the hook data is the world command that's about to be sent to the world.

Other hook events don't use hook data at all.

In a moment we'll discuss how to create hooks that use hook data, but first we'll discuss how to create simple hooks that respond with an ordinary world command.

7.5.3 Adding hooks

Let's create a hook which fires every time a world command is sent, and which plays a sound effect in response. Don't forget that sound effects won't work at all until you turn them on:

```
;sound on
;snd on
```

Now you can create the hook.

- From the main window menu, click 'Edit > World hooks'
- In the 'Hook stimulus' combo (drop-down) box, select 'send_cmd'
 - Don't select 'sending_cmd' by mistake!
- In the 'Hook response' box, enter the client command *';playsoundeffect beep'*
- In the 'Name' box, enter the name 'myhook'
- Click the 'Add' button to create the hook

7.5.4 Editing hooks

Like triggers, aliases, macros and timers, hooks are customisable. Select 'myhook' by clicking on it, and then click the 'Edit' button.

The new edit window contains three tabs. The 'Name' tab allows you to change the stimulus and response, and also allows you to enable or disable the hook.

The 'Attributes' tab allows you to customise the hook - for example, to disable the hook automatically after its hook event happens for the first time.

When Axmud is responding to hook events, it checks hooks in the same order every time. Sometimes the order is important, so the 'Before / after' tab allows you to name the hooks that should be checked before or after this one.

7.5.5 Using hook data

A hook's response can be an ordinary world command, a client command (like the one we created earlier, which uses *';playsoundeffect'*) or any other kind of instruction.

Those instructions include Perl commands. As explained in Section 4.4, a Perl command is a mini-Perl programme, all on one line. The programme's return value is processed as a normal instruction such as a world command.

Let's create one of these hooks now. This hook will add the word 'sail' before every world command. It will convert 'north' into 'sail north' or 'south' into 'sail south'. We'll be using the hook event 'sending_cmd'. This hook event happens whenever you type a world command, but before the command is checked against aliases.

If you're familiar with the Perl language, it will be obvious how the mini-programme works. If not, it will be obvious how to modify the mini-programme - and you can use the same mini-programme with many other hooks.

Don't forget that Perl commands are disabled by default. This hook won't work until you enable them again.

The mini-programme - including the initial forward slash character which starts all Perl commands - looks like this:

```
/my $string = 'sail '; return $string . $hookData;
```

The mini-programme's job is to convert the hook data - a string of text - into a modified string of text. The modified string is the programme's 'return value'. The return value is sent as a world command.

Now you're ready to create the hook.

- From the main window menu, click 'Edit > World hooks'
- In the 'Hook stimulus' combo (drop-down) box, select 'sending_cmd'
 - Don't select 'send_cmd' by mistake!
- In the 'Hook response' box, copy-and-paste the mini-programme from above (including the initial forward slash character and the final semicolon)
- In the 'Name' box, enter the name 'myhook2'
- Click the 'Add' button to create the hook

This particular mini-programme adds the word 'sail' to every command, but you can change 'sail' to any word you like.

If you'd prefer to add a word *after* the original command, instead of before it, you could use this mini-programme instead:

```
/my $string = ' axe'; return $hookData . $string;
```

This mini-programme could convert the command 'get' into 'get axe', the command 'drop' into 'drop axe' or the command 'sell' into 'sell axe'.

7.5.6 Redirect mode

Astute readers might now be asking themselves why there isn't an easier way to convert 'north' into 'sail north' automatically.

In fact, there are several possible methods, including the clever use of aliases. But a much easier way is to use Axmud's 'redirect mode'.

```
;help redirectmode
```

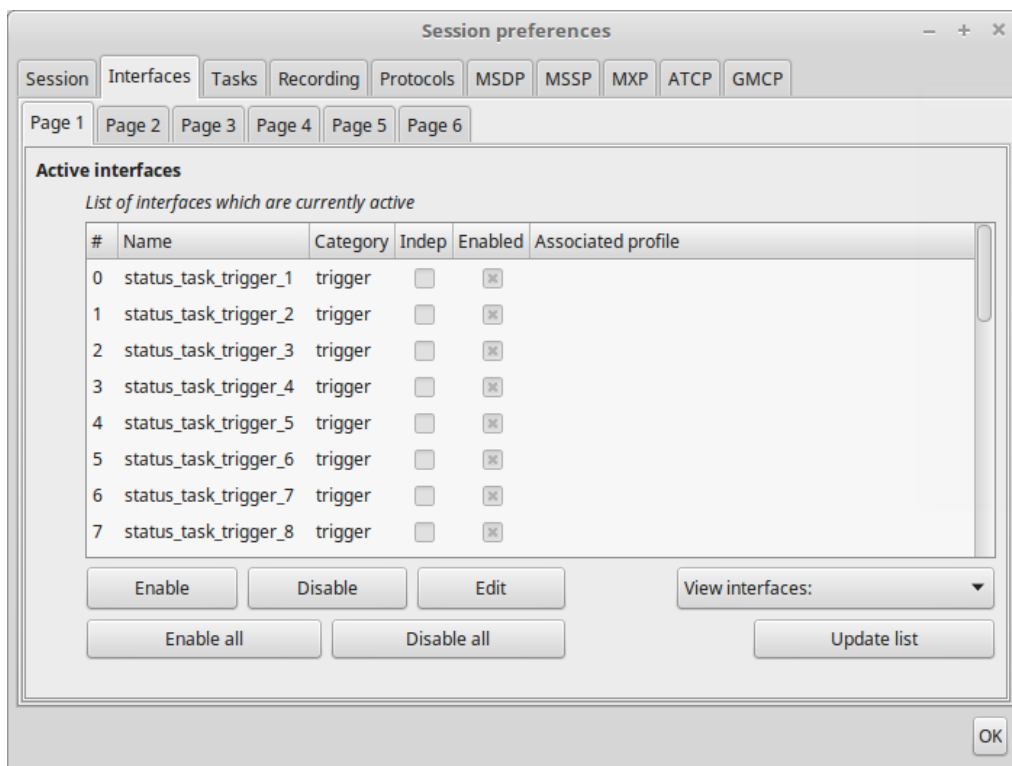
7.6 Active and inactive interfaces

As explained in detail in Section 5, Axmud interfaces (trigger, aliases, macros, timers and hooks) are either 'active' or 'inactive'.

You can create several triggers called 'mytrigger', each belonging to a different profile, but only one of them is active (actually in use). The others are inactive.

Actually, when an interface becomes active, Axmud makes a copy of it. The copy, not the original, is the 'active' interface.

You can see a current list of active interfaces from the main window menu (click on 'Edit > Active interfaces'). If the Status task is running, the list will look like this:



There is one more important thing to explain.

If you modify the original (inactive interface), the corresponding copy (active interface) is automatically updated.

- From the main window menu, click on 'Edit > World triggers'
- Enable (or disable) one of the triggers you created earlier
- Now, from the main window menu, click on 'Edit > Active interfaces'
- Find the corresponding trigger in the list. It will also be enabled (or disabled)

However, if you modify the copy (the active interface), *no changes are made to the original* (the inactive interface). Any changes are therefore *temporary* and will disappear at the end of the session.

8 Axmud scripting

All modern MUD clients allow their users to write scripts to help automate various aspects of the game.

Axmud offers several different approaches. The approach you choose will depend on your coding skills (from 'complete newbie' to 'it was me that invented the Internet') and on how much time you want to spend (ranging from 'I only have 30 seconds' to 'I'm going to re-invent the Internet!')

8.1 Types of script

Axmud interfaces - triggers, aliases, macros, timers and hooks - have already been discussed extensively (in Sections 5 and 7). They require a basic knowledge of regular expressions for which you'll find a tutorial in Section 6.

Missions are extremely simple scripts which require no programming knowledge at all. Just write a sequence of commands like 'north', 'open door' and 'disembowel dwarf', add a few pauses at the right moments, and you're finished! Missions are discussed in Section 9.

Axbasic is Axmud's own scripting language, fully compatible with other flavours of BASIC.

Although Axbasic is fairly primitive by today's standards, it's more than adequate for nearly everything you might want to do in a MUD. Axbasic is discussed in Section 10.

Tasks are scripts written in Perl. Tasks are designed so that all of the boring stuff - scheduling, handling of interfaces, maintaining task windows - is handled automatically, leaving you free to design ever more efficient ways of slaughtering your enemies.

Axmud provides twenty or so built-in tasks, some of which are more useful than others. They are discussed in Section 11.

Finally, it's possible to write your own Perl plugins. Plugins provide a way of modifying almost any aspect of Axmud's functionality. Besides writing new tasks, you can add new client commands, modify the main window menu, create new edit windows and much more besides. Plugins are described very briefly in Section 12, but you'll have to wait for the Axmud manual before you can seriously consider writing your own.

9 Recordings and missions

A mission is an extremely simple script that requires no knowledge of programming at all. (See Section 8 for a summary of the types of script available in Axmud.)

Missions are created by *recording* them. That is to say, you start by pressing a 'record' button. Then, you send your character around the world, slaying enemies, looting treasures, and so on. Axmud records every world command that is sent.

When you've finished, you can convert the recording into a mission. (There are other uses for recordings, too.)

9.1 Starting a recording

Recordings can also be started/stopped/paused/resumed from the main window menu. In addition, you can use the following client command to start/stop a recording:

```
;record  
;rcd
```

You can use the following command to pause/resume a recording:

```
;pauserecording  
;prc
```

Starting a new recording erases the contents of any previous recording.

During a recording, every world command is added to the recording. However, client commands and some other types of instruction, such as Perl commands, are *not* added.

Recordings can also be started/stopped/paused/resumed from the main window menu. In addition, you can use the following client command to start/stop a recording:

9.2 Creating a mission

When you've finished recording you can create the new mission. The contents of the recording - even if it's not finished - will be added to the mission automatically.

If you want to first check the contents of the recording, you can do so:

```
;listrecording  
;lrc
```

Every mission needs a unique name, so choose one now. Some words are reserved, so if Axmud refuses to create the mission, try choosing a different name.

```
;addmission myquest  
;amn myquest
```

9.3 Starting the mission

Starting a mission *plays back* the recording, re-sending all the world commands that were sent the first time.

Missions can be done all at once or one step at a time. Use the following command to do the mission all at once:

```
;startmission myquest  
;smn myquest
```

Alternatively, use the following command to do the mission one step at a time:

```
;startmission myquest -i  
;smn myquest -i
```

Use this command to do the next step (by sending the next world command):

```
;nudgemission  
;nmn
```

9.4 Adding breaks

Rather than using '*nudgemission*' repeatedly, you'll often want to split the mission into Sections. You can do this by adding breaks.

During the recording, use this command at any time to add a break:

```
;break  
;brk
```

Once the recording has been saved as a mission, you can start the mission in the normal way:

```
;startmission myquest  
;smn myquest
```

Axmud will execute commands until it finds the first break. It will then wait until you ask it continue the mission, which you can do with the '*mission*' command:

```
;mission  
;mn
```

The mission will then continue until the next break (or until the end of the mission). You could also use the '*nudgemission*' command after a break, if you only want to advance by the next step.

9.5 Editing missions

Missions are stored in the current world profile.

- Open the world's edit window with the `';editworld'` command
- Scroll through the tabs until the 'Mission' tab is visible
- Select the mission you created earlier by clicking it
- Click the 'Edit' button to open the mission's edit window

Axmud missions consist of a series of commands which are executed from beginning to end. Each command has its own line.

The first character in every line specifies which type of command this is. A normal world command begins with a greater than (>) character:

```
> kill orc
```

Your missions will be much easier to read if you add some space between the > character and the command itself, but this is optional.

Recordings don't record client commands, but in this window you can add a client command starting with a semicolon (;) character. Because this is a mission, you can add space between the semicolon and the command itself.

```
; playsoundeffect beep
```

You can also add a speedwalk command:

```
. 3nw2s
```

In this window you can also add a comment, starting with a hash (#) character. Comments are displayed in the main window in the same colour as an Axmud system message.

```
# Don't forget to phone the restaurant!
```

Comments are normally used immediately before a break. During the break, the user can intervene to achieve some objective that's too complex for the mission - solving a puzzle, perhaps, or killing an unpredictable number of orcs.

An ordinary break is a line with a single 'at' (@) character on it:

```
# Kill the orc by yourself!  
@
```

There are three other kinds of break. A 'trigger break' creates a temporary trigger, using the pattern specified on the same line:

```
t You kill the orc
```

The mission waits until the trigger fires - that is to say, until the world sends some text which matches the pattern 'You kill the orc'. When that happens, the mission automatically resumes. (The temporary trigger is automatically destroyed.)

A 'pause break' waits for a certain period of time. For example, to wait for ten seconds, add the line:

```
p 10
```

9.6 Locator breaks

The Locator task is one of Axmud's built-in tasks, and is responsible for interpreting room descriptions received from the world. (Axmud users prefer the term 'room statements'.)

The interpreted data is then available for use by any other part of the Axmud code, including the automapper.

One of task's features is its ability to monitor movement commands such as 'north', 's', 'up' and 'enter'. The task is able to make an educated guess about how many room statements the world is about to send.

If we add a Locator break to our mission, it will wait until the Locator task is not expecting any more room statements. When the Locator task reports that it isn't expecting any more room statements - in other words, that the character has arrived at their destination - the mission resumes automatically.

Here's an example of a mission with a Locator break. The last-but-one line is the Locator break itself - a line with just the letter 'l' (for Locator) on it.

```
> north
> north
> northeast
> east
> open door
> in
l
> kill orc
```

Note that if slowwalking is turned on (see Section 4.4.6), the Locator break will last for at least as long as it takes to clear the command queue.

10 Axbasic

Axbasic is Axmud's own scripting language, based on the BASIC language first released in the 1960s and which was commonly used on home computers in the 1970s and 80s.

BASIC is considered old-fashioned, but it has a distinct advantage in that most people already know how to use it and, even if not, it's simple enough to be learned in just a few hours.

10.1 Axbasic script example: Hello world!

Axmud's data directory (folder) can be found in your home directory. It contains a sub-directory called 'scripts' which contains three example scripts.

'hello.bas' is the traditional 'Hello world!' script:

```
REM A trivial Axbasic script

PRINT "Hellow world!"
END
```

If the temptation is too great, you can run the script to see what it does:

```
;runscript hello
;rs hello
```

The '*runscript*' command assumes that there is a script called '*hello.bas*' in this directory. If it finds one, the script is executed. (This behaviour can be modified so that other directories are checked too, if desired.)

10.2 Axbasic script example: Test script

The '*test.bas*' example script does nothing, at first. You can use it for testing Axbasic code.

Its main advantage is that it can be run with *any* of the following commands:

```
;runscript test
;rs test

;runscript
;rs
```

10.3 Axbasic script example: Hunt The Wumpus!

The third example script, 'wumpus.bas', is a copy of the 1972 classic *Hunt the Wumpus*.

It's not much fun to play, but nevertheless, Axbasic is compatible with BASIC programmes from this era and will run *Hunt the Wumpus* without complaints:

```
;runscript wumpus
;rs wumpus
```

10.4 Testing scripts

Axbasic scripts can be tested without being run. Use the following command:

```
;checkscript wumpus
;cs wumpus
```

Hopefully, the test will report that there are no errors. (It would be surprising if you could find one, 45 years after the game was written.)

You can use the same command to check the 'test.bas' script, simply by omitting the script name:

```
;checkscript
;cs
```

There is also a useful little command which opens a script in a text editor:

```
;editscript wumpus
;es wumpus
```

As always, if you leave out the script name, Axmud uses the 'test.bas' script:

```
;editscript
;es
```

10.5 Running scripts as a task

You can use the '*runscript*' command for simple scripts. It runs the script from beginning to end, without pausing.

This is fine for simple scripts, but often you'll want something more flexible. You will want the script to pause at certain times, or to wait for something to happen, or to display text in its own window. In these situations, you can use the Script task - one of Axmud's built-in tasks - to run the script on your behalf.

To run a script from inside the Script task, use the '*runscripttask*' command:

```
;runscripttask wumpus
;rst wumpus
```

As always, if you don't specify a script name, the *'test.bas'* script is run:

```
;runscripttask  
;rst
```

Some Axbasic keywords such as WAITARRIVE and WAITTRIG won't work unless the script is run from within a task. Here's an example of what scripts like these can do. (All lines beginning with a ! character are comments, which are ignored.)

```
! Kill an orc and return home  
  
! Move to the killing zone  
MOVE "north"  
MOVE "northwest"  
MOVE "north"  
SEND "open door"  
MOVE "in"  
  
! Wait for your character to arrive  
WAITARRIVE  
  
! Kill the orc  
SEND "kill orc"  
  
! Create a trigger to wait for the orc's death  
WAITTRIG "You kill the orc"  
  
! The orc is now dead; go back home  
SEND "open door"  
MOVE "out"  
MOVE "south"  
MOVE "southeast"  
MOVE "south"  
  
! All Axbasic scripts must contain an END statement  
END
```

10.6 Axbasic help

Axmud provides extensive documentation on Axbasic's keywords and functions. A summary can be seen using this command:

```
;axbasichelp  
;abh
```

The same command can be used to show help on a particular topic:

```
;axbasichelp send  
;axbasichelp waittrig
```

Axbasic's syntax is virtually identical to that of True BASIC. If you want to know more about writing BASIC scripts in general, you can dip into the TRUE BASIC manual, a free download from

http://www.truebasic.com/free_and_demos

10.7 Retrieving Axmud data

Axbasic scripts enjoy full access to Axmud's internal data. Here's a brief example of how to exploit that capability.

```
REM What am I?  
PEEK guild$ = "guild.current.name"  
  
IF guild$ = "thief" THEN  
    PRINT "I am a thief!"  
ELSE  
    PRINT "I am not a thief!"  
END IF  
  
END
```

Most of Axmud's internal data can be accessed using strings like "race.current.name", and some of it can be modified using a POKE statement.

(Note that, if you haven't set your current character's guild, then of course this script would not work as intended.)

11 Tasks

Tasks are scripts written in Perl.

Tasks are designed so that all the boring stuff - scheduling, handling of interfaces, maintaining task windows - is handled automatically, leaving you free to design ever more efficient ways of slaughtering your enemies.

11.1 Built-in tasks

Axmud provides twenty or so built-in tasks, some of which are more useful than others. This is a brief summary of what they do.

- Advance task
 - Automatically advances your character's skills. This task might be useful at a few worlds, but will be completely useless at many others.
- Attack task
 - Monitors attacks and, like the Status task, updates the data stored in the current character profile. (The task doesn't initiate attacks - that's your job.)
- Chat task
 - An instant messenger service. Contains an almost complete implementation of the MudMaster and zChat protocols (only encryption has not yet been implemented).
- Compass task
 - Hijacks your keyboard's keypad, allowing you to move around the world with a single keypress. This is a convenient way to set up macros and to enable/disable them at will.
- Condition task
 - Works alongside the Inventory task. Tries to keep track of the condition of items in the your character's inventory (e.g. is your armour in perfect condition, or is it falling to pieces?)
- Debugger task
 - Diverts (or copies) system messages (error, warning, debug and improper arguments messages) into a separate task window.
- Divert task
 - Diverts text from the main window to the task's own window. Mostly used for social messages (chats, tells, and so on), but can be configured to divert text matching any pattern.
- Frame task
 - Used by the MXP and Pueblo protocols to create new windows for various purposes.

- Inventory task
 - Keeps track of your character's inventory, displaying a summary in the task's own window.
- Launch task
 - Displays a list of Axbasic scripts in a task window. You can add, edit and delete scripts, as well as running them, by clicking on the window's buttons.
- Locator task
 - Interprets room statements (descriptions) received from the world. The interpreted data is then available to other parts of the Axmud code, such as the automapper. The task window is optional. If open, it shows a summary of the current location.
- Notepad task
 - A notepad that allows you to keep notes that are preserved for future sessions. There's a separate notepad for every current profile (one for the current world, one for the current character, and so on). The notes are available to view whenever the profile is current.
- RawText task
 - Another debugging task. Displays all text received from the world, including all escape sequences and special characters.
- RawToken task
 - Yet another debugging task. Displays all text received from the world split into tokens. Newline characters count as a single token, as do escape sequences. Uninterrupted sequences of normal characters are interpreted as a single token.
- Script task
 - Works alongside scripts written in Axmud's own scripting language, Axbasic (see Section 10)
- Status task
 - Keeps track of your character's XP, health points and so on. The intercepted data is then available to other parts of the Axmud code, such as the your own scripts and the gauges visible near the bottom of the main window. Data stored in the current character profile is automatically updated. The task window is optional. If open, it shows a summary of the intercepted data.
- TaskList task
 - A debugging task that displays a list of all tasks running at the moment. The list is updated several times a second.
- Watch task
 - Re-displays any text displayed in all Divert and Chat task windows, across every session. Useful for when you are connected to several worlds simultaneously.

11.2 Task dependencies

Some Axmud tasks try to interpret text from the world, in some cases storing the interpreted data in a profile.

Axmud does not have some kind of magical ability to recognise text from any world and in any language. Instead, it must be *taught* how to recognise certain patterns. (If you are using one of the pre-configured worlds, most of this work will already have been done.)

Those patterns are typically stored in a profile, usually the world profile.

Some tasks are particularly dependent on the patterns you've collected. The Locator task won't work at all unless Axmud has been taught how to interpret a room statement (description). The Status task won't work very well unless Axmud has been taught how to interpret text like 'You have 1000 gold coins'. The Divert, Inventory, Condition, Advance and Attack tasks are also reliant on data stored in profiles. Some tasks, however, don't refer to data stored in any profile.

Tasks that depend on one or more profiles automatically reset themselves if you set a current profile (for example, if you change the current character from 'Gandalf' to 'Bilbo' using a `';setchar'` command).

11.3 Task commands

There are several client commands which can be used with all tasks. In addition, many tasks have a set of client commands devoted to them.

You can find out more about each kind of task, and the client commands they use, with the `';taskhelp'` command:

```
;taskhelp status
;th status
```

Commands like `';taskhelp'` are quite flexible about how the task is specified. All of the following commands will work:

```
;taskhelp loc
;taskhelp locator
;taskhelp locator_task
;taskhelp locator_task_3
```

This is how to start and stop a task:

```
;starttask locator
;st loc

;halttask locator
;ht loc
```

Most of the built-in tasks are 'jealous', meaning that you can't run two copies of them at the same time.

You can also pause and resume tasks:

```
;pausetask locator
;pt loc

;resumetask locator
;rt loc
```

In an emergency, you can use the '*freezetask*' command, which freezes all currently-running tasks. Use the same command again to unfreeze them.

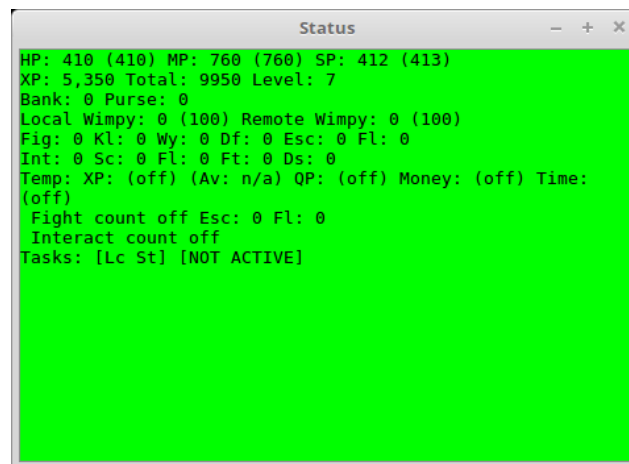
You might also find the '*listtask*' command useful. It displays a list of all the tasks that are currently running. (This list is called the 'current tasklist'.)

```
;listtask
;lt
```

All of these actions are also available in the object viewer window (see Section 4.6).

- In the left-hand column, double-click on 'Tasks' and then click on 'Available tasks'.
- Select a task by clicking on it
- Then click the 'Start', 'Halt', 'Pause', 'Safe resume' or 'Help' buttons

11.4 The Status task



The Status task keeps track of your character's XP, health points and so on. The intercepted data is then available to other parts of the Axmud code, such as the your own scripts and the gauges visible near the bottom of the main window. Data stored in the current character profile is automatically updated. The task window is optional. If open, it shows a summary of the intercepted data.

The window's layout can be customised. The picture above shows the layout used with the Dead Souls pre-configured worlds; other pre-configured worlds will use a different layout.

11.4.1 Updating the Status task

The task collects data passively: in other words, if you type the 'score' command, the world will send back some text, the Status task will (hopefully) recognise that text, and the XP displayed in the task window will be updated. If you don't type 'score' for twenty minutes, the value will be twenty minutes out of date.

If this is inconvenient, you can ask the task to send commands like 'score' at regular intervals:

```
;addstatuscommand score 60
;asc score 60
```

This will send the 'score' command every 60 seconds. You can use any command and any time interval. If the command contains spaces, you should enclose it within diamond brackets:

```
;addstatuscommand <look at watch> 60
;asc <look at watch> 60

;activatestatustask
;ast
```

The task won't send these commands right away; you must first put the task into 'active' mode:

```
;activatestatustask
;ast
```

To stop sending the commands, you can put the task into 'disactivated' mode:

```
;disactivatestatustask
;dst
```

However, this is only half the story. Several MUD protocols send data that's normally invisible to users, and this data typically includes information like the character's current XP.

If you're connected to a world which sends this data, you'll notice the task window and/or main window gauges update themselves without any intervention from you.

11.4.2 Status task background

The task window's background colour changes according to your character's current health. Green means the character is relatively healthy, red means the character is at death's door, and black means the character is actually dead.

The colour can also change when the character dies, gets resurrected, falls asleep, wakes up, loses consciousness or comes around (but only if these events are implemented by the game).

11.4.3 Working with the Attack task

The Attack task - if it is also running - monitors any fights involving your character. Typically it looks out for patterns like 'You kill (.*)' and '(.*) runs away!'

It is also capable of monitoring other kinds of interaction - muggings for thief characters, healings for cleric characters, and so on.

The task gathers information and passes it on to the Status task, so that the combined data can be displayed in one window. You'll notice that the Status task's fight counter (Fig), Kill counter (Kl) and so on are only updated when both the Status and Attack tasks are running.

11.4.4 Status task counters

Sometimes you'll want to track the number of kills, or how much XP has been gained, in a certain zone of the world. The temporary counters (on the line that starts 'Temp') can do this.

To start the temporary counters, or to reset them back to zero, use this command:

```
;resetcounter  
;rsc
```

11.4.5 Modifying Status task patterns

Once again, it must be stressed that Axmud cannot magically interpret text from all worlds. It must be taught how to recognise patterns. If you're using a pre-configured world, some of those patterns have already been added.

Most of the patterns used by the Status and Attack tasks are stored in the current world profile. (See Section 6 for a brief tutorial about how to write your own patterns.)

- Open the world's edit window with the '*editworld*' command
- Click on the 'Status' tab to see patterns used by the Status task
- The task window's layout can be configured on 'Page 2'
- Click on the 'Attack', 'Fight' and 'Interaction' tabs to see patterns used by the Status task

If you make any changes to these patterns, they won't be applied until you reset the Status and/or Attack tasks, for example:

```
;resettask status  
;rtt stat
```

By the way, some of the values collected by the Status task are stored in the current character profile.

- Open the character's edit window with the '*editchar*' command
- Click on 'Settings > Page 1' to see the character's current XP
- Click on the 'Attacks' tab to see information collected by the Attack task

11.5 The Locator task

The Locator task interprets room statements (descriptions) received from the world.

The information that can be collected varies from world to world, but it typically includes the room's title, verbose description, exit list and one or more lines describing the room's contents.

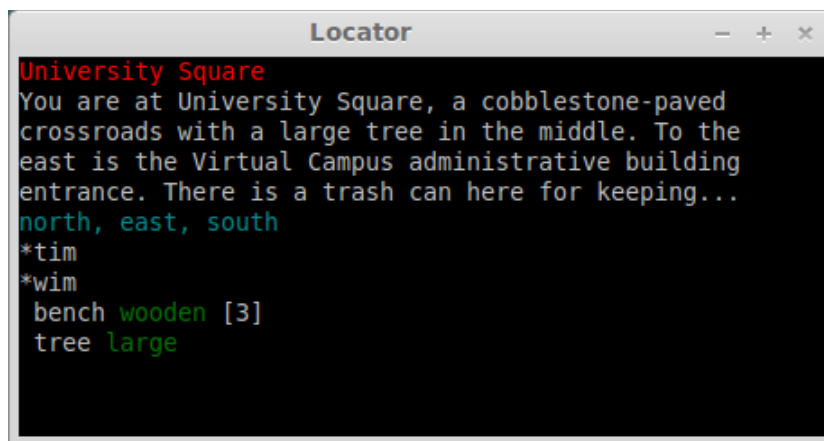
Once collected, the interpreted data is available to other parts of the Axmud code, notably the automapper. The automapper relies on the Locator task; if the task is not running, the automapper cannot function.

In fact, the Locator task is so important that Axmud provides a 'Locator wizard'. The wizard teaches Axmud how to recognise room statements (with just a little help from you). See Section 15.1

The Locator task can run with a task window or without one. When the window is open, it displays a summary of the interpreted data.

11.5.1 The Locator task window

Here we can see a typical location in Dead Souls.



```
Locator
University Square
You are at University Square, a cobblestone-paved
crossroads with a large tree in the middle. To the
east is the Virtual Campus administrative building
entrance. There is a trash can here for keeping...
north, east, south
*tim
*wim
bench wooden [3]
tree large
```

In Axmud terminology, a 'room statement' is the complete description of the room received from the world, including the list of exits and the room's contents list. The Locator task tries to capture as much of this information as possible.

The different parts of a room statement are called 'components'. The red line at the top is the room's title. (Some worlds use a brief description, and this typically appears in red, instead.)

Below that is the verbose description component. The whole description has been captured, but the task shortens it to fit the window.

The room contains six objects. Tim and Wim are non-player characters (NPCs), so there is an asterisk (*) next to their names. If one of your scripts has marked them as killed, this will change to a hyphen (-).

There are three identical wooden benches. You'll notice that the most important word - the noun - comes first, and that everything else follows it (in a different colour). You'll also notice that the task saves space by listing multiple inanimate objects on the same line.

11.5.2 Resetting the task

The Locator task keeps track of the world commands you send and tries to work out how many room statements to expect. If you send the following sequence of commands:

```
n;n;n;n;n;n;n;n;n;n;n;n;n;n;n;n;n;n;n;n;n;n
```

...the task will be expecting twenty room statements. These statements will be counted off as they are received; you'll see a number in the window's title bar which will (hopefully) decrease until the task is expecting no more rooms.

Occasionally the task will lose track. If you notice that the character has arrived at their destination but the task is still expecting five more rooms, then the task needs to be reset.

```
;resetlocatortask  
;rlc
```



This main window button is a more convenient way of resetting the Locator task. You can also use the menu: click on 'Tasks > Locator Task > Reset task'.

11.5.3 The current dictionary

How does the Locator task know that Tim is a living, breathing NPC, but that the wooden bench is not?

In a different context, Axmud could simply consult a standard English dictionary (or one in French, and so on). Unfortunately, most MUDs invent their own names for NPCs, and most player characters (PCs) at these worlds also have names that won't be found in any standard dictionary.

Axmud uses its own dictionary objects. Besides many other things, these dictionaries store list of vocabulary.

Since most worlds have their own unique names for everything, Axmud allocates each world profile its own dictionary that has the same name as the world. (This is not a fixed rule; worlds can share dictionaries, if required.)

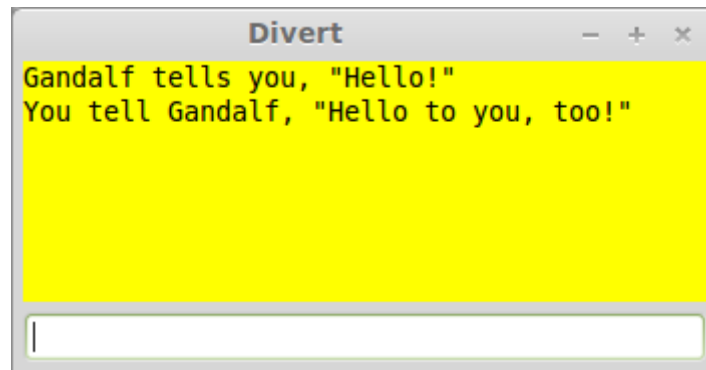
There are several ways to add new words to an Axmud dictionary. For example, to add Slartibartfast to the dictionary as an NPC, you can use the dictionary's edit window.

- Use the '*editdictionary*' client command
 - If you don't specify a dictionary name, the current world's dictionary is used
- Click on 'Nouns > Page 1'
 - This page lists sentient beings. Non-sentient creatures are added in Page 2
- In the box labelled 'Word', add 'Slartibartfast'
- Click the 'Add' button, then the 'OK' button at the bottom of the window

11.6 The Divert task

The Divert task channels text from the main window to the task's own window.

It's mostly used for social messages (chats, tells, and so on), but the task can be configured to divert text matching any pattern. This is very useful for making sure you don't miss an important message during the heat of battle.



When some text is diverted, the window's background colour changes temporarily. In this case, it has turned yellow signifying that the text is a 'tell' message.

For your convenience, the window provides a command entry box. Any type of instruction, including client commands and speedwalk commands, can be typed in this box. This is useful if you're connected to a world, but busy doing something else; as long as the Divert window is visible on your desktop, you can reply to your friends when they talk to you.

If you're connected to several worlds at the same time, you might also find the Watch task useful. Everything displayed in Divert and Chat task windows, in any session, is copied into any Watch task window that's open.

11.6.1 Adding divert patterns

Many of the pre-configured worlds already know how to divert social messages to the Divert task's window, but you can add new patterns, if you want.

It's possible to use a client command for this (see the help for `';adddivertpattern'`) but it's easier to add patterns in the world's edit window.

- Open the edit window with the `';editworld'` command
- Click on the 'Divert' tab
- Patterns for 'tell' messages can be added in 'Page 1'
 - 'Tell ignore patterns' are for lines of text that look like a 'tell' message, but aren't
 - e.g. 'The shopkeeper tells you, I'm not buying that!'
- Patterns for public channels can be added in 'Page 2'
- Other kinds of patterns can be added in 'Page 3' and 'Page 4'

11.7 The Chat task

The Chat task is an instant messenger service. It is an almost complete implementation of the MudMaster and zChat protocols (only encryption has not yet been implemented).

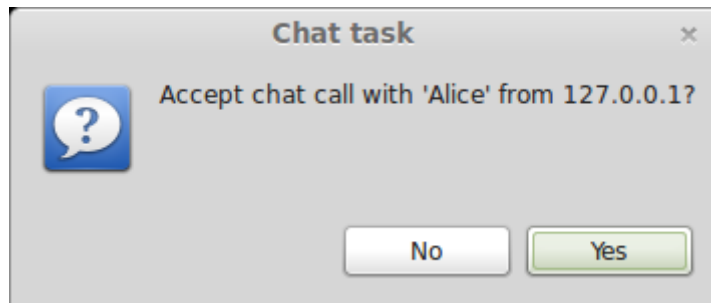
Several popular MUD clients implement these protocols, so your friends don't need to be using Axmud; they only need to be using a compatible MUD client.

There are some issues of security to consider, so we recommend you read the whole of this Section before using the task.

You can get started by listening out for incoming calls.

```
;chatlisten  
;listen
```

When you receive an incoming call, you can either accept or decline it.



For outgoing calls you'll need to know your friend's IP address and the port they're using for chat sessions.

Calls can be made with either protocol. The zChat protocol is slightly more powerful, so let's use that.

```
;chatzcall 159.200.36.251 4050  
;zcall 159.200.36.251 4050
```

Most MUD clients use the port 4050, so it's normally safe to omit it.

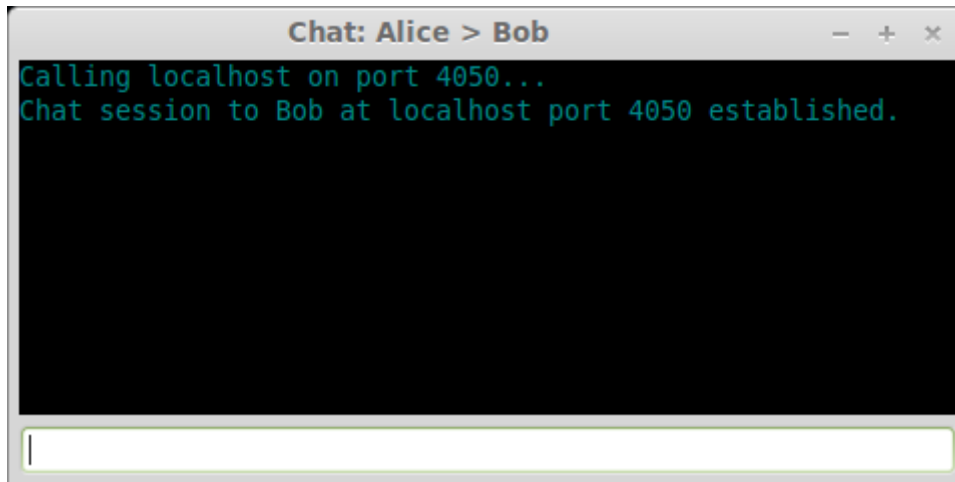
```
;chatzcall 159.200.36.251  
;zcall 159.200.36.251
```

You can also call someone using the main window menu.

- To listen for incoming calls, click on 'Tasks > Chat task > Listen for incoming calls'
- To make an outgoing call, click on 'Tasks > Chat task > Chat using zChat...'

11.7.1 Chat task commands

Once a connection is established, a task window will appear.



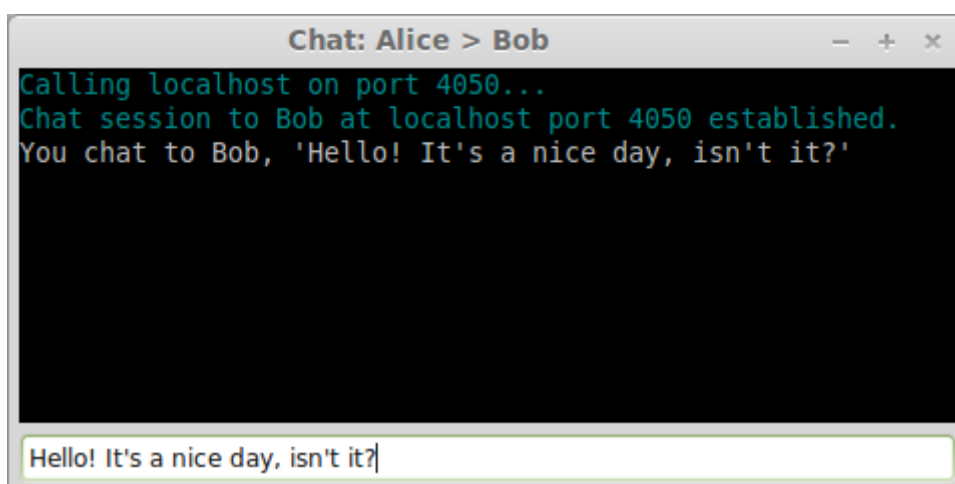
There are about thirty client commands used with the Chat task, all of which are typed in the main window. For a list of them, type:

```
;taskhelp chat  
;th chat
```

There is a *separate* list of commands used in the task's own window. All of these commands begin with the usual semicolon (;) character. To see this list, type '*help*' in the task window.

11.7.2 Chatting and emoting

To chat with someone, just type into the window.



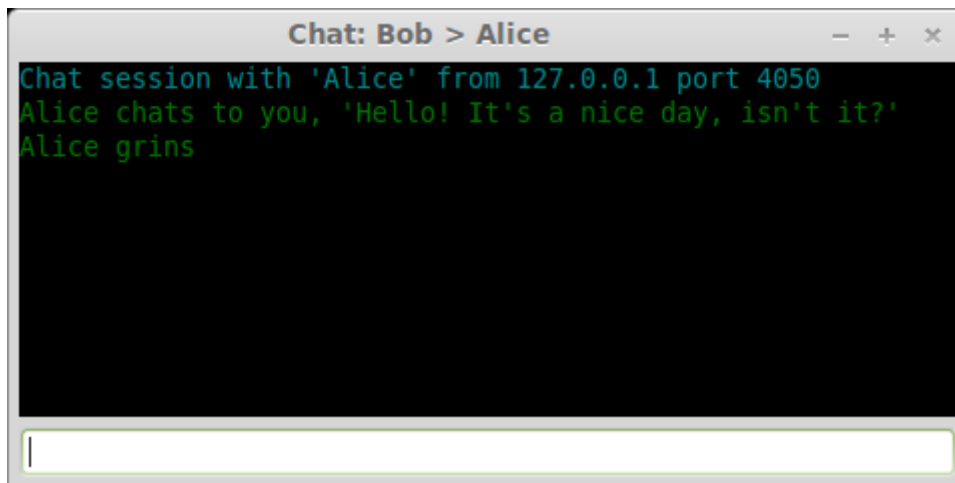
You can also send emotes. In the main window, type:

```
;emote grins
```

In the task window, any command that starts with a colon (:) is also treated as an emote:

```
:grins
```

In either case, your friend will see something like this:



To hang up, just close the window. Alternatively, use the following command in the main window, which will hang up on everyone:

```
;chathangup  
;chu
```

11.7.3 Chat contacts

It would be rather inconvenient to type an IP address like '159.200.36.251' every time you wanted to chat with a friend. Luckily, Axmud allows you to save your chat contacts so they're available at a click of a button.

- In the main window menu, click 'Edit > Axmud preferences'
- When the preference window opens, click 'Chat > Page 4'
- Add your friend's name, IP address and port, and click the 'Add' button

On 'Page 3' of the same edit window you can choose the name, email address and icon which are sent to your friends, whenever you open a chat session with them.

- If you don't specify a name here, the current character's name is used
- Axmud provides several hundred icons to choose from.
 - Cycle through them with the '<<' and '>>' buttons
 - If you're feeling lucky, try the 'Use random' button
- When you find an icon you like, click on 'Apply these changes now'
 - If you have any chat sessions open, your friends will be able to see the name/email/icon immediately

11.7.4 Firewalls

If you aren't able to establish a chat session with someone, it's probably because a firewall (yours or your friend's) is blocking the connection.

Configuring a firewall is beyond the scope of this guide, but you can probably find the help that you need somewhere on the internet. Unblocking port 4050 will usually solve the problem.

11.7.5 Security considerations

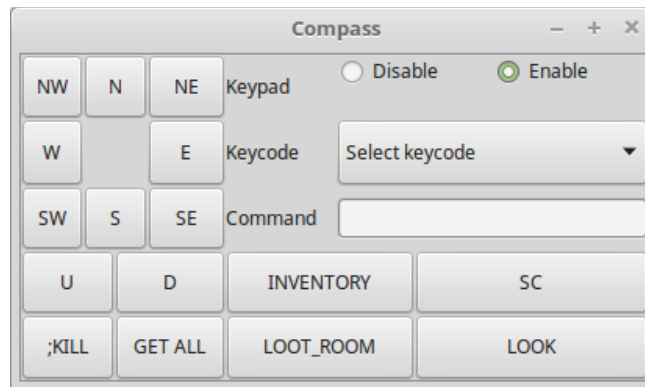
The MudMaster and zChat protocols are by now rather ancient, so there are some risks associated with using Axmud's Chat task.

Firstly, you should note that Axmud's authors are not experts in computer security. Using the Chat task *should* be safe, but there are no guarantees that a determined miscreant won't be able to find a way to abuse it.

The Chat task is able to send and receive files (see the help for ';chatsendfile'), view what is happening in your friend's connections (see the help for ';chatsnoop') and even to control those sessions remotely.

None of those features are enabled by default; you must give explicit permission before receiving a file or allowing someone to snoop on your connection. Nevertheless, you should exercise caution before allowing someone you've never met to gain a foothold on your computer.

11.8 The Compass task



The Compass task hijacks your keyboard's keypad, allowing you to move around the world with a single keypress.

- On full-size keyboards, the keypad is usually on the right - the keys arranged to resemble a calculator
- On smaller keyboards - especially laptop keyboards - the keypad shares other keys which you need for typing commands
 - Users of small laptops won't find the Compass task very useful
- The keypad is hijacked as soon as you start the task (by default)
 - To liberate the keypad, click the 'Disable' button.
- Use the keypad's '8' key to move north, the '2' key to move south, and so on

Other keypad keys are also hijacked, for example:

- Use the plus (+) key to move up and the minus (-) key to move down
- Use the 5 key to send a 'look' command to the world
- Use the divide key to send an 'inventory' command to the world

You can change the command sent when some keypad keys are pressed.

- In the combo (drop-down) box labelled 'Select keycode', select 'kp_divide'
- In the box just below, type 'kill orc' (and press RETURN)
- The new command is now sent every time you press the keypad divide (/) key

In addition to using keypad keys, you can click the task window's buttons. These buttons work even when 'Disable' is selected.

- Changing a key's command also changes the corresponding button

12 Initial and custom tasks

12.1 Initial tasks

When you first run Axmud, the Setup wizard window asks you which tasks should be started automatically at the beginning of each session.

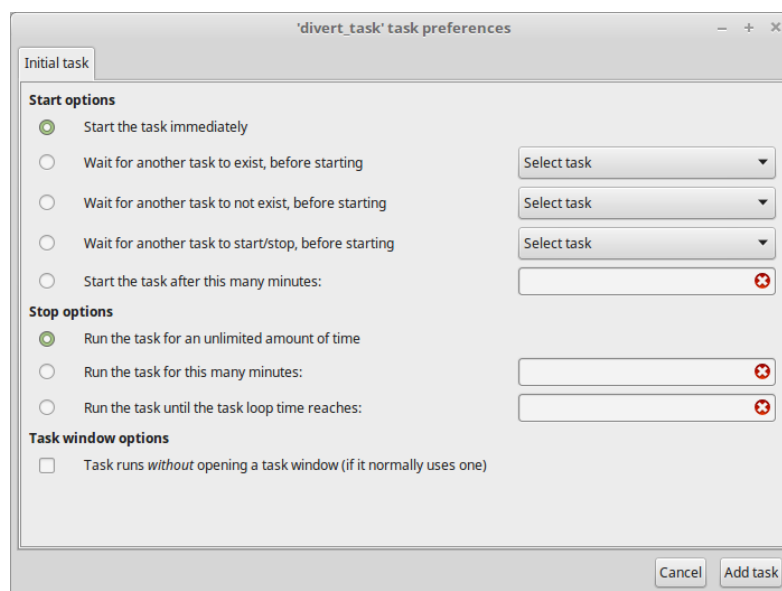
There are several lists of 'initial tasks'. The main one - the so-called 'global initial tasklist' - can be viewed from the object viewer window.

- Open the object viewer window using the `;openobjectviewer` command, or by clicking the button in the main window
- When the window opens, click on 'Tasks > Global initial tasklist'

The object viewer window also lists the *types* of task that are available, including both built-in tasks and any tasks you have written yourself.

- In the object viewer window, click on 'Tasks > Available tasks'
- Click the Divert task to select it
- Then click on 'Add initial'
 - A few tasks, such as the Chat task, can only be started with a client command
 - Therefore they can't be added to any initial tasklist

Axmud has a number of task scheduling options. When you click the 'Add initial' button, this window will appear:



You'll probably never need any of these options, so you can click the 'Add task' button right at the bottom of the window right away.

12.1.1 Profile initial tasks

Besides the global initial tasklist, each world profile also has its own initial tasklist. These tasks are started every time you connect to the world. (They are also started if you change the current world profile with a `;'setworld'` command.)

Tasks can be added to a profile's initial tasklist using the profile's edit window.

- Open the edit window using the `;'editworld'` command
- When the window is open, click 'Initial > Page 1'
- In the combo (drop-down) box near the bottom-left corner, select 'compass_task'
- Click the 'Add initial task' button

Other types of profile have their own initial tasklists, too. The character profile's initial tasks are started whenever you connect to the world and log in with that character.

- Many tasks (such as the Locator task) are 'jealous', which means only one copy of them can run at a time
- If you add the Locator task to multiple initial tasklists, only one copy will start

12.2 Editing tasks

Some aspects of a task's behaviour can be modified.

12.2.1 Editing current tasks

For example, the Divert task's window changes its background colour whenever a 'tell' message is received. If you find the bright yellow uncomfortable, you can change it.

- If the Divert task isn't already running, start it with `;'starttask divert'`
- From the main window menu, click 'Tasks > Divert task > Edit current task...'
- When the edit window opens, click on the 'Parameters' tab
- Find the combo (drop-down) box on the line marked 'Tell colour'. It is currently set to YELLOW. (Bold colours are displayed in CAPITAL LETTERS)
- Change the colour to MAGENTA
- Click the 'OK' button at the bottom of the window

The current Divert task will reset itself automatically. The next 'tell' message you receive will use the new colour.

12.2.2 Editing initial tasks

Any changes you make to the *current* divert task are lost at the end of the session (or when you halt the task).

However, you can edit initial tasks, and any changes you make will be changes are retained between sessions.

- Open the object viewer window using the `;'openobjectviewer'` command, or by clicking the button in the main window
- When the window opens, click on 'Tasks > Global initial tasklist'
- Select the Divert task you created earlier by clicking on it
- Click the 'Edit' button
- When the edit window opens, click the 'Parameters' tab, modify the colours, and then click the 'OK' button.

In this example, any changes you make will be applied the next time you connect to *any* world. If you edit a world profiles's initial tasks, the changes will be applied the next time you connect to *that* world.

12.3 Custom tasks

'Custom' tasks can be modified in the same way as initial tasks. However, unlike initial tasks (which usually start when you connect to a world), custom tasks can be started whenever you need them.

12.3.1 Adding custom tasks

You can create a new custom task from the object viewer window.

- Open the object viewer window using the `;'openguiwindow'` command, or by clicking the button in the main window
- When the window opens, click on 'Tasks > Available tasks'
- Select the Watch task by clicking on it
- Then click the 'Add custom' button

All custom tasks must have a unique name. When the task preferences window opens, you can enter this name in the box near the top. (Once again, you will probably never use the scheduling options in the rest of the window.)

Enter a name like 'mywatch' and then click the 'Add task' button at the bottom of the window.

12.3.2 Editing custom tasks

Once the custom task has been created, you can modify its behaviour.

- In the object viewer window, click on 'Tasks > Custom tasklist'
- Select the task by click on it
- Then click the 'Edit' button

You can modify the task's colours in the same way as before. Click the edit window's OK button when you're finished.

12.3.3 Starting custom tasks

You can now start your custom Watch task at any time. All custom tasks are available when connected to any world.

```
;startcustomtask mywatch  
;sct mywatch
```


13 Axmud plugins

Axmud plugins are Perl scripts that can modify almost any aspect of Axmud's functionality. Besides writing new tasks, you can add new client commands, modify the main window menu, create new edit windows and much more besides.

The Axmud manual (when it is written) will contain a full description of how to write plugins. This Section contains the briefest possible introduction for experienced Perl users who don't want to wait.

13.1 The '*plugins*' directory

Axmud's data directory (folder), which can be found in your home directory, contains a '*plugins*' sub-directory. Axmud will not interfere with this directory, so we suggest that you store your plugins there.

The '*plugins*' sub-directory contains its own '*help*' directory. If you write new commands and new tasks, you should write help files, too. If Axmud can't find a help file in its own directories, it will try looking in *.../axmud-data/plugins/help*.

The Axmud package comes with some standard plugins; they are not loaded automatically, and can be found in the */amudclient/plugins* directory.

13.2 Loading plugins

Plugins can be loaded when Axmud starts, or on demand.

This is how to add an initial plugin - one that is loaded whenever Axmud starts:

- Open the preferences window using '*;editclient*'
- When the window is open, click on 'Plugins > Page 2'
- There are two 'Add' buttons for adding initial plugins
 - 'Add standard' opens the directory where Axmud's standard plugins are stored
 - 'Add custom' opens the directory where your own plugins should be stored. If you haven't written any plugins, this directory will be empty
- Select a plugin (ending with *.pm*), and click the 'OK' button to add it

If you want to load a plugin immediately, do this:

- In the same preference window, click on 'Plugins > Page 1'
- the 'Add' button to load a plugin that starts whenever Axmud starts
- Alternatively, click on 'Plugins > Page 1' to add a plugin on demand
- As before, there are two 'Load' buttons for loading plugins

13.3 Plugin headers

All Axmud plugins are Perl 5 module files (ending *.pm*).

Each plugin file must begin with a header in a fixed format. If the header doesn't exist or is in the wrong format, the plugin is not loaded.

```
#!/usr/bin/perl
package NAME;
#: Version: VERSION
#: Description: ONE LINE DESCRIPTION
#: Author: AUTHOR'S NAME
#: Copyright: COPYRIGHT MESSAGE
#: Require: AXMUD VERSION
#: Init: STRING
```

Some parts of the header are compulsory, some are optional, and there is flexibility in the order of the lines.

- All headers must begin with the Perl shebang
- After the first line, there can be any number of empty lines, or lines containing ordinary comments
- The package line must appear before the *Version*, *Description*, *Author*, *Copyright*, *Require* and *Init* lines
 - The package NAME must not be the name of a plugin that's already been loaded, or the word 'axmud' itself
- Everything after the package line can appear in any order
 - Duplicate lines replace an earlier one, for example *'#: Author: JRR Tolkien'* replaces an earlier *'#: Author: JK Rowling'*
- The *Version* and *Description* lines are compulsory; the plugin won't load without them
 - VERSION should be in the form 'v1.0.0' / 'V1.0.0' / '1.0.0'. If VERSION is not in this form, the plugin is not loaded
- The *Author*, *Copyright*, *Require* and *Init* lines are optional
 - *Init* lines specify if the plugin should start enabled or disabled when it is loaded
 - STRING should be one of the following words: 'enable', 'disable', 'enabled' or 'disabled'

13.4 Disabling plugins

Once loaded, plugins cannot be un-loaded (but you can, of course, remove a plugin from the list of initial plugins that's loaded every time Axmud starts).

However, plugins can be disabled (to a certain extent) using the 'Disable' button in the client preference window's 'Plugins > Page 1' tab. When a plugin is disabled, any tasks it created are halted and any client commands it created will no longer work.

A disabled plugin can be re-enabled at any time with the 'Enable' button on the same page. Tasks that were halted when the plugin was disabled will not magically re-start themselves; but you can start them manually in the normal way.

13.5 Writing plugins

(This Section assumes you are familiar with Perl object-orientated programming.)

The main Axmud object is an instance of Games::Axmud::Client, stored in the global variable

```
$axmud::CLIENT
```

Note that Axmud uses CAPITAL LETTERS for the small number of global variables, and camel-class nomenclature for everything else.

Sessions are an instance of Games::Axmud::Session, and handle a connection to a single world.

If all sessions share a main window, retrieving the current session - the one that's currently visibly in that main window - is easy:

```
$axmud::CLIENT->currentSession
```

If each session has it's own main window, things are a little trickier. However, any task or client command you write already knows which session it belongs to.

Tasks store their session in a standard instance variable (IV):

```
$self->session
```

If you write a new client command, the bulk of the code will be in the `->do` function. The session is passed an argument whenever that function is called, so you can simply use

```
$session
```

In the Axmud code, all objects inherit a 'generic' object, `Games::Axmud` (found in `.../lib/Games/Axmud.pm`).

This generic object provides a number of methods available to everything. These methods are mostly used for retrieving or modifying values stored as instance variables.

For example, the following call will replace the value of a scalar, list or hash instance variable:

```
$self->ivPoke(IV_NAME, VALUE, VALUE, VALUE...)
```

This call will examine a hash instance variable, and retrieve the value stored as a key-value pair:

```
$value = $self->ivShow(IV_NAME, KEY)
```

Plugins are mostly used to add new client commands, tasks and so on. This is done using calls to methods in the `Games::Axmud::Client` object.

13.5.1 Adding client commands

All client commands inherit from a generic command object, `Games::Axmud::Generic::Cmd` (found in `.../lib/Games/Axmud/Generic.pm`). This object documents a command's IVs.

The code for individual client commands are found in `.../lib/Games/Axmud/Cmd.pm`.

If you write new client commands, they should be in the form

```
Games::Axmud::Cmd::Plugin::MyCommand
```

Once written, the plugin must inform Axmud of the new client command's existence:

```
$axmud::CLIENT->addPluginCmds(  
    $pluginName,  
    'MyCommand',  
    'OtherCommand',  
    ...  
);
```

In the example above, 'MyCommand' must match the last part of the client command package name, *Games::Axmud::Cmd::Plugin::MyCommand*.

13.5.2 Adding tasks

All tasks inherit from a generic task object, *Games::Axmud::Generic::Task* (found in *.../lib/Games/Axmud/Generic.pm*). This object documents a task's IVs.

The code for individual tasks are found in *.../lib/Games/Axmud/Task.pm*.

If you write new tasks, they should be in the form

```
Games::Axmud::Task::MyTask
```

Note that there is no need to add the word *Plugin*, as there is for client commands.

Once written, the plugin must inform Axmud of the new task's existence:

```
$axmud::CLIENT->addPluginTasks(  
    $pluginName,  
    $taskPackage,  
    $taskFormalName,  
    $referenceToTaskLabelList,  
    $taskPackage2,  
    $taskFormalName2,  
    $referenceToTaskLabelList2,  
    # ...  
);
```

14 The desktop

At any time during a typical Axmud session there may be several windows open. Axmud tries to arrange these windows so that they don't overlap (as far as possible). This is generally called 'window tiling'.

Note that **window tiling has not yet been implemented on MS Windows. Windows users should skip Section 14 entirely, as none of it applies to their system.**

Note also that if you're using settings optimised for visually-impaired users, window tiling is also turned off, so none of Section 14 applies to you unless you're planning to turn it back on.

If window tiling is turned off in general and you want to turn it on (or vice-versa), you can use the following commands:

```
;activategrid
;agr

;disactivategrid
;dgr
```

By default, multiple connections can share a single main window, or they can each have their own main window. To switch between the two, use the following command:

```
;togglshare
;tsh
```

For technical reasons, the change won't be applied until you restart Axmud.

14.1 Zonemaps

The blueprint for arranging windows is called a 'zonemap'.

You're free to design your own zonemaps (from the main window, click 'Edit > Axmud Preferences > Workspaces > Page 7'), but most of the time you'll be using one of Axmud's standard zonemaps.

14.2 Workspaces

Axmud supports multiple workspaces, both real and virtual.

By 'real' workspaces, we mean multiple physical monitors. Many modern systems, especially Linux systems, implement 'virtual' workspaces. meaning that the user has a single monitor which can switch between one of several virtual workspaces. Axmud supports setups.

Only one workspace - the one in which Axmud opens - is used by default, but it's easy to set up Axmud to use multiple workspaces. If you use multiple workspaces, you can either use the same zonemap in every workspace, or apply a specific zonemap to each separate workspace.

14.3 Zones

A zonemap consists of one or more 'zones'. A zone can contain one or more windows. If the zone is full of windows, it can refuse to accept more window, or it can stack windows on top of each other.

Some zones are only allowed to contain one type of window. For example, there is normally a zone which contains *only* the main window. Some zones might contain only one type of task window. Other zones can contain any kind of window.

14.4 Standard zonemaps

Axmud provides a number of standard zonemaps.

Some of them are designed for use on modern widescreen (1920x1080) monitors and others are designed for use on smaller monitors.

In addition, some are designed for use when sessions share a single main window and others are designed for use when sessions have their own main window.

When Axmud first ran, you probably selected the most suitable zonemap for your system, but if not, it's easy to change to a different one.

These zonemaps are designed for shared main windows:

- *'single'* - The entire workspace is covered by a single zone. Windows (even main windows) use their default sizes. Windows cannot be stacked on top of each other
- *'single2'* - A modified version of *'single'*, in which windows can be stacked on top of each other
- *'basic'* - Designed for use with small monitors. There are two zones, one large one for the main windows, and a small one for everything else
- *'extended'* - Designed for use with all monitors. The main window and Status/Locator task windows go on the left, everything else goes on the right
- *'widescreen'* - Designed for use with widescreen monitors. The same as *'extended'*, but there is room for more windows on the right (since the main window doesn't need to be so large)

These zonemaps are designed for separate main windows:

- *'horizontal'* - The workspace is divided into two halves, left and right, with each half controlled by a single session. Windows cannot be stacked on top of each other
- *'horizontal2'* - A modified version of *'horizontal'*, in which windows can be stacked on top of each other
- *'vertical'* - The workspace is divided into two halves, top and bottom, with each half controlled by a single session. Windows cannot be stacked on top of each other
- *'vertical2'* - A modified version of *'horizontal'*, in which windows *can* be stacked on top of each other

14.5 Setting zonemaps

When you change zonemaps, you can change the zonemap for all workspaces, or just a single workspace.

In addition, you can specify how Axmud should behave whenever it starts: how many workspaces to use, and which zonemap to use on each workspace.

The easiest way to accomplish all of these tasks is to use the client preference window, so open that window before continuing (from the main window menu, select 'Edit > Axmud Preferences > Workspaces').

14.5.1 Setting the workspace direction

Axmud's default workspace is the one in which it opens. If you need additional workspaces, those workspaces are added in a consistent direction.

To set this direction, click 'Page 1', choose an option from the combo (drop-down) box, and click the 'Set' button next to it.

14.5.2 Setting initial workspaces

Now click on 'Page 4'. This page lists the workspaces Axmud uses when it starts. This list always contains at least one workspace - the one in which Axmud opens.

You can add additional workspaces.

- Select a zonemap in the combo (drop-down) box
- Click 'Add workspace' to add a workspace using that zonemap as its default

You can also modify the default zonemap used by a workspace that's already in the list

- Select a workspace from the list by clicking on it
- Select a zonemap in the combo (drop-down) box
- Click 'Modify workspace'

14.5.3 Modifying current workspaces

You can apply the same kinds of changes to workspaces that Axmud is using right now. These changes won't be retained when Axmud next starts.

Click on 'Page 3'. To add an additional workspace:

- Select a zonemap from the combo (drop-down) box
- Click the 'Use' button

14.5.4 Editing zonemaps

You can add new zonemaps and modify existing ones on 'Page 7'.

- Enter a zonemap name in the box on the left side, next to the 'Name' label
- Click the 'Add' button
- Select the new zonemap in the list
- Click the 'Edit' button
- Click the 'Zone models' tab
- You can create zones on the grid by clicking the top-left and bottom-right corner of the area you want, and then by clicking the 'Add zone model' button
- If you want to modify a zone's properties, for example by putting a limit on the windows it can contain, select the zone by clicking it and then click the 'Edit' button

Standard zonemaps can't be modified by the user, but you *can* create a clone of a standard zonemap, and edit that instead.

- Select an existing zonemap from the list by clicking it
- Enter a name for the clone in the box on the right side, next to the 'Clone' button
- Click the 'Clone' button
- Select the cloned zonemap in the list
- Click the 'Edit' button to modify it

14.6 Layers

Sometimes there isn't enough room on the desktop to arrange all of Axmud's windows without them overlapping each other.

Axmud solves this problem by arranging windows in layers. This isn't obvious at first, because the default layer is the one at the bottom. When that layer is full, new windows are placed in a higher layer.

Windows in higher layers aren't visible. Windows in lower layers might be visible, but will be obscured by windows in the current layer, which are always visible.

To move up and down a layer, use the *pointing hand* buttons near the top of the main window. Alternatively you can use the following client commands:

```
;layerup  
;lup  
  
;layerdown  
;ldn
```

If a zone is allowed to contain only one window, then that window will always be visible. (Many of the standard zonemaps reserve a zone for the exclusive use of the main window).

Edit, preference and wizard windows are considered temporary; it's assumed that you are going to modify some value and close them shortly afterwards. Axmud does not attempt to fit them inside a zone.

14.7 External windows

It's possible to grab an external window - one that is nothing to do with Axmud - and to arrange it on the desktop using the first available space. This is quite useful if you want to watch a video while slaughtering your enemies.

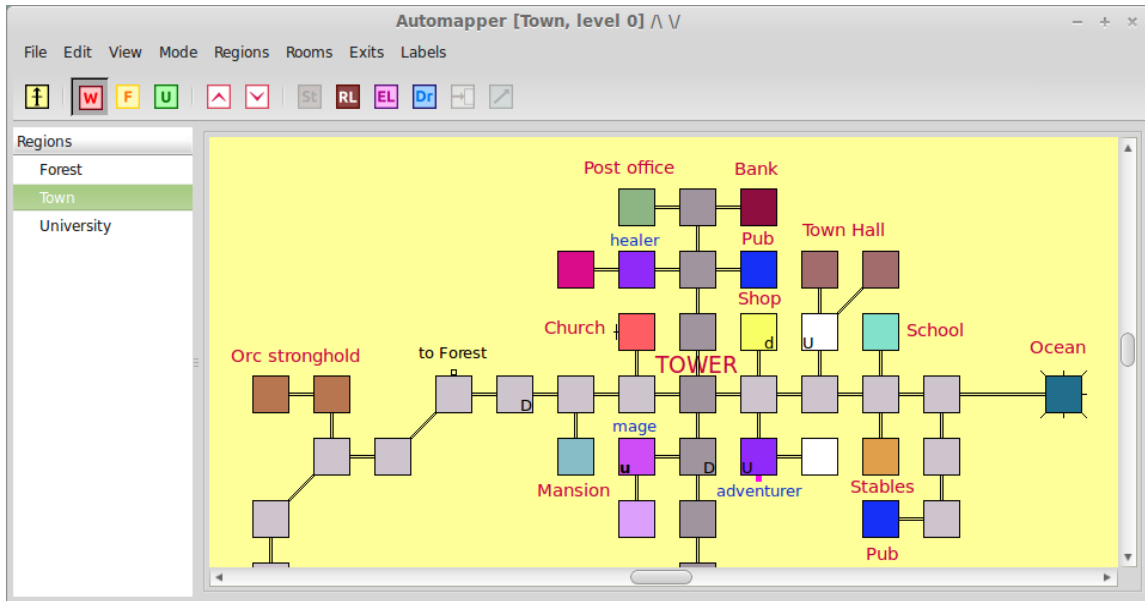
```
;grabwindow vlc  
;gw vlc
```

This commands 'grabs' the first window it finds whose name matches the pattern 'vlc'. The pattern is case-sensitive, so you should use 'vlc', not 'VLC'.

To release all external windows and return them to their previous sizes and positions:

```
;banishwindow  
;bw
```

15 The Automapper window



To open the automapper window, click this button in the main window or use the `';map'` command.

15.1 Introduction

Axmud has an extremely powerful automapper, but it won't work at all if the Locator task can't recognise the character's location (or if the Locator task isn't running at all).

If you're using a pre-configured world, you're good to go: you can start drawing maps immediately. If not, the current world profile needs to be configured first.



Axmud provides a *Locator wizard* which can handle most of the configuration process (with just a little assistance from you). Simply click this button in the main window and follow the instructions.

Hint 1

The wizard has a 100% accuracy rate with some worlds, but struggles to cope with other worlds. If you have problems configuring your favourite MUD, visit us at the Axmud website (the link is at the beginning of this document) and we'll try to help.

Hint 2

The wizard can learn to recognise all the components of a room statement: the room title, the verbose description, the exit list and the contents list. In most cases, the only component you really need is the exit list. If you're having problems, run the wizard again, ignoring everything except the line with the list of exits.

Hint 3

A few worlds (such as LambdaMoo and MUD1) don't use exit lists in their room descriptions. The Locator wizard won't work at these worlds, but the Automapper *can* still be configured correctly.

15.2 Automapper buttons

Near the top of the automapper window is a set of buttons.



These buttons perform many of the same actions that can be performed from the window's menu.

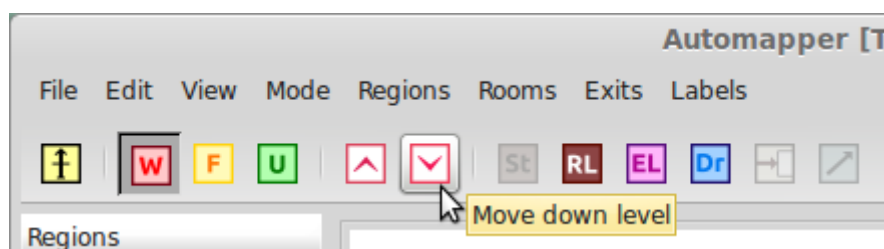
The buttons that can't be clicked represent actions that can't be performed at the moment. When you first open the window, almost *none* of the buttons can be clicked.

There are too many buttons to fit on the window, so they have been divided into sets. The compass button on the left switches between button sets. If you click it once, the next set of buttons will appear:



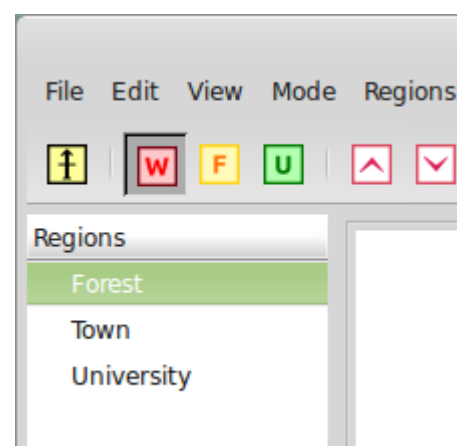
If you keep clicking on the compass button, you'll eventually arrive back at the original button set.

It's not always obvious what an button does. Let your mouse hover an button for a short explanation.



15.3 Creating regions

Worlds can contain many thousands of rooms so it's usually necessary to divide your maps into regions. The area on the left of the window lists all the regions you've created.



The first step in drawing a new map is always to create a region.

- In the automapper window menu, click 'Regions > New region'
- In the dialogue window that appears, give the region a name
 - Each region must have a unique name
 - The usual Axmud naming rules do *not* apply
 - Region names can contain spaces and punctuation
- Don't select a parent region, for now
- Click the 'OK' button to create the region

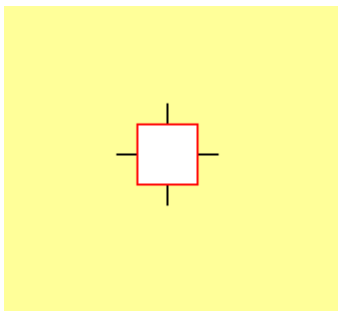
15.4 Creating rooms

Once you've created the first region, the map will turn from white to yellow. You can now add your first room.

There's more than one way to add a room, but this is the simplest way:

- Make sure the Locator task is running and that it's displaying the current room
 - If you need to reset the Locator task, you can click the brown 'RL' button
- Right-click the map and select 'Add first room'

The new room will appear in the middle of the map. It should have the same number of exits as the room displayed in the Locator task's current window.



```
Locator
[#2] Village Road Intersection
You are in the main intersection of the village.
Saquivor road extends north and south, intersected
east to west by a road that leads west toward a
wilderness, and east toward shore.
north, east, south, west
sign
tower great
```

15.5 Automapper modes

The automapper can operate in one of three modes. You can switch between them by clicking on the 'W', 'F' and 'U' buttons.



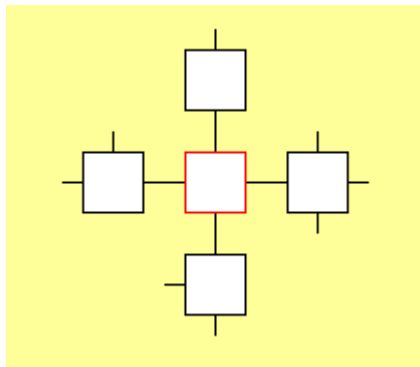
'U' is **'Update mode'**. As your character move around the world, the automapper will automatically draw new rooms and update existing rooms.

'F' is **'Follow mode'**. As your character moves around the world, the automapper tracks your location but doesn't make any significant changes to the map.

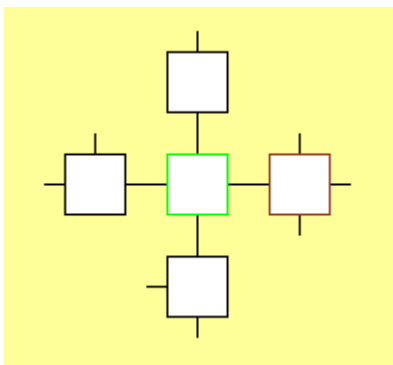
'W' is **'Wait mode'**. The automapper will ignore what your character is doing.

When you first open the automapper window it is in Wait mode. However, if you create a new room by right-clicking the map and selecting 'Add first room', the automapper automatically switches to Update mode.

The current room is always drawn in a different colour. In 'Update mode' it is drawn with a red border.



In other modes, the current room is drawn in a *slightly* different colour. In 'Follow' mode it is drawn with an orange border, and in 'Wait mode' it is drawn with a pink border.



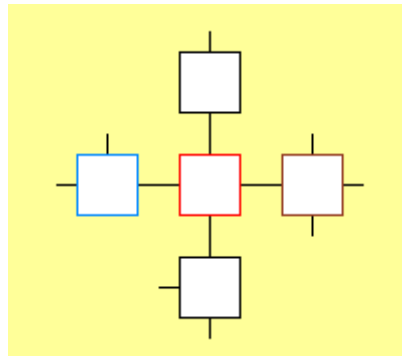
As you move around the world, sometimes the automapper will get lost. This often happens when the current room (captured by the Locator task) isn't the one the automapper was expecting.

When the automapper gets lost, the *previous* current room is drawn with a green border.

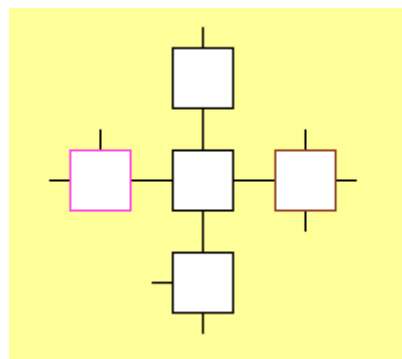
Since there is no longer a current room, the automapper won't modify any of the existing rooms by mistake.

15.6 Selecting rooms

You can select a room by clicking it. A selected room is drawn blue.



If you select the *current* room, it is drawn purple (which represents a mixture of blue and red).



You can unselect a room by clicking on an empty area of the map (or by clicking again on the selected room). You can select multiple rooms by holding down the SHIFT or CTRL keys, and then clicking on the rooms you want to select.

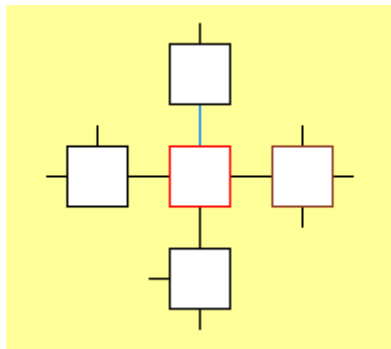
Selecting a room (or rooms) makes some of the menu items (and buttons) available. For example, when there is a single selected room, you can click on the menu item 'Rooms > Edit room'. When there are multiple selected rooms that menu item can't be clicked.

When the automapper gets lost, you can set the current room by right-clicking it and selecting 'Set current room'. The automapper window has a handy button which sets the selected room as the current room.



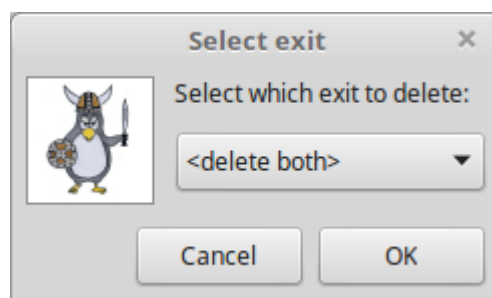
15.7 Selecting exits

You can select an exit by clicking on it or near it. A selected exit is also drawn blue.



Actually, Axmud treats the selected exit as two distinct exits: one leading south from the room at the top and one leading north from the room in the middle.

Therefore, if you right-click on the exit and click 'Delete exit', you'll be asked which of the two exits you want to delete. You can choose to delete one, or the other, or both.

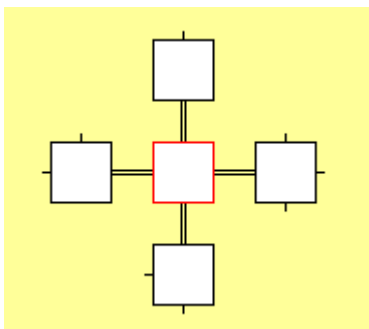


15.8 Simple and complex exits

The automapper displays exits in two different styles. These styles are called 'simple' and 'complex'. You can also choose to display no exits at all.

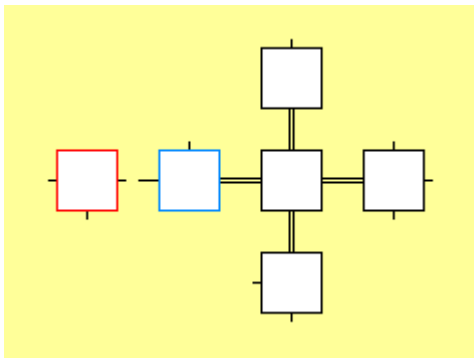
When you first start drawing a map, the automapper uses 'simple' exits.

- In the automapper menu, click 'View > Exits (all regions) > Draw complex exits'
- The map will change to look like this:

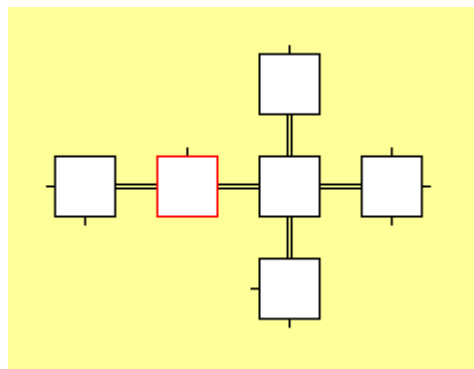


- The double lines represent a 'two-way exit' - that is to say, if you go north from the middle room, you can then go south and arrive back where you started
- The short single lines represent an 'incomplete exit' - one that leads to an unknown destination

Now, look what happens if we go west, twice:



- The west exit from the selected (blue) room is an 'uncertain' exit - we know that we can travel west to arrive at the current (red) room, but we don't yet know if we can then travel east to arrive back where we started
- The corresponding exit from the current (red) room is another 'incomplete' exit



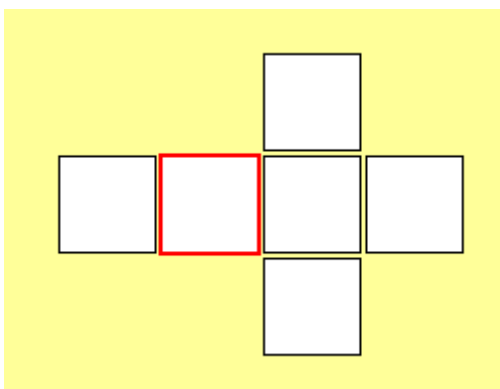
- If you now move east, the automapper will redraw the exit as a 'two-way' exit.
- You can easily switch between 'simple' and 'complex' exits. Switching styles has no effect on the map itself; the only thing that changes is the way the map is drawn

15.8.1 Drawing regions without exits

Some regions might have a thousand or more rooms, with perhaps ten of thousands of exits. Maps like these cannot be drawn instantaneously; in fact, on some slower computers, the drawing process might take a few seconds.

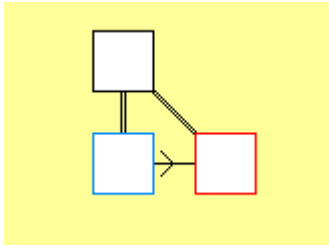
Besides drawing simple and complex exits, you can also opt to draw no exits at all.

- In the automapper menu, click 'View > Exits (all regions) > Use region exit settings'
 - You can now set each region's settings individually
- Select a region that has many rooms
- Click 'View > Exits (current region) > Draw no exits'



- In this mode, it is sometimes difficult to see the current room
- If this is a problem, try clicking 'View > Draw current room > Draw emphasised room'
- If that's still not comfortable, you can opt to fill in the whole box by clicking 'View > Draw current room > Draw filled-in room'

15.9 One-way exits



Here we can see a one-way exit. The arrow tells us that we can travel east from the selected (blue) room to the current (red) room, but that we can't travel west to get back to where we started.

This usually happens when the current (red) room doesn't have a 'west' exit at all. However, you can convert an existing exit into a one-way exit:

- Right-click on the exit
- Click 'Set exit type > Set one-way > Mark exit as one-way'

15.10 Connecting rooms

It's easy to connect an exit to a room.

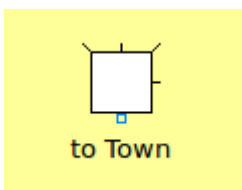
- Right-click an exit and choose 'Connect to click'
- Click on the destination room

Sometimes it can be tricky to click on the right exit, but there are always alternative methods.

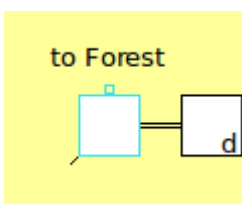
- Right-click on a room and choose 'Select exit'
- In the dialogue window, choose an exit and click the 'OK' button

There is also a handy connect-to-click button. The procedure is the same as before:

- Select an exit (left-click on it, or use the method described just above)
- Click the button
- Click on the destination room



- If the destination room is in a different region, the exit is drawn as a 'region exit' (an empty square).
- Click the region exit to select it



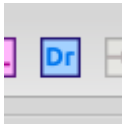
- The exit's destination room (in another region) is now drawn with a cyan border

15.11 Moving rooms

Rooms can be moved to another part of the map or even to a different region altogether.

First select the rooms you want to move. Hold down the SHIFT or CTRL key to select multiple rooms. Then hold down the ALT-GR key (it's just to the right of the space bar) and simply drag-and-drop the room to its new position.

A few keyboards don't have such a key, in which case you can use the 'Drag mode' button.

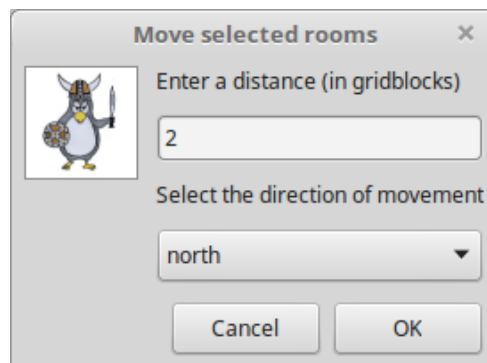


- Click the button to turn on 'Drag mode'
- Drag the room(s) to their new positions
- Click the button again to turn off 'Drag mode'

15.11.1 Moving multiple rooms

There are two options for moving multiple rooms. Here is the first one.

- Hold down the SHIFT or CTRL keys, and select multiple rooms
- From the menu, select 'Edit > Move selected rooms...'
- In the dialogue window that appears, enter a distance and a direction



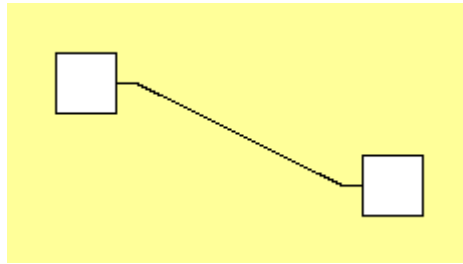
The second option allows you to move a room (or rooms) to another region entirely.

- Hold down the SHIFT or CTRL keys, and select multiple rooms
- From the menu, select 'Edit > Move selected rooms to click'
 - Alternatively, click the blue arrow button near the top of the automapper window
- Select a new region
- Click an empty area of the map in that region

When you move rooms around, exits are automatically re-drawn. If necessary, they are re-drawn as region exits.

15.12 Bent exits

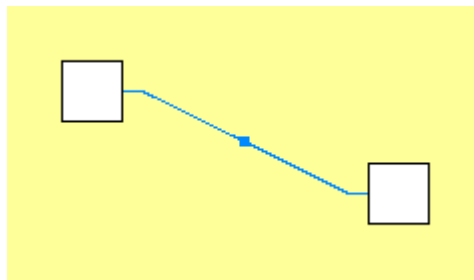
While moving rooms around the map, you'll already have noticed that exits are able to bend.



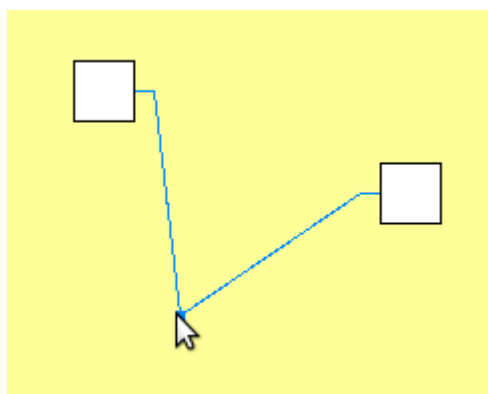
If an exit is re-drawn in an inconvenient way - overlapping a room, perhaps - you can add additional bends to the exit:

- Right click on the exit at the point where you want to add the bend
- In the pop-up menu that appears, choose 'Add bend'

Bends are usually invisible, but are highlighted when the exit is selected.



The exit bend can be dragged around the map, just like any other object (hold down ALT-GR, or click the 'Drag mode' button near the top of the automapper window).

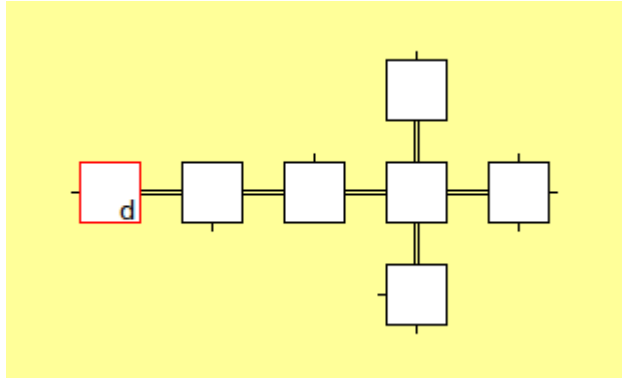


If you remove the bends, the exit will be automatically re-drawn in the original way.

- Right-click the exit near the bend you want to remove
- In the pop-up menu that appears, choose 'Remove bend'

15.13 Up and down

The exits 'up' and 'down' are drawn as the letters 'u' and 'd' inside the room:

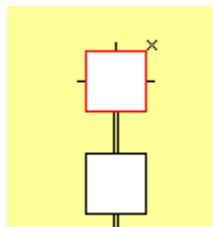


This 'down' exit is actually an incomplete exit (whose destination is unknown), because the 'd' has been drawn as a lower-case 'd'. A two-way exit would be drawn as an upper-case 'D'.

15.14 Non-primary directions

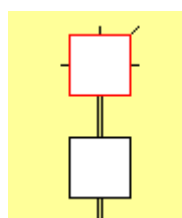
Axmud's so-called 'primary directions' are the sixteen cardinal directions ('north', 'southeast', 'northnorthwest' etc) plus 'up' and 'down'. (Most worlds don't use directions like 'northnorthwest', but they are available if you want them.)

Exits in other directions (such as 'enter', 'out' or 'climb wall') are initially drawn as an 'x'.



These exits should be assigned a primary direction before they are used. There are two ways of doing this:

- Right-click on the exit and choose 'Allocate map direction > Choose direction...'
- In the dialogue window that appears, choose one of the available primary directions, such as 'northeast'
- Click the 'OK' button. The exit is redrawn

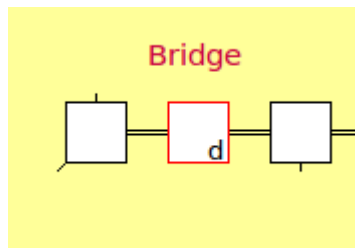


The second method is useful for rooms which have two exits leading to the same destination room - one of them a primary direction like 'east', the other a non-primary direction like 'enter pub'. (Dead Souls, one of Axmud's pre-configured worlds, has a few rooms like this.)

- Right-click on the exit and choose 'Allocate shadow...'
- In the dialogue window that appears, select the east exit
- Click the 'OK' button. The 'east' and 'enter pub' exits are now drawn as a single exit
 - The 'east' exit is called the 'enter pub' exit's 'shadow exit'

15.15 Labels

Besides rooms and exits, you can also draw labels on your maps.

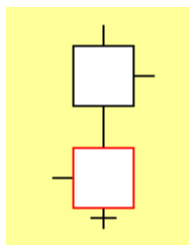


- Right-click on the map and choose 'Add label here'
- In the dialogue window that appears, enter any text
- Click the 'OK' button. The label appears at the position you originally clicked

Labels can be selected by clicking on them (like rooms, they are drawn blue when selected.) Labels can also be right-clicked or dragged around the map, in exactly the same way as rooms (see Section 15.11)

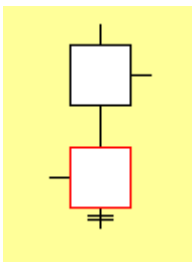
15.16 Exit ornaments

When your character tries to move south but walks into a closed door, the automapper redraws the exit (assuming that the current world has been configured to recognise closed doors).



An exit with a single line through it is some kind of door that can be opened (and presumably closed). The extra line is called an 'exit ornament'.

A locked door is drawn with *two* lines through it.

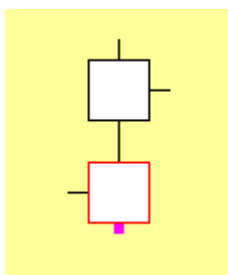


You can add an ornament to an exit manually.

- Right-click on the exit and choose 'Set ornaments > Pickable exit'

An exit which can be picked (by a thief, using a lockpick) is drawn as an empty rectangle. A breakable exit (representing a locked door that can be broken down by brute force) is drawn as a letter I, perpendicular to the exit.

An impassable exit (representing an exit that can't actually be used, at least not by ordinary players) is drawn as purple square.



An exit that has an exit ornament can be reset, so that it has no ornament.

- Right-click on the exit and choose 'Set ornaments > No ornament'

15.16.1 Twin exit ornaments

Two-way exits are normally drawn as a single line (or a parallel pair of lines, in 'complex exits' mode). This actually represents two exits: one leading north, perhaps, and one leading south back to where you started. The south exit is called the 'twin exit' of the north exit.

Axmud usually adds an exit ornament for both an exit *and* its twin at the same time. If this behaviour isn't what you want, you can turn it off.

- Select any exit on the map
- In the automapper menu, de-select 'Exits > Set ornaments > Also set twin exits'

15.17 Assisted moves

Section 15.14 describes how to draw an exit like 'out' as a primary direction like 'north' or 'southeast'.

The automapper has a function called 'assisted moves'. If you're in a room with only one exit - 'out' - and if this room is drawn as 'east', you can leave the room by typing *either* 'out' or 'east'.

Unfortunately, because of technical limitations, assisted moves only apply a few exits at a time. You're safe to assume that these command sequences will have the same effect:

```
north;north;out
north;north;east
```

...but the following pair of command sequences might not have the same effect, 100% of the time:

```
n;n;n;e;e;e;ne;nw;ne;e;e;se;e;se;s;s;sw;e;se;u;u;u;s;n;out
n;n;n;e;e;e;ne;nw;ne;e;e;se;e;se;s;s;sw;e;se;u;u;u;s;n;east
```

Assisted moves can be turned off, if they're an annoyance.

- From the automapper menu, de-select 'Mode > Assisted moves > Allow assisted moves'

Assisted moves are not available when 'redirect mode' is turned on (see Section 7.5.6).

15.17.1 Assisted moves using exit ornaments

Section 15.16 describes how to modify an exit to mark it as a door, a locked door, an impassable exit, and so on. Modified exits are drawn with an exit ornament - for example, an exit with a door is drawn with a line through it.

When you try to move through an exit that has (for example) an openable door, the automapper will try to intercept the world command:

```
east
```

...and convert it into a command sequence that will prevent your character from bumping into the door:

```
open east door;east
```

If this an annoyance, you can turn it off.

- From the automapper menu, de-select 'Mode > Assisted moves > Open doors before move'

You can also turn on 'Protected moves' mode, which blocks any world command which would move the character through an impassable or non-existent exit.

- Select 'Mode > Assisted moves > Allow protected moves'

15.17.2 Custom assisted moves

Sometimes the generic assisted move will not work - for example, it might not be possible to move through the exit by using the usual 'open east door' command.

If so, you can specify the precise commands to use in an assisted move.

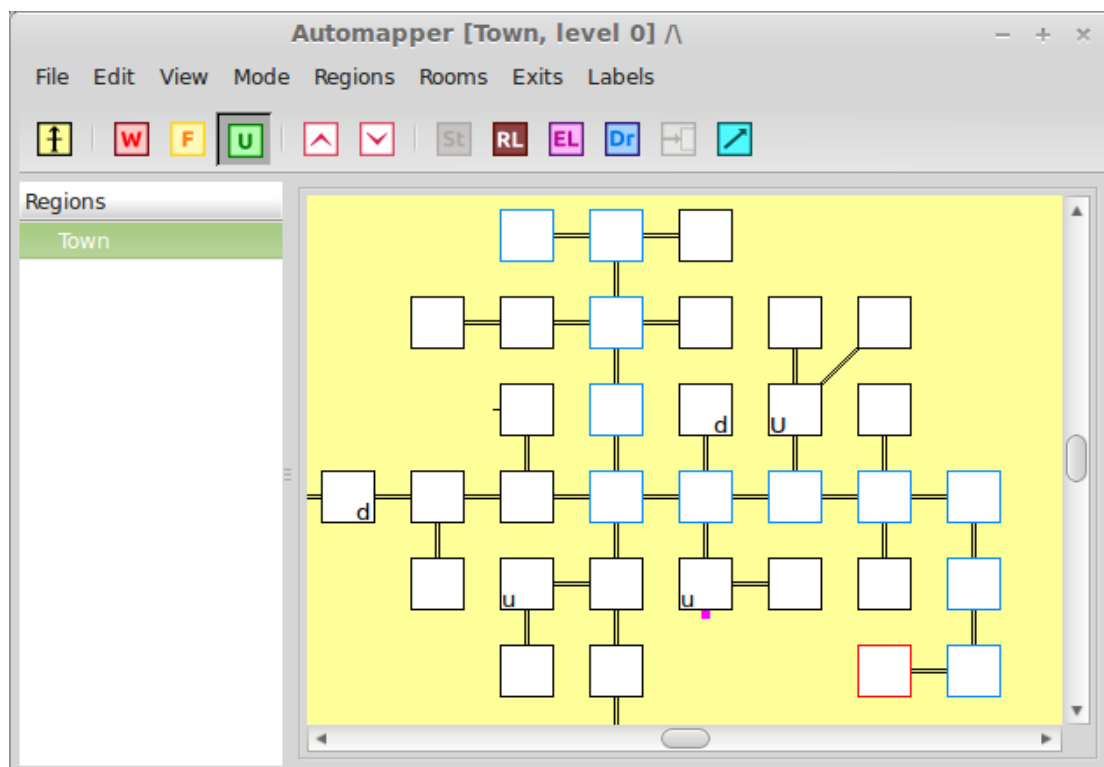
- Right-click on an exit and choose 'Set assisted move...'
- In the dialogue window that appears, enter a new command sequence at the bottom
 - For example, 'knock on door;north'
- Click on 'OK' to store the new assisted move

A custom assisted move of this kind will be used instead of the normal one. In other words, the automapper will not try to insert an 'open door' command while moving through this exit.

15.18 Pathfinding

The automapper can find the shortest path between two rooms virtually instantaneously, even if it means crossing dozens of different regions.

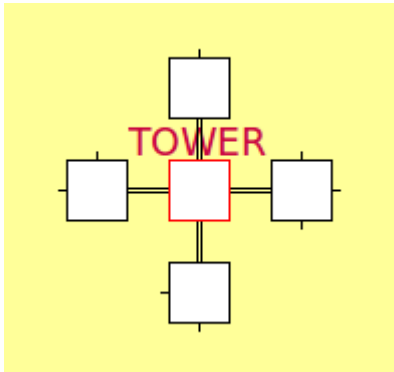
Double-click on a room to go there via the shortest possible path. (You can also right-click the destination room and select 'Go to room'.)



All rooms along the path are selected so that you can clearly see the route. When you arrive, click on an empty area of the map to de-select them all.

15.19 Room tags

A good way to mark important rooms is with a room tag. Rooms tags belong to a specific room.



- Right-click the parent room and choose 'Set room text > Set room tag...'
- In the dialogue window that appears, enter a room tag
- Short room tags are better - no more than five or six letters, ideally

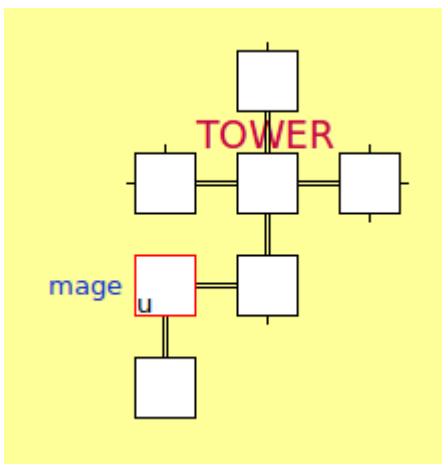
To help distinguish them from labels, room tags are drawn larger, and usually in CAPITAL LETTERS. (This behaviour can be turned off, if desired.)

Like all other objects, room tags can be selected and dragged around the map. If, by any chance, you lose track of which room tag belongs to which room:

- Right click on the room tag and choose 'Reset position'

15.20 Room guilds

Rooms that belong to a guild can be marked some text that's drawn slightly smaller than a label, and in a different colour.



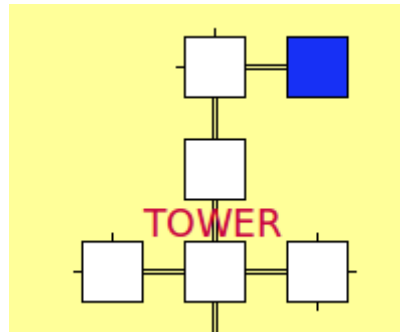
- Right-click on a room and choose 'Select room text > Select room guild...'
- In the dialogue window that appears, enter the room's guild

If you haven't added any guild profiles yet (see Section 5.2), then of course it won't be possible to set a room guild.

The room guild text can be selected, dragged around the map, right-clicked and reset, just like room tags.

15.21 Room flags

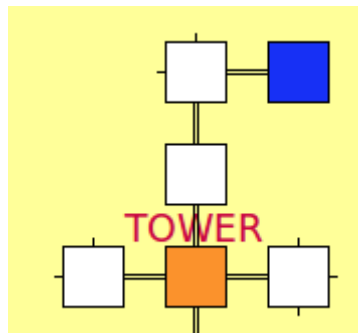
The colour of a room can be changed to show what kind of room it is. For example, all pubs can be drawn in blue.



Every room has dozens of 'room flags' which can be turned 'on' or 'off'. When a room is first created, all of its room flags are 'off'.

If we turn the room flag for pubs on, the room turns blue; if we turn the flag off, the room turns back to white.

Another example is the room flag which means 'this room is the centre of the world'. When this flag is 'on', the room is drawn orange.



Each room can have more than one flag 'on'. If a room is both a pub and the centre of the world, Axmud will choose one colour or the other.

In fact, there is a standard list of room flags whose order doesn't often change. If a room has several room flags 'on', the flag that's highest on the list is the one that is drawn. (As it happens, the room flag which means 'this room is the centre of the world' is above the room flag which means 'this room is a pub' - so the room would be drawn orange, not blue.)

15.21.1 Room flag groups

Room flags are divided into groups. These groups are:

- Markers - rooms that are important, dangerous, or need to be examined more closely
- Navigation - rooms at the centre of the world, or located on routes; rooms that contain signposts, vehicles or require which swimming or flying
- Commercial - rooms where you can buy or sell things
- Buildings - rooms that are other kinds of building
- Structures - rooms that are gates, walls, towers, tunnels, bridges or other structures
- Terrain - rooms that are in a forest, in a desert, in a swamp (and so on)
- Objects - rooms that contain notable objects
- Light - rooms that are outside, inside, underground or dark
- Guilds - rooms that are inside a guild area
- Quests - rooms that are important for a certain quests
- Attacks - rooms where fighting isn't allowed or where one of your characters has died

Room flags from any of these groups can be used to colour a room.

15.21.2 Releasing room flag filters

A useful automapper feature is to show room flags from one (or more) of the groups, ignoring all the rest. For example, you might decide that you only want to see room flags from the 'buildings' group or from the 'terrain' group. You might also decide that you want to see room flags from all groups *except* the 'markers' group.

To stop showing a particular group, you apply a filter for that group. This action *filters out* room flags from the group, allowing all other room flags to pass through.

By default, *all* filters are released. The first step, then, is to apply them all.

- In the automapper menu, de-select 'View > Room filters > Release all filters'

All filters are now applied, and no rooms are coloured in. You can now choose one (or more) filter to release.

- To colour all pubs blue, select 'View > Room filters > Release commercial filter'
- To colour the centre of the world orange, select 'View > Room filters > Release navigation filter'

15.21.3 Applying room tags

To mark a room as a pub - that is to say, to turn on the room's 'pub' room flag:

- Right-click the room and choose 'Toggle room flags > Commercial > Room is a pub'
- Repeat this step to turn the room flag off again

To turn on the room flag that means 'this room is the centre of the world':

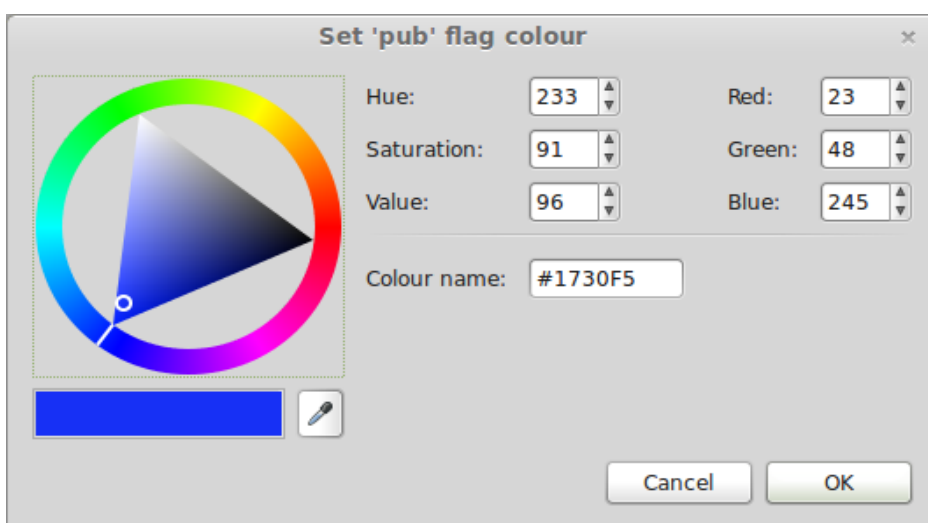
- Right-click the room and choose 'Toggle room flags > Navigation > Room is centre of world'
- Repeat this step to turn the room flag off again

Of course, the rooms will only be coloured blue and orange if the 'navigation' and 'commercial' filters have been released.

15.21.4 Editing room flag colours

If you don't like the colour the automapper is using for a particular flag, you can easily change it.

- In the automapper menu, click 'Edit > Edit world model...'
- When the edit window is open, click the 'Room flags' tab
- Scroll through the list until you find the 'pub' room flag
- Click the line to select it
- Click the 'Change' button
- When the dialogue window appears, select a new colour
- Click the 'OK' buttons in both windows to start using the new colour



15.22 The painter

While exploring a region of the world - a forest, perhaps - it would be quite tedious to set the room flags for every new room individually. The solution is to use the automapper's 'painter'.

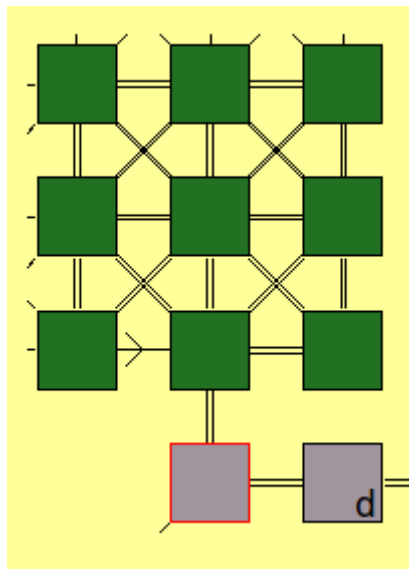
The first step is to turn the painter on:

- From the automapper menu, select 'Mode > Painter > Painter enabled'

Now we can ask the painter to 'paint' each new room green, using the 'room is in a forest/wood' room flag:

- From the automapper menu, choose 'Mode > Painter > Edit painter'
- Wait for the edit window to open
 - There are several tabs, but we're only interested in the first one
- Find the combo (drop-down) box near the bottom of the window, and select the 'forest' room flag
- Click the 'Use' button to add the room flag to the list above
- Click the 'OK' button to close the window

Now, as you move around the forest, every new room will be drawn green.



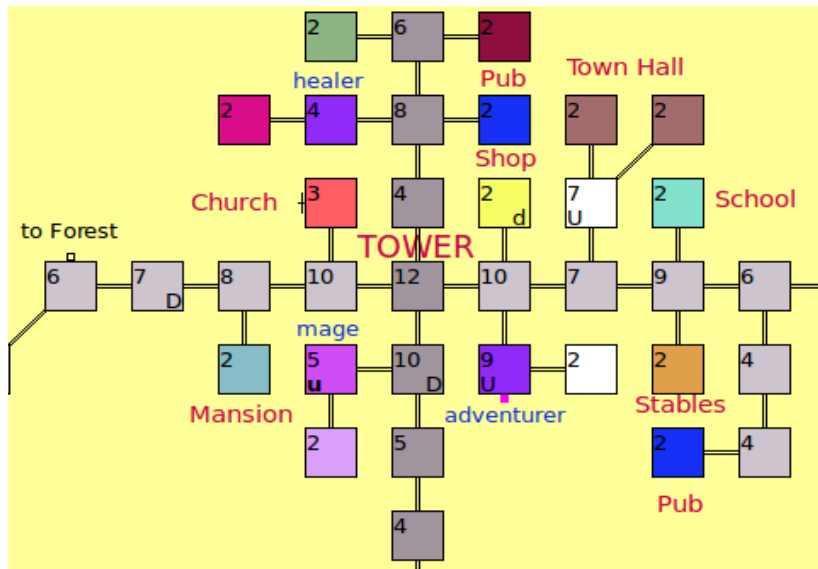
To turn off the painter, de-select 'Mode > Painter > Painter enabled'.

The painter remembers its settings, so the next time you enable the painter, it will resume painting new rooms green.

15.23 Room interiors

The automapper can show statistics about each room. One useless but entertaining feature is to show how many times a room has been visited by the current character.

- In the automapper menu, select 'View > Room interiors > Draw character visits'



15.24 Updating maps

As your character moves around the world, the automapper adds new rooms to the map (but only when it is in Update mode).

Details about each new room are stored. In particular, the room's title, verbose description and list of exits are all stored, if they are available.

For existing rooms, the title and verbose description are updated, if they have changed since the character's last visit.

If you don't want to store (or update) all of these components, you can change the default behaviour. For example, to prevent the automapper from storing (or updating) verbose room descriptions:

- From the automapper window, de-select 'Mode > Update rooms > Update room descriptions'

15.24.1 Matching rooms

When your character moves to an existing room, the automapper compares the room on the map to the room captured by the Locator task. If it's not the same room, the automapper decides that it is now lost.

By default, only the room's exits are checked: if the room on the map has the same exits as the room captured by the Locator's task, the automapper considers that the rooms match.

If you want the automapper to be more fussy - for example, if you want the rooms to have the same room title, as well as the same exits - you can change the default behaviour.

- From the automapper window, select 'Mode > Match rooms > Match room titles'

If you de-select all of the items in the 'Mode > Match rooms' menu, the automapper will never get lost. (This is not recommended, *especially* when the automapper is in 'update mode'.)

When comparing verbose descriptions, only the first 100 characters are compared by default. You can increase or reduce that number, if you want.

- From the automapper menu, choose 'Mode > Verbose characters...'

15.24.2 Hidden exits

Some rooms have an exit which is visible some of the time, but not visible at other times.

This can confuse the automapper, which typically expects the Locator task's captured room to have exactly the same exits as a room on the map.

The solution is to mark the exit as a hidden exit:

- Right-click the exit and choose 'Set exit type > Set hidden > Mark exit hidden'

When comparing rooms, the automapper will ignore any exits that are marked hidden. For example, if you draw a room with a normal north exit and a hidden south exit, it will match both of these rooms:

This is a cobblestone road, leading south.
Obvious exits: north, south

This is a cobblestone road, leading south.
Obvious exits: north

Some rooms have exits which are *never* visible in the room's exit list. You can add these kinds of exit to the map, as well.

- Right-click a room and choose 'Add exit > Add hidden exit...'
- A dialogue window appears. In the combo (drop-down) box at the top, select a direction like 'south'
 - The entry box just below will be set to the same value
- Click the 'OK' button to add the exit
 - 'Hidden' exits are always visible on the map

15.25 Backing up maps

It's a good idea to make backup copies of your maps from time to time.

Actually, maps are stored in a 'world model', and it's this file which needs to be backed up.

- From the automapper menu, click 'File > Save/export world model...'
- If the world model needs to be saved, you'll be asked if you want to save it; in most cases you should click 'Yes'
 - If you click 'No', you'll be backing up the world model as it was, the last time you saved it
- A dialogue window appears. Decide where you want to save the back-up file
 - You can also change the file's name, if you want
 - It goes without saying that you shouldn't change the .tgz part

When you're ready to import the back-up file, follow this procedure. (The data in the back-up file will replace the data currently stored in memory, **and** the world model file that was most recently saved.)

- From the automapper menu, click 'File > Import/load world model...'
- In the dialogue window that appears, select the file you saved earlier
- Then click the 'OK' button

15.25.1 Sharing maps

If you've worked very hard on your maps, you might like to claim some credit before sharing them with other people.

- From the automapper menu, click 'Edit > Edit world model...'
- An edit window will appear showing the 'General' tab
- Enter any text you like into the 'Author', 'Date', 'Version' and 'Description' boxes
- Click the 'OK' button to close the window

You can now share your maps with other Axmud users.

- Create a back-up of the world model, as described above
- Share the back-up file with your friends
- They can load the map by importing the back-up file, as described above

15.26 Miscellaneous features

15.26.1 Opening the automapper window automatically

If you use the automapper regularly, it's convenient to set the window to open every time you connect to the world.

- From the automapper menu, select 'Mode > Start-up flags > Open automapper on startup'

15.26.2 Creating temporary regions

If you want to find your way around a random area - a maze that resets every time someone enters it, perhaps - you can use a temporary region.

- From the automapper menu, click 'Regions > New temporary region...'
- In the dialogue window that appears, add a name like 'temp'
- Click the 'OK' button to create the temporary region
- Right-click on the empty map and choose 'Add first room'

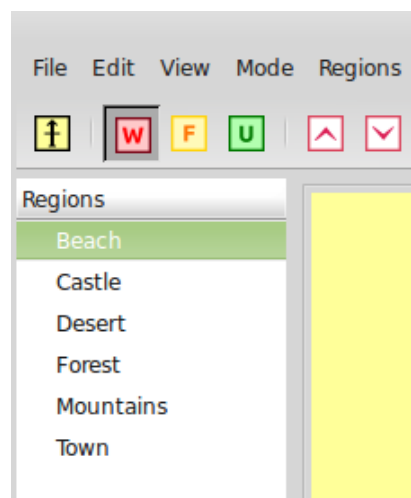
Temporary regions are deleted at the end of the session. (Actually, this is not quite true: they are deleted the next time you connect to the world, at the beginning of the session, before the automapper window opens.)

However, you can delete temporary regions at any time. If you want to delete *all* the temporary regions you've created:

- From the automapper menu, click 'Regions > Delete temporary regions'

15.26.3 Organising regions

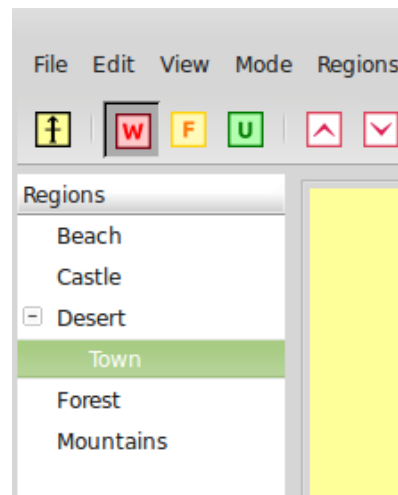
If your map has a lot of regions it's possible to organise them into groups.



For example, if the town is in the middle of the desert, you can make the desert the 'parent' of the town region.

- In the region list, double-click on 'Town' to make it the current region
- From the automapper menu, click 'Regions > Edit region features > Change parent'
- In the dialogue window that appears, select 'Desert'
- Click the OK button to close the window

The list of regions will now look like this:



- Click on the minus (-) sign, next to Desert, to hide its child regions
- Click on the new plus (+) to reveal the Desert's child regions

15.26.4 Favourite regions

The region list is normally in alphabetical order. If one region is at the centre of the world, or if you spend most of your time in a particular region, you can move it to the top of the list.

- In the region list, double-click on 'Mountains' to make it the current region
- From the automapper window, click 'View > Window components > Move current region to top'

15.26.5 Running scripts

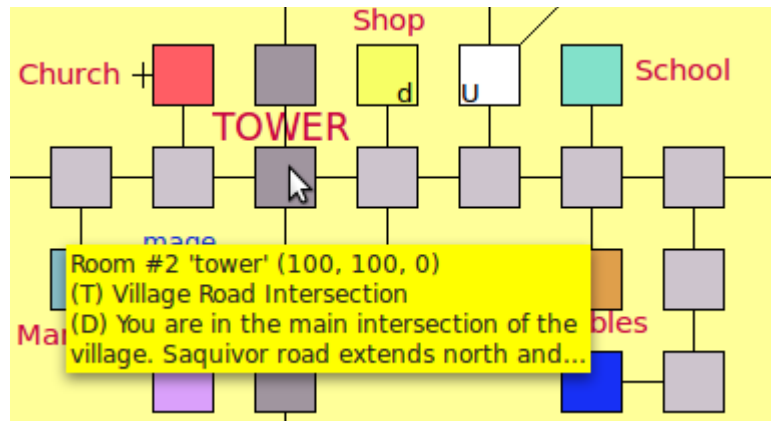
You can specify an Axbasic script that should be run, every time your character enters a particular room.

- Right-click the room and choose 'Edit room...'
- When the edit window is open, click on 'Room > Page 1'
- Enter the name of one or more Axbasic scripts in the box at the bottom of the window

15.26.6 Tooltips

If you want to find out more about a room, one method is to open its edit window. Exits also have their own edit windows. (Those edit windows can be opened from the automapper window's menus.)

A more convenient method is to let your mouse hover above the room (or exit). After a moment, Axmud will display tooltips for the object.



If this becomes annoying, automapper tooltips can be disabled.

- From the automapper menu, de-select 'Mode > Other flags > Show tooltips'

15.26.7 Using the automapper without a window

The automapper actually consists of three separate components - the 'automapper object', which tracks the character's current position, the 'world model', which stores the data, and the automapper window itself, which draws rooms, exits and labels.

As a result, the automapper *object* can track your character's position, even when the automapper *window* is closed.

- Open the automapper window by clicking the button in the main window
- Make sure the automapper window is in 'Follow mode' or 'Update mode'
 - Click on the yellow 'F' or the green 'U' buttons near the top of the window
- Right-click the current room and click 'Set current room'
- Close the automapper window and move around the world
- The character's position on the map is displayed in the Locator task window

Even though the automapper *window* is closed, the automapper *object* is still tracking your character's position. You can re-open the automapper window at any time; it should display the correct current room.

If you want Axmud to forget your character's position when you close the window, you can do so:

- From the automapper menu, de-select 'Mode > Start-up flags > Follow character after closing'

You can also set your character's position *without* opening the automapper window at all.

Every room on the map has a unique number. If you happen to know the current room's number, you can use it in a `'setroom'` command.

```
;setroom 5961  
;srm 5961
```

However, a much better approach is to use room tags (see Section 15.19). If you have marked the important rooms on your map with a memorable room tag, simply navigate to the nearest tagged room, and then:

```
;setroom tower  
;srm tower
```

You can add a switch to set the automapper window's mode. Use `-f` to put the automapper into 'follow' mode, and `-u` to put it into 'update' mode.

```
;setroom -f 5961  
;setroom -u tower
```

If you want to stop tracking your character's position, without opening the automapper window, you can use the `'resetroom'` command.

```
;resetroom  
;rrm
```

15.26.8 Using unusual exits

Cryosphere, which is one of Axmud's pre-configured worlds, uses the standard north-south west-east directions, but some zones in the game use a fore-aft port-starboard system. In addition, areas of the game set on a radial space station use a hubwards-rimwards anticlockwise-clockwise system.

If you're using the pre-configured world profile, exploring this game with the Axmud automapper should be easy. You can move your character through a 'port' exit using *any* of the following commands:

```
west  
w  
port  
p
```

However, to move your character through a 'starboard' exit, there are only three available commands that the automapper will recognise:

```
east
e
starboard
```

This is due to the fact that you can abbreviate 'port' with 'p', but you can't abbreviate 'starboard' with 's', because the abbreviation 's' has already been allocated to 'south'.

Here's a brief explanation of how to configure the automapper for games that use directions like these.

Axmud uses a standard set of directions which are called 'primary' directions. The primary directions are:

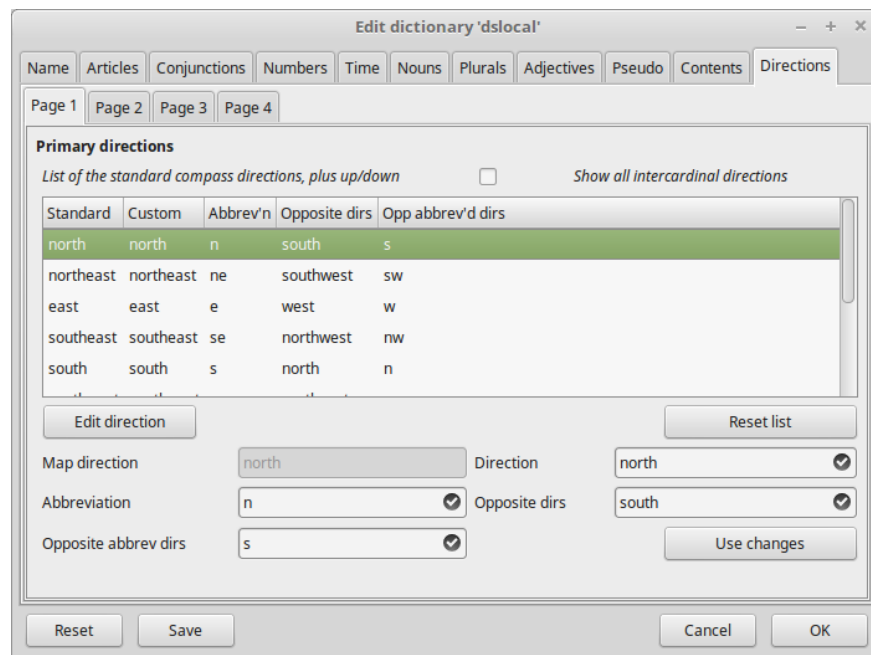
```
north south west east / northwest southwest southeast
northeast / northnortheast eastnortheast eastsoutheast
southsoutheast southsouthwest westsouthwest westnorthwest
northnorthwest / up down
```

Most worlds use these very same directions, in which case there's no need to configure anything. However, if a world uses a different set of directions *all the time* - for example, when the language of the game isn't English - we can customise each primary direction.

To set custom primary directions, first open the current dictionary's edit window:

```
;editdictionary
;edy
```

In the new window, click on the 'Directions' tab. You'll see a list of primary directions.



Very few games use 'northnorthwest', etc, so they aren't listed in this window by default. Such directions are also not listed in most automapper dialogue windows by default.

Even though they aren't listed, directions like 'northnorthwest' are always available.

- In this window, you can click the checkbox near the top of the window ('Show all intercardinal directions').
- In the automapper window, click on the menu item 'Mode > Other flags > Show all directions in dialogues'. This setting will apply to most automapper dialogue windows.

15.26.9 Primary directions in other languages

Axmud knows how to set custom primary directions for about a dozen languages.

- In the edit window you just opened, click the 'Name' tab
- Use the combo (drop-down) box to select a different language
- Click the 'Switch language' button

These languages can also be set when using the Locator wizard (see Section 15.1).

If Axmud doesn't know about a language you can set the custom primary directions yourself. This would be the procedure to set up a French-language dictionary, if Axmud didn't already know how.

- In the same edit window, click the 'Directions' tab
- In the direction list, click 'north', then click the 'Edit direction' button
- The boxes below that button will be automatically updated
- In the 'Direction' box, change the text to 'nord' (the French word for north)
- In the 'Abbreviation' box, the text is already set to 'n'
- In the 'Opposite dir[ection]s' box, change the text to the French word for south, 'sud'
- In the 'Opposite abbrev[iated] dir[ection]s' box, the text is already set to 's'
- Click on the 'Use changes' button to update the list
- Repeat this process for the remaining primary directions
- When you're finished, click the 'Save' button at the bottom of the window

By the way, Axmud continues to use a set of *standard* primary directions - the English words 'north', 'southeast', 'up' and so on - internally, regardless of which *custom* primary directions you set. (You will probably only notice this if you start poking around deep inside Axmud's edit windows.)

15.26.10 Secondary directions

Axmud uses exactly eighteen primary directions - the ones described in Section 15.26.8 - but it also uses a set of 'secondary' directions.

Secondary directions help Axmud to distinguish between things that are exits and things which aren't exits. The default set of secondary directions are:

```
in / out
entrance / exit
steps / lift
gate / gateway / door / doorway
hole / gap / crack / opening
portal / vortex
```

Secondary directions also help Axmud to work out which exits should be drawn facing each other. As an example, if your character moves through a 'hole' exit and you find yourself in a room with the exits 'north' and 'out', it's a good bet that the 'out' exit will lead back to the original room.

You can also use secondary directions for worlds like *Cryopshere*, by adding 'port' and 'starboard' as secondary directions, and by specifying that the automapper should draw them as 'west' and 'east' exits respectively.

You can modify the list of secondary directions in any way you please. (The only rule is that you can't add a word that's already being used as a primary direction.)

It's not necessary to add *every possible word* to your list of secondary directions. If Axmud encounters an exit called 'cupboard', this exit will be (temporarily) drawn in the first available direction (see Section 15.14). In fact, most users will never need to modify any of the secondary directions.

(The previous Section demonstrated how you could change Axmud's custom primary directions, which are English words, into a set of French words at the click of a button. Doing so will not affect this list of secondary directions, which must be modified separately.)

15.26.11 Secondary directions in *Cryosphere*

To set custom secondary directions, first open the current dictionary's edit window:

```
;editdictionary  
;edy
```

In the new window, click on 'Directions > Page 2' tab. You'll see a list of secondary directions.

Cryosphere uses the standard north-south west-east directions in some zones, but in other zones it uses fore-aft port-starboard. This is how to add them to the dictionary, so that 'fore' is treated as the equivalent of 'north', 'aft' is treated as 'south', and so on. (Once again, if you are using the pre-configured world profile to *Cryosphere*, this has already been done for you.)

- Click on the 'Add/Edit direction' button
- In the 'Direction' box, change the text to 'fore' (the equivalent of north in some game zones)
- In the 'Abbreviation' box, change the text to 'f' (the game uses 'f' as an abbreviation of 'fore')
- In the 'Opposite dir[ection]s' box, change the text to 'aft' (the equivalent of south in some game zones)
- In the 'Opposite abbrev[iated] dir[ection]s' box, change the text to 'a' (the game uses 'a' as an abbreviation of 'aft')
- In the 'Auto-allocate to' combo box, select 'north', so that the automapper can automatically draw 'fore' exits in the northerly direction
- Click the 'Use changes' button
- Repeat the process to add 'aft' (south), 'port' (west) and 'starboard' (east).
- When you're finished, click the 'Save' button at the bottom of the window

Some zones of *Cryosphere* use 'rimwards' (north), 'hubwards' (south), 'anticlockwise' (west) and 'clockwise' (east). These can be added in the same way. Remember that the game abbreviates both 'aft' and 'anticlockwise' with 'a', so you'll need to decide which direction you want to abbreviate. (The pre-configured world profile abbreviates 'anticlockwise', not 'aft', as 'a'.)

16 Axmud for visually-impaired users

Axmud can be run with settings optimised for users with a visual impairment. These settings don't require JAWS, NVDA or any other screenreader, but they do require a text-to-speech engine of some kind.

On Microsoft Windows, the Axmud installer includes a copy of the eSpeak engine. No further configuration is required. Simply run the installer, find Axmud in your system's Start menu, and open the menu item that's marked for visually-impaired users.

Most modern Linux systems have the eSpeak engine installed 'out-of-the-box'. If not, you should install it yourself before starting Axmud.

16.1 Compatible speech engines

Axmud supports other speech engines, too. (Not all of them are free.)

On Microsoft Windows, Axmud also supports espeak-ng and Swift (which is a part of Cepstral). On Linux, Axmud supports espeak-ng, Flite, Festival and Swift.

The rest of this section assumes that you're using optimised settings.

16.2 Connecting to a world

When you start Axmud, a series of dialogue windows will appear, asking you to select a world, a character and a password.

The first dialogue window asks you to select a world. Axmud comes with numerous pre-configured worlds, so you can either choose one of those, or you can create a new world.

If you create a new world and/or a new character, they are saved and will be available the next time you start Axmud.

In the first dialogue window, you can use your UP and DOWN cursor keys to select a world. When you find the world you want, press your TAB key to move to the OK button, and then press your ENTER key to 'click' that button.

Many of the remaining windows will ask you to type something, such as a character name or a password. After typing you can simply press the ENTER key to move to the next window. (Using your TAB and ENTER keys to 'click' the OK button will also work, but is unnecessary.)

If you specify a character name and a password, Axmud can usually login to the world on your behalf. If not, you'll have to type the name and password every time you connect to the world.

16.3 Playing the game

Axmud normally runs with multiple windows. There is a main window, in which most of the text received from the world is displayed. There are task windows, which typically display information about your character's health, the current location, and so on. There is also an automapper window and a great many edit windows used to modify Axmud's stored data.

Axmud's optimised settings dispense with many of these windows (although they are available, if you want them). Most of the time only the single main window will be open. When you connect to a world, everything is displayed in that main window, as well as being read out.

You can type commands like 'north' and 'inventory' and 'kill orc'. When you press ENTER, these world commands are sent directly to the world.

16.3.1 Client commands

Axmud provides its own commands, which we call 'client commands'. These client commands can be used to customise almost any aspect of Axmud's behaviour and to modify its stored data.

If you're using optimised settings, most client commands won't be of much use. However, there are a number of client commands which have been designed specifically for you. We'll spend the rest of this Section discussing how to use them.

All client commands begin with a semicolon. The first command we normally teach generates a (very) long list of all client commands, so you should probably resist the temptation to type it now. That command is the `;help` command - typed as a semicolon, followed by the word 'help', with no spaces in between.

Another useful client command is the `;save` command - typed as a semicolon, followed by the word 'save', with no spaces in between. This command produces a single-line response, so it's safe to type it now and press your ENTER key.

(By the way, Axmud's stored data is automatically saved whenever you disconnect from a world, and the autosave feature is turned on by default, so you probably don't need to save anything manually.)

If you get disconnected, the main window will not close. You can use the `;reconnect` command (a semicolon, followed by the word 'reconnect') to re-establish a connection to the same world.

Alternatively, you can use the `;connect` command to connect to a different world. If you want to connect to Discworld MUD, you can type `;connect discworld`. If you have already set up a character profile called Gandalf, you can type `;connect discworld gandalf`.

After disconnecting, you can halt Axmud by typing the `;stopclient` command. ('stopclient' is typed as a single word with no spaces.)

You can use the `;speak` command to test Axmud's text-to-speech abilities. `;speak` will read out a test message, but you can specify your own message with a command like `;speak hello world`.

16.4 Enabling/disabling text-to-speech

Text-to-speech is always enabled when using optimised settings, but for other users it's disabled by default.

Those users can type `;speech on` to enable text-to-speech or `;speech off` to disable text-to-speech.

16.4.1 Customising text-to-speech

By default, Axmud uses the eSpeak engine to read out everything in the same voice. However, Axmud supports multiple speech engines, and those engines often provide more than one voice. It's possible to configure Axmud to use different engines and voices in different situations.

For example, Axmud can use a male voice to read out text received from the world, and a female voice to read out system messages.

If you're using a pre-configured world, then Axmud is able to collect information about the state of the world. You can ask Axmud to read out this information whenever you need it (even if it's no longer visible in the main window).

For example, Axmud can tell you about your character's current location or the number of health points they have left. And if you want this information read out in a unique voice, you can do that, too.

The rest of this Section describes how to do all of those things. You can read it later, if you don't need those features yet.

16.4.2 Disabling some speech

You can use the client command `';speech'` to change what is read out, and what is not.

For example, `';speech system off'` will stop Axmud system messages from being read out. `';speech system on'` will tell Axmud to start reading them again.

`';speech error off'` will stop system error messages from being read out.

`';speech command off'` will stop every world command you type from being read out.

`';speech dialogue off'` will stop dialogue windows from being read out. (This includes the dialogue windows you see when you start Axmud, asking for a world, character name and password, so you should probably turn them on again before you quit using Axmud.)

`';speech receive off'` will stop text received from the world being read out.

Usually, received text isn't read out until the login has been completed. This is useful if the world displays a lot of graphics before asking for your name and password. If you want everything to be read out, you can type `';speech login off'`.

Speech generated by tasks is turned 'off' by default. You can turn it 'on' by typing `';speech task on'`. In Section 16.6 we'll discuss how to tell each task what they should read out.

16.4.3 Configuring the Festival engine

The `;speech` command is also used to change how Axmud uses the Festival engine (which is only available on Linux).

You will probably never need to change anything. However, if the Festival server on your system is running on a different port to the standard one, port 1314, you can change it by typing something like `;speech port 5000`.

If you don't want Axmud to start the Festival server automatically, you can type `;speech auto off`.

If Axmud doesn't start the Festival server automatically, you can either start it yourself, or you can type `;speech restart`. If you start it yourself, you can type `;speech reconnect` to let Axmud connect to the server.

16.5 Text-to-speech configurations

Axmud allows you to customise the sound of the speech produced by its supported speech engines.

Depending on which engine you're using, you can specify the engine, the voice used, the word speed, pitch and/or volume.

When using the eSpeak engine, Axmud can change the voice, speed and pitch. When using espeak-ng, everything can be changed. When using Flite, only the voice can be changed. When using Festival, the voice, word rate (not speed) and volume can be changed. When using Swift, the voice, pitch and volume can be changed, as well as the word speed (on Microsoft Windows) and word rate (on Linux).

We mentioned already that Axmud can use different engines and voices in different situations.

Axmud has a number of text-to-speech configurations, one for each situation. If you modify a configuration, your changes are stored and will still apply the next time you use Axmud.

16.5.1 Configuring the 'receive' configuration

The text-to-speech configuration called 'receive' is used for text received from the world.

The configuration stores your choice of speech engine, as well as your choice of voice, word speed (or word rate), word pitch and volume.

To change any of these things, we use the client command `;modifyconfig` (a semicolon, followed by a single word without spaces).

Actually, nearly all client commands have abbreviations, so this case you can type `;config` instead (a semicolon, followed by the word 'config').

The `;config` command is always followed by the name of the configuration we want to modify, and then by the modifications we want to make.

For example, text received from the world is read out using the eSpeak engine by default. eSpeak is rather primitive, so if Festival, Flite or Swift are installed on your system, you might prefer to use one of them instead.

The command to type is `';config receive engine festival'`.

To use the Flite engine, the command is `';config receive engine flite'`. To use the Swift engine, if you've purchased it, the command is `';config receive engine swift'`.

You can use the same command to change the voice. The list of available voices varies from engine to engine and from system to system, so it's up to you to work out which voices are available on your system.

To change the voice, first switch the engine back to eSpeak by typing `';config receive engine espeak'`.

Then, type `';config receive voice en-scottish'`. The last word is the letter E and N, followed by a hyphen, followed by the word 'scottish', all in lower-case letters and with no spaces.

If you specify a voice that isn't available on your system (or if you misspell the voice), you won't hear anything. In that case, you can use the `';config'` command to return the configuration to its default settings, by using the same command you would use to change the engine. Type `';config receive engine espeak'`.

You can use the `';speak'` command to test this configuration's new voice. The command to use is `';speak receive'`.

16.5.2 Configuring the speed, pitch and volume

Now we'll customise the rest of the configuration. Make sure you're using the eSpeak engine by typing the command `';config receive engine espeak'`.

To change the voice speed, type the command `';config receive speed 120'`. This will slow down the voice. To speed it up, type `';config receive speed 180'`. The speed can be anything between 10 and 200 words per minute.

The Festival and Swift engines use a word rate, rather than a word speed. The 'normal' rate is 1. To make the voice 50% slower, type `';config receive rate 0.5'`. To make it 100% faster, type `';config receive rate 2'`. The rate can be any value between 0.5 and 2.

To make the word pitch higher, type `';config receive pitch 80'`. To make it lower, type `';config receive pitch 20'`. With the eSpeak engine, the pitch can be any value between 0 and 99.

The Swift engine uses a different pitch scale. To make the pitch 50% lower, type `';config receive pitch 0.5'`. To make it 100% higher, type `';config receive pitch 2'`. The value can be anything between 0.1 and 5.

The eSpeak engine doesn't allow us to change the volume, but if you've modified the configuration to use the Festival or Swift engines, you can increase the volume by typing `';config receive volume 2'`. The volume can be any value between 0.33 and 6.

Again, to reset the voice, speed, rate, pitch and volume to default values for this engine, type `';config receive engine espeak'`.

16.5.3 Using different configurations

The text-to-speech configuration called 'receive' is used for text received from the world, but there are a number of other configurations ready for your use.

The 'system' configuration changes the sound of the speech used to read system messages. You might change this configuration's word speed by typing `';config system speed 200'`.

The 'error' configuration is used to read system error messages. The 'command' configuration is used to read out commands sent to the world. The 'dialogue' configuration is used to read out text from dialogue windows.

16.6 Configuring text-to-speech for tasks

Axmud comes with a number of built-in tasks, some of which have text-to-speech capabilities. The configuration called 'attack', 'chat', 'divert', 'locator', 'status' and 'watch' change the sound of the voice used by each of those tasks.

Tasks often have their own task windows. When using optimised settings, these windows don't open (but the tasks are still running in the background).

We'll discuss each task in turn.

16.6.1 The Locator task

The Locator task captures information about the character's current room, including the room's title, description, list of exits and sometimes even its contents.

The Locator task doesn't magically know how to interpret all possible worlds, so it's useful only when connected to one of Axmud's pre-configured worlds, or when connected to a world you've configured yourself (or which someone has configured for you).

The client command `';read'` will instruct the Locator to read out some of the information it has gathered.

For example, type `';read title'` to hear the current room's title (if known).

Type `';read description'` to hear the current room's description. If it's a very long description, type `';read description 100'` to hear the first 100 characters.

Type `';read exit'` to hear the exit list, and type `';read content'` to hear the contents list.

16.6.2 Auto-reading the room title

The Locator task is able to read out information about the current room automatically, every time the world sends that information.

This is probably not very useful, since all text received from the world is already read out by default. However, perhaps you want to limit the amount of text that gets read out, in which case you can stop Axmud reading out all received text by typing `';speech receive off'`.

You can then use the client command `';switch'`.

If you type `';switch title'`, the Locator will start reading out the room title, every time it is received. If you type `';switch title'` again, the Locator will stop reading out the room title.

To read out room descriptions automatically, type `';switch description'`.

To read out exit lists automatically, type `';switch exit'`.

To read out content lists automatically, type `';switch content'`.

16.6.3 Auto-read the room title permanently

`';switch title'` will make the Locator task automatically read out room titles for the rest of the session. However, the next time you start Axmud, a brand new Locator task will start, and it won't know that it's supposed to read out room titles.

The client command `';permswitch'` is the solution to this problem. (It's spelled with a semicolon, followed by perm and switch as a single word, without spaces.)

To read out room titles in this session, and in all future sessions, type `';permswitch title'`.

16.6.4 The Status task

The Status task gathers information about your character, such as their health points and experience points.

Like the Locator task, the Status task doesn't magically know how to gather this information. Both tasks work best when connected to a pre-configured world.

The client command `';read'` can be used to read out some of this information, as long as the world provides the information and Axmud knows how to collect it.

Type `';read health'` to read out the character's health points. You can also type `';read energy'`, `';read magic'`, `';read guild'`, `';read social'`, `';read experience'`, `';read level'` and `';read lives'`.

`';read status'` will read the character's life status - alive, dead, passed out or asleep.

`';read alignment'` will read the character's alignment (good or evil).

`';read age'` will read the character's age.

`';read time'` will read the game time.

`';read bank'` and `';read purse'` will reveal information about your character's finances. Money is tracked by the Inventory task, so you will need to start that task before you can use it.

The client commands `;starttask` and `;addinitialtask` are both typed as a semicolon followed by a single word with no spaces.

To start the Inventory task, type `;starttask inventory`. To make the inventory task start in every session, type `;addinitialtask inventory`.

16.6.5 Status task alerts

It would be very useful for the Status task to issue an audible warning when your character's health points fall to a certain level, and again when they recover.

You can do this with the client command `;alert`.

To get an audible warning when your character's health points fall to 20%, type `;alert health down 20`.

To get a warning when they recover to 90%, type `;alert health up 90`.

You can also use the `;alert` command to get audible warnings for your character's energy, magic, guild and social points, where available.

The `;alert` command only affects the current session. If you want these audible warnings every time you run Axmud, use the `;permalert` command. (It's spelled with a semicolon, followed by perm and alert as a single word, without spaces.)

For example, to warn about low health points every time you play, type `;permalert health down 20`.

By the way, you can use the `;switch` command to make the Status task read a notification when your character dies, falls asleep, passes out or recovers. Type `;switch life` to turn on these notifications, or to turn them off again.

16.6.6 The Attack task

The Attack task keeps track of your fights.

You can use `;switch` or `;permswitch` to get an audible notification of fights that have just finished (and, for some worlds, when they start.).

Type `;switch fight` to hear the results of fights, and the same command to stop hearing them. Type `;switch interaction` to hear the results of interactions.

16.6.7 The Divert task

The Divert task is commonly used to divert certain lines of text from the main window into a separate task window. It's handy for making sure you don't miss an important messages in the heat of battle.

Visually-impaired users can use the Divert task to make sure that important messages are read out in a different voice - just modify the configuration called 'divert'.

To turn on the reading of all diverted lines of text, type `;switch divert`. Use the same command to turn them off. Use the `;permswitch` command to make this change in this and all future sessions.

You can also turn on or off specific lines of text. The Divert task distinguishes between tells (private communications between players), social messages (public communication channels), custom messages (any other line of text that should be diverted) and warnings generated by some part of Axmud's code. Each type of message makes the task window turn a different colour and a sound effect is also heard.

If you're using optimised settings then there is probably no task window open. To turn on the reading out of all tells, type `;switch tell`. You can also type `;switch social`, `;switch custom`, `;switch warning`.

16.6.8 The Chat task

The Chat task allows players to communicate with each other directly, independent of the world in which they are playing. It uses a standard protocol, so users of different MUD clients can talk to each other.

This form of peer-to-peer communication isn't much used any more, but it's available if you need it.

To read out all Chat task communications, type `;switch chat`. Use the same command again to turn it off.

You might prefer to hear only incoming communications or only outgoing communications. For that, type `;switch chat in` or `;switch chat out`. To read out group communications, type `;switch chat echo`.

To read out Chat system messages, type `;switch chat system` or `;switch chat remote`.

So-called 'snooping' - the ability for your chat partner to see what's happening in your session - is turned off by default. If you've turned it on, you can make Axmud read out so-called 'snooping' text by typing `;switch chat snoop`.

16.6.9 The Compass task

The Compass task converts your keyboard's keypad - the collection of number keys usually found on the right-hand-side of the keyboard - into a device that can execute a whole command with a single key-press.

Optimised settings start this task automatically, so if you press the keypad's 8 key, Axmud will send a 'north' command to the world. The 2 key sends 'south', the 4 key sends 'west' and the 6 key sends 'east'. The 7, 9, 1 and 3 keys send 'northwest', 'northeast', 'southwest' and 'southeast' respectively. The plus key sends 'up' and the minus key sends 'down'.

Don't forget that they keypad's keys can be turned on and off with the 'Num lock' button found on most keyboards.

The remaining keypad keys can be customised. Each key can either send a world command or executes a client command. For example, the 5 key sends the world command 'look'. The 0 key executes the client command `;kill`, which attacks an NPC in the current room (if there are any).

If you want to customise the 0 key, you can use the `';compass'` command.

To make the 0 key do something nicer, type `';compass 0 say hello'`.

To make the keypad ENTER key save Axmud's files, type `';compass enter ;save'`. In this command, both the words 'compass' and 'save' are preceded by a semicolon, with no space between a word and the semicolon before it.

The keys that can be customised are the keypad 0, 5, multiply, divide, full stop or period and ENTER keys. The remaining keypad keys are used for standard movement commands and can't be customised directly.

To disable the keypad, so that its keys behave normally, type `';compass off'`. To reenable it, typing `';compass on'`.

Of course, the `';compass'` command will only make changes for this session. If you want a Compass task to start every session, you have two choices.

Firstly, you can create a task that starts in every session by typing the command `';addinitialtask compass'`. Then you can customise the keypad for this and all future sessions using the `';permcompass'` command.

Secondly, you can create a task that starts whenever you connect to a particular world. For example, to create a task that starts every time you connect to Discworld MUD, type the command `';addinitialtask compass discworld'`. In this situation, the customisation is done with the `';worldcompass'` command.

The commands - `';compass'` and `';permcompass'` behave in the same way as `';compass'` does, so you use all three of them to disable or reenable they keypad and to set the commands for each key.

That's the end of this Section for visually-impaired users.

17 Peek/Poke Operations

Much of Axmud's internal data is available to you, and to any scripts and plugins you write.

This data can be accessed using the client commands `'peek'` and `'poke'`. Axbasic scripts can access the data through a number of statements, for example PEEK and POKE. Plugins (written in perl) can access the data directly, or through a call to `GA::Session->parsePeekPoke`.

A peek operation allows you to see internal data. A poke operation allows you to modify it. Most of Axmud's internal data can be seen via a peek operation, but in general, data can only be modified in a poke operation when it's safe to do so.

Peek and poke operations take a string as an argument. The string is separated into a compulsory object part and an optional IV (instance variable) part.

(In Perl terminology, an object is a collection of data, and an IV is a variable stored in that collection. The variable can be a scalar, list or hash.)

An example of an object part is `'prof.deathmud'`, representing a world profile called `'deathmud'`, and an example of an IV part is `'port'`. Thus, you can access the world's port by peeking the string `'prof.deathmud.port'`.

The object part is translated into the blessed reference of the object. (In Perl terminology, the blessed reference is a name given to each distinct object.)

If an IV part is specified, the peek/poke operation returns the value of the IV. If no IV part is specified, the peek/poke operation returns the blessed reference of the object.

This section contains a complete list of strings that can be used in peek/poke operations.

In some cases, the string corresponds to a particular IV in a particular object, e.g. `'world.current'` refers to the current world profile (the Perl class is `GA::Session->currentWorld`).

In some cases, the string refers to a whole object, e.g. `'dict.current'` refers to the current dictionary (the Perl class is `GA::Session->currentDict`). In those cases, you can add any valid IV to the string, e.g. `'dict.current.weaponHash'`. The IV part is case-sensitive, so `'dict.current.weaponhash'` will not work.

Axmud global variables like `$axmud::SCRIPT` are used here as if they were IVs in the `GA::Client` object (for convenience).

The values returned by many strings depend on the session. For example, `'dict.current'` will return this session's current dictionary, not some other session's current dictionary.

Although it's possible to directly modify an IV's value with a poke operation, doing so is **STRONGLY DISCOURAGED**.

It is almost always better to use client commands and Axmud's edit, preference and wizard windows, which take care of all the complications that might not be obvious to you.

If you do need to modify values directly (for testing purposes, perhaps), you should at least read the comments in the source code for the object you're modifying. (Most of what you need is in the `->new` function.)

All Axmud objects have a `->_privFlag` IV. If the value is `TRUE`, that object's IVs are 'private' and cannot be modified in a poke operation at all. If the value is `FALSE`, that object's IVs are 'public' and can be modified in a poke operation.

Objects whose `->_privFlag` is `FALSE` are designed (for the most part) as simple stores of data, so it's usually safe to modify them directly (but still read the source code first).

All IVs beginning with an underline, such as `->_privFlag` itself, are 'standard' IVs that can't be modified in any circumstances.

The strings specified below return the following kinds of value in a peek/poke operation:

- `o` - object whose `->_privFlag` is `FALSE` (IVs are public)
- `O` - object whose `->_privFlag` is `TRUE` (IVs are private)
- `s` - scalar
- `S` - read-only scalar
- `f` - scalar flag (always `TRUE` or `FALSE`)
- `F` - read-only scalar flag (always `TRUE` or `FALSE`)
- `l` - list
- `L` - read-only list
- `h` - hash
- `H` - read-only hash
- `X` - read-only scalar, list or hash (for JSON data structures)

`L atcp.list`

- List of (all) ATCP packages received (from `GA::Session->atcpDataHash`)

`L atcp.list.PACKAGE`

- List of ATCP packages received in the form `PACKAGE[.subpackage][.message]` (from `GA::Session->atcpDataHash`)

`L atcp.list.PACKAGE.SUBPACKAGE`

- List of ATCP packages received in the form `PACKAGE[.SUBPACKAGE][.message]` (from `GA::Session->atcpDataHash`)

`S atcp.data.PACKAGE[.SUBPACKAGE][.MESSAGE]`

- Value of undecoded data string in the ATCP package (from `GA::Session->atcpDataHash`)

`X atcp.val.PACKAGE[.SUBPACKAGE][.MESSAGE][.json]`

- A value embedded within the JSON data. 'json' is made up of any number of 'INDEX' and 'KEY' components, in any order. The returned value is the scalar, list or hash value stored within the embedded list/hash references; or 'undef' if 'json' doesn't match any embedded scalar value (from `GA::Session->atcpDataHash`)

s buffer.size.display
 - Custom maximum display buffer size (GA::Client->customDisplayBufferSize)

s buffer.size.instruct
 - Custom maximum instruction buffer size (GA::Client->customInstructBufferSize)

s buffer.size.cmd
 - Custom maximum world command buffer size (GA::Client->customCmdBufferSize)

o buffer.client.instruct.NUMBER
 - Client's instruction buffer object NUMBER (from GA::Client->instructBufferHash)

S buffer.client.instruct.count
 - No. instructions processed altogether (GA::Client->instructBufferCount)

S buffer.client.instruct.first
 - NUMBER of earliest surviving buffer object (GA::Client->instructBufferFirst)

S buffer.client.instruct.last
 - NUMBER of most recent buffer object (GA::Client->instructBufferLast)

o buffer.client.cmd.NUMBER
 - Client's world command buffer object NUMBER (from GA::Client->cmdBufferHash)

S buffer.client.cmd.count
 - No. commands processed altogether (GA::Client->cmdBufferCount)

S buffer.client.cmd.first
 - NUMBER of earliest surviving buffer object (GA::Client->cmdBufferFirst)

S buffer.client.cmd.last
 - NUMBER of most recent buffer object (GA::Client->cmdBufferLast)

o buffer.session.display.NUMBER
 - Session's display buffer object NUMBER (from GA::Session->displayBufferHash)

S buffer.session.display.count
 - No. lines processed altogether (GA::Session->displayBufferCount)

S buffer.session.display.first
 - NUMBER of earliest surviving buffer object (GA::Session->displayBufferFirst)

S buffer.session.display.last
 - NUMBER of most recent buffer object (GA::Session->displayBufferLast)

o buffer.session.instruct.NUMBER
 - Session's instruction buffer object NUMBER (from GA::Session->instructBufferHash)

S buffer.session.instruct.count
 - No. instructions processed altogether (GA::Session->instructBufferCount)

S buffer.session.instruct.first
 - NUMBER of earliest surviving buffer object (GA::Session->instructBufferFirst)

S buffer.session.instruct.last
 - NUMBER of most recent buffer object (GA::Session->instructBufferLast)

o buffer.session.cmd.NUMBER
 - Session's world command buffer object NUMBER (from GA::Session->cmdBufferHash)

S buffer.session.cmd.count
 - No. commands processed altogether (GA::Session->cmdBufferCount)

S buffer.session.cmd.first
 - NUMBER of earliest surviving buffer object (GA::Session->cmdBufferFirst)

S buffer.session.cmd.last
 - NUMBER of most recent buffer object (GA::Session->cmdBufferLast)

- o cage.current.TYPE.CATEGORY
 - Current cage of type 'TYPE' (e.g. 'trigger') associated with the category 'CATEGORY' (from GA::Session->currentCageHash)
- o cage.name.NAME
 - Cage called 'NAME' (from GA::Session->cageHash)
- o cage.TYPE.PROF
 - Cage of type 'TYPE' (e.g. 'trigger') associated with the profile called 'PROF' (from GA::Session->cageHash)
- L cage.type.list
 - Customisable list of cage types (GA::Client->cageTypeList)

- o char.current
 - Current character profile (GA::Session->currentChar)
- o char.NAME
 - Character profile called 'NAME' (from GA::Session->profHash)

- o chat.contact.NAME
 - Chat contact object called 'NAME' (from GA::Client->chatContactHash)
- S chat.name
 - Chat task name to broadcast (GA::Client->chatName)
- S chat.email
 - Chat task email to broadcast (GA::Client->chatEmail)
- S chat.mode
 - Chat task accept mode (GA::Client->chatAcceptMode)
- S chat.icon
 - Chat task icon to broadcast (from GA::Client->chatIcon)

- S client.name
 - \$SCRIPT (Axmud global variable)
- S client.name.short
 - \$NAME_SHORT (Axmud global variable)
- S client.name.article
 - \$NAME_ARTICLE (Axmud global variable)
- S client.version
 - \$VERSION (Axmud global variable)
- S client.date
 - \$DATE (Axmud global variable)
- S client.basic.name
 - \$BASIC_NAME (Axmud global variable)
- S client.basic.article
 - \$BASIC_ARTICLE (Axmud global variable)
- S client.basic.version
 - \$BASIC_VERSION (Axmud global variable)
- S client.authors
 - \$AUTHORS (Axmud global variable)
- S client.copyright
 - \$COPYRIGHT (Axmud global variable)

S client.url
 - \$URL (Axmud global variable)
 S client.file.name
 - \$NAME_FILE (Axmud global variable)
 L client.file.list
 - @COMPAT_FILE_LIST (Axmud global variable)
 S client.dir.base
 - \$BASE_DIR (Axmud global variable)
 S client.dir.data
 - \$DATA_DIR (Axmud global variable)
 L client.dir.list
 - @COMPAT_DIR_LIST (Axmud global variable)
 L client.ext.list
 - @COMPAT_EXT_LIST (Axmud global variable)
 F client.mode.blind
 - \$BLIND_MODE_FLAG (Axmud global variable)
 F client.mode.safe
 - \$SAFE_MODE_FLAG (Axmud global variable)
 F client.mode.test
 - \$TEST_MODE_FLAG (Axmud global variable)
 L client.license
 - @LICENSE_LIST (Axmud global variable)
 L client.credit
 - @CREDIT_LIST (Axmud global variable)

S client.start.time
 - Client start time (system time, in secs) (GA::Client->startTime)
 S client.start.clock
 - Time client started (GA::Client->startClock)
 S client.start.date
 - Date client started (GA::Client->startDate)
 S client.start.clockstring
 - Time client started, formatted string (GA::Client->startClockString)
 S client.start.datestring
 - Date client started, formatted string (GA::Client->startDateString)
 S client.delay.prompt
 - Time to wait before using a prompt (GA::Client->promptWaitTime)
 S client.delay.login
 - Time to wait before showing login warning (GA::Client->loginWarningTime)
 L client.iv
 - List of IVs required in all Perl objects (from GA::Client->constIVHash)
 L client.reserved
 - List of reserved names (from GA::Client->constReservedHash)

- o colour.scheme.NAME
- o color.scheme.NAME
 - Colour scheme object called 'NAME' (from GA::Client->colourSchemeHash)
- S colour.system.cmd
- S color.system.cmd
 - Colour for sent world commands (GA::Client->customInsertCmdColour)
- S colour.system.text
- S color.system.text
 - Colour for system messages (GA::Client->customShowTextColour)
- S colour.system.error
- S color.system.error
 - Colour for system error messages (GA::Client->customShowErrorColour)
- S colour.system.warning
- S color.system.warning
 - Colour for system warning messages (GA::Client->customShowWarningColour)
- S colour.system.debug
- S color.system.debug
 - Colour for system debug messages (GA::Client->customShowDebugColour)
- S colour.system.improper
- S color.system.improper
 - Colour for system improper arguments messages
(GA::Client->customShowImproperColour)
- F colour.flag.invisible
- F color.flag.invisible
 - Convert text colour if background colour is the same (GA::Client->convertInvisibleFlag)
- S colour.rgb.TAG
- S color.rgb.TAG
 - RGB (e.g. '#ABCDEF') for standard colour TAG (from GA::Client->colourTagHash and
->boldColourTagHash)
- S colour.misc.xterm
- S color.misc.xterm
 - xterm colour cube in use (GA::Client->currentColourCube)
- F colour.misc.osc
- F color.misc.osc
 - OSC colour paletters allowed (GA::Client->oscPaletteFlag)

- o custom.current.CATEGORY
 - Current custom profile of category 'CATEGORY' (from GA::Session->currentProfHash)
- o custom.NAME
 - Custom profile called 'NAME' (from GA::Session->profHash)

- F debug.protocol.telnet
 - Show incoming option negotiation info (GA::Client->debugTelnetFlag)
- F debug.protocol.telnet.short
 - Show incoming option negotiation short info (GA::Client->debugTelnetMiniFlag)
- F debug.protocol.log
 - Write log for incoming option negotiation, telopt.log (GA::Client->debugTelnetLogFlag)
- F debug.protocol.msdp
 - Show MSDP data sent to Locator/Status tasks (GA::Client->debugMsdpFlag)
- F debug.protocol.mxp
 - Show MXP errors (GA::Client->debugMxpFlag)
- F debug.protocol.mxp.comment
 - Show MXP comments (GA::Client->debugMxpCommentFlag)
- F debug.protocol.pueblo
 - Show Pueblo errors (GA::Client->debugPuebloFlag)
- F debug.protocol.pueblo.comment
 - Show Pueblo comments (GA::Client->debugPuebloCommentFlag)
- f debug.protocol.atcp
 - Show ATCP data (GA::Client->debugAtcpFlag)
- f debug.protocol.gmcp
 - Show GMCP data (GA::Client->debugGmcpFlag)
- F debug.line.numbers
 - Show explicit line numbers (GA::Client->debugLineNumsFlag)
- F debug.line.tags
 - Show explicit colour/style tags (GA::Client->debugLineTagsFlag)
- F debug.locator.some
 - Show some Locator debug messages (GA::Client->debugLocatorFlag)
- F debug.locator.all
 - Show all Locator debug messages (GA::Client->debugMaxLocatorFlag)
- F debug.locator.move
 - Show Locator's movement command list (GA::Client->debugMoveListFlag)
- F debug.obj.parse
 - Show debug messages when parsing objects (GA::Client->debugParseObjFlag)
- F debug.obj.compare
 - Show debug messages when comparing objects (GA::Client->debugCompareObjFlag)
- F debug.error.plugin
 - Show explanatory messages if plugin can't be loaded
(GA::Client->debugExplainPluginFlag)
- F debug.error.iv
 - Show debug message for non-existent IVs (GA::Client->debugCheckIVFlag)
- F debug.error.table
 - Show debug message if table object can't be added/resized
(GA::Client->debugTableFitFlag)
- F debug.error.trap
 - Trap Perl errors/warnings in 'main' window (GA::Client->debugTrapErrorFlag)

S desktop.panel.left
 - Size of left panel (GA::Client->customPanelLeftSize)

S desktop.panel.right
 - Size of right panel (GA::Client->customPanelRightSize)

S desktop.panel.top
 - Size of top panel (GA::Client->customPanelTopSize)

S desktop.panel.bottom
 - Size of bottom panel (GA::Client->customPanelBottomSize)

S desktop.controls.left
 - Size of left-side window controls (GA::Client->customControlsLeftSize)

S desktop.controls.right
 - Size of right-side window controls (GA::Client->customControlsRightSize)

S desktop.controls.top
 - Size of top-side window controls (GA::Client->customControlsTopSize)

S desktop.controls.bottom
 - Size of bottom-side window controls (GA::Client->customControlsBottomSize)

O dict.current
 - Current dictionary object (GA::Session->currentDict)

O dict.NAME
 - Dictionary object called 'NAME' (from GA::Client->dictHash)

o exit.number.NUMBER
 - Exit model object with number 'NUMBER' (from
 GA::Obj::WorldModel->exitModelHash)

o exit.newest
 - Most recently-created exit model object (GA::Obj::WorldModel->mostRecentExitNum)

S exit.count
 - No. exits in model (including deleted exits) (GA::Obj::WorldModel->exitObjCount)

S exit.actual
 - No. exits in model (not including deleted exits)
 (GA::Obj::WorldModel->exitActualCount)

S external.browser
 - Command to run a web browser (GA::Client->browserCmd)

S external.email
 - Command to open email app (GA::Client->emailCmd)

S external.audio
 - Command to open audio player (GA::Client->audioCmd)

S external.editor
 - Command to open text editor (GA::Client->textEditCmd)

F file.config.load
 - Load config file (GA::Client->loadConfigFlag)
 F file.config.save
 - Load config file (GA::Client->saveConfigFlag)
 F file.data.load
 - Load data files (GA::Client->loadDataFlag)
 F file.data.save
 - Save data files (GA::Client->saveDataFlag)
 F file.start.delete
 - Delete config/data files on startup (GA::Client->deleteFilesAtStartFlag)
 F file.load.fail
 - File operation failure (GA::Client->fileFailFlag)
 F file.save.backup
 - Backup files before saving (GA::Client->autoRetainFileFlag)
 F file.save.modify
 - Data saved in any file objects have been modified (GA::Client->showModFlag)
 F file.autosave.flag
 - Auto-save turned on (GA::Client->autoSaveFlag)
 S file.autosave.time
 - Minutes between autosaves (GA::Client->autoSaveWaitTime)
 O file.client.NAME
 - Client file object called NAME (from GA::Client->fileObjHash)
 O file.session.NAME
 - Session file object called NAME (from GA::Session->sessionFileObjHash)

L gmcp.list
 - List of (all) GMCP packages received (from GA::Session->gmcpDataHash)
 L gmcp.list.PACKAGE
 - List of GMCP packages received in the form PACKAGE[.subpackage][.message] (from GA::Session->gmcpDataHash)
 L gmcp.list.PACKAGE.SUBPACKAGE
 - List of GMCP packages received in the form PACKAGE.SUBPACKAGE[.message] (from GA::Session->gmcpDataHash)
 S gmcp.data.PACKAGE[.SUBPACKAGE][.MESSAGE]
 - Value of undecoded data string in the GMCP package (from GA::Session->gmcpDataHash)
 X gmcp.val.PACKAGE[.SUBPACKAGE][.MESSAGE][.json]
 - A value embedded within the JSON data. 'json' is made up of any number of '.INDEX' and '.KEY' components, in any order. The returned value is the scalar, list or hash value stored within the embedded list/hash references; or 'undef' if '.json' doesn't match any embedded scalar value (from GA::Session->gmcpDataHash)

o guild.current
 - Current guild profile (GA::Session->currentGuild)
 o guild.NAME
 - Guild profile called 'NAME' (from GA::Session->profHash)

O iface.name.NAME
 - Active interface called 'NAME' (from GA::Session->interfaceHash)

O iface.number.NUMBER
 - Active interface numbered 'NUMBER' (from GA::Session->interfaceNumHash)

O imodel.TYPE
 - Interface model of the type 'TYPE', e.g. 'trigger' (from GA::Client->interfaceModelHash)

S instruct.sigil.client
 - Instruction sigil for client commands (GA::Client->constClientSigil)

S instruct.sigil.forced
 - Instruction sigil for forced world commands (GA::Client->constForcedSigil)

S instruct.sigil.echo
 - Instruction sigil for echo commands (GA::Client->constEchoSigil)

S instruct.sigil.perl
 - Instruction sigil for Perl commands (GA::Client->constPerlSigil)

S instruct.sigil.script
 - Instruction sigil for script commands (GA::Client->constScriptSigil)

S instruct.sigil.multi
 - Instruction sigil for multi commands (GA::Client->constMultiSigil)

S instruct.sigil.speedwalk
 - Instruction sigil for speedwalk commands (GA::Client->constSpeedSigil)

S instruct.sigil.bypass
 - Instruction sigil for bypass commands (GA::Client->constBypassSigil)

F instruct.enable.echo
 - Echo command sigil enabled (GA::Client->echoSigilFlag)

F instruct.enable.perl
 - Perl command sigil enabled (GA::Client->perlSigilFlag)

F instruct.enable.script
 - Script command sigil enabled (GA::Client->scriptSigilFlag)

F instruct.enable.multi
 - Multi command sigil enabled (GA::Client->multiSigilFlag)

F instruct.enable.speed
 - Speed command sigil enabled (GA::Client->speedSigilFlag)

F instruct.enable.bypass
 - Bypass command sigil enabled (GA::Client->bypassSigilFlag)

S instruct.separator
 - World command separator (GA::Client->cmdSep)

F instruct.world.confirm
 - Confirm world commands in session's default tab (GA::Client->confirmWorldCmdFlag)

F instruct.world.convert
 - Convert capitalised world commands (GA::Client->convertWorldCmdFlag)

F instruct.world.preserve
 - Preserve world commands in entry box (GA::Client->preserveWorldCmdFlag)

F instruct.other.preserve
 - Preserve other commands in entry box (GA::Client->preserveOtherCmdFlag)

F instruct.multi.max
 - Send multi commands to all sessions (GA::Client->maxMultiCmdFlag)

F instruct.complete.mode
 - Auto-complete mode (GA::Client->autoCompleteMode)

F instruct.complete.type
 - Auto-complete navigation type (GA::Client->autoCompleteType)

F instruct.complete.parent
 - Auto-complete registry location (GA::Client->autoCompleteParent)

O keycode.obj.current
 - Current keycode object (GA::Client->currentKeycodeObj)

O keycode.obj.NAME
 - Keycode object called 'NAME' (from GA::Client->keycodeObjHash)

S keycode.current.KEYCODE
 - Current value for the standard KEYCODE value (from GA::Obj::Keycode->keycodeHash)

S keycode.default.KEYCODE
 - Default value for the standard KEYCODE value (from GA::CLIENT->constKeycodeHash)

F log.allow
 - Logging is enabled (GA::CLIENT->allowLogsFlag)

F log.client.FILE
 - Flag to show whether the client logfile called 'FILE' is being written (from GA::Client->logPrefHash)

F log.world.FILE
 - Flag to show whether the world logfile called 'FILE' is being written (from GA::Profile::World->logPrefHash)

F log.delete.standard
 - Delete client logfiles when client starts (GA::Client->deleteStandardLogsFlag)

F log.delete.world
 - Delete world logfiles when world becomes current (GA::Client->deleteWorldLogsFlag)

F log.start.day
 - Start new logfile every day (GA::Client->logDayFlag)

F log.start.client
 - Start new logfile when client starts (GA::Client->logClientFlag)

F log.prefix.date
 - Prefix every line with date (GA::Client->logPrefixDateFlag)

F log.prefix.time
 - Prefix every line with time (GA::Client->logPrefixTimeFlag)

F log.image
 - Record images in 'receive' logfile (GA::Client->logImageFlag)

S log.event.before
 - Lines to write before Status task event (GA::Client->statusEventBeforeCount)

S log.event.after
 - Lines to write after Status task event (GA::Client->statusEventAfterCount)

S loop.client.time
 - Client loop time (GA::Client->clientTime)
 O loop.client.obj
 - Client loop object (GA::Client->clientLoopObj)
 F loop.client.spin
 - Client loop is spinning (GA::Client->clientLoopSpinFlag)
 S loop.client.delay
 - Client loop delay time (GA::Client->clientLoopDelay)
 S loop.session.time
 - Session loop time (GA::Session->sessionTime)
 S loop.session.obj
 - Session loop object (GA::Session->sessionLoopObj)
 F loop.session.spin
 - Session loop is spinning (GA::Session->sessionLoopSpinFlag)
 F loop.session.child
 - Session loop's child loop is spinning (GA::Session->childLoopSpinFlag)
 S loop.session.delay
 - Session loop delay time (GA::Session->sessionLoopDelay)

L misc.months
 - Customisable list of short months (GA::Client->customMonthList)
 L misc.days
 - Customisable list of short days (GA::Client->customDayList)

s model.author
 - World model author (GA::Obj::WorldModel->author)
 s model.date
 - World model date (GA::Obj::WorldModel->date)
 s model.version
 - World model version (GA::Obj::WorldModel->version)
 l model.descripList
 - World model description (GA::Obj::WorldModel->descripList)
 o model.number.NUMBER
 - World model object with number 'NUMBER' (from GA::Obj::WorldModel->modelHash)
 o model.region.NUMBER
 - World model region object with number 'NUMBER' (from
 GA::Obj::WorldModel->regionModelHash)
 o model.room.NUMBER
 - World model room object with number 'NUMBER' (from
 GA::Obj::WorldModel->roomModelHash)
 o model.weapon.NUMBER
 - World model weapon object with number 'NUMBER' (from
 GA::Obj::WorldModel->weaponModelHash)
 o model. armour.NUMBER
 - World model region object with number 'NUMBER' (from
 GA::Obj::WorldModel->armourModelHash)

- o model.garment.NUMBER
 - World model garment object with number 'NUMBER' (from GA::Obj::WorldModel->garmentModelHash)
- o model.char.NUMBER
 - World model char object with number 'NUMBER' (from GA::Obj::WorldModel->charModelHash)
- o model.minion.NUMBER
 - World model minion object with number 'NUMBER' (from GA::Obj::WorldModel->minionModelHash)
- o model.sentient.NUMBER
 - World model sentient object with number 'NUMBER' (from GA::Obj::WorldModel->sentientModelHash)
- o model.creature.NUMBER
 - World model creature object with number 'NUMBER' (from GA::Obj::WorldModel->creatureModelHash)
- o model.portable.NUMBER
 - World model portable object with number 'NUMBER' (from GA::Obj::WorldModel->portableModelHash)
- o model.decoration.NUMBER
 - World decoration region object with number 'NUMBER' (from GA::Obj::WorldModel->decorationModelHash)
- o model.custom.NUMBER
 - World model custom object with number 'NUMBER' (from GA::Obj::WorldModel->customModelHash)
- O model.regionmap.NAME
 - World model regionmap called 'NAME' (from GA::Obj::WorldModel->regionmapHash)
- o model.kchar.NAME
 - World model known character called 'NAME' (from GA::Obj::WorldModel->knownCharHash)
- o model.mstring.STRING
 - World model minion string 'STRING' (from GA::Obj::WorldModel->minionStringHash)
- o model.tag.TAG
 - World model room object with room tag 'TAG' (from GA::Obj::WorldModel->roomTagHash)
- o model.newest
 - Most recently-created world model object (GA::Obj::WorldModel->mostRecentNum)
- S model.count
 - No. objects in world model (including deleted objects) (GA::Obj::WorldModel->modelObjCount)
- S model.actual
 - No. objects in world model (not including deleted objects) (GA::Obj::WorldModel->modelActualCount)
- S model.lightStatus
 - Current light status (GA::Obj::WorldModel->...)
- S model.defaultGridWidthBlocks
 - Default grid width (in blocks) (GA::Obj::WorldModel->...)
- S model.defaultGridHeightBlocks
 - Default grid height (in blocks) (GA::Obj::WorldModel->...)
- S model.defaultBlockWidthPixels
 - Default block width (in pixels) (GA::Obj::WorldModel->...)

S model.defaultBlockHeightPixels
 - Default block height (in pixels) (GA::Obj::WorldModel->...)

S model.defaultRoomWidthPixels
 - Default room width (in pixels) (GA::Obj::WorldModel->...)

S model.defaultRoomHeightPixels
 - Default room height (in pixels) (GA::Obj::WorldModel->...)

S model.maxGridWidthBlocks
 - Max grid width (in blocks) (GA::Obj::WorldModel->...)

S model.maxGridHeightBlocks
 - Max grid height (in blocks) (GA::Obj::WorldModel->...)

S model.maxBlockWidthPixels
 - Max block width (in pixels) (GA::Obj::WorldModel->...)

S model.maxBlockHeightPixels
 - Max block height (in pixels) (GA::Obj::WorldModel->...)

S model.maxRoomWidthPixels
 - Max room width (in pixels) (GA::Obj::WorldModel->...)

S model.maxRoomHeightPixels
 - Max room height (in pixels) (GA::Obj::WorldModel->...)

S model.defaultMapWidthPixels
 - Default map width (in pixels) (GA::Obj::WorldModel->...)

S model.defaultMapHeightPixels
 - Default map height (in pixels) (GA::Obj::WorldModel->...)

S model.currentRoomMode
 - Room drawing mode (GA::Obj::WorldModel->...)

S model.roomInteriorMode
 - Room interior mode (GA::Obj::WorldModel->...)

S model.drawExitMode
 - Exit drawing mode (GA::Obj::WorldModel->...)

F model.drawOrnamentsFlag
 - Draw exit ornaments (GA::Obj::WorldModel->...)

S model.horizontalExitLengthBlocks
 - Length of horizontal exits (in blocks) (GA::Obj::WorldModel->...)

S model.verticalExitLengthBlocks
 - Length of vertical exits (in blocks) (GA::Obj::WorldModel->...)

F model.drawBentExitsFlag
 - Draw bent exits (GA::Obj::WorldModel->...)

F model.matchTitleFlag
 - Match room titles (GA::Obj::WorldModel->...)

F model.matchDescripFlag
 - Match room descriptions (GA::Obj::WorldModel->...)

S model.matchDescripCharCount
 - No. characters to match in descriptions (GA::Obj::WorldModel->...)

F model.matchExitFlag
 - Match room exits (GA::Obj::WorldModel->...)

F model.analyseDescripFlag
 - Analyse room descriptions (GA::Obj::WorldModel->...)

F model.matchSourceFlag
 - Match room's source code file (GA::Obj::WorldModel->...)

F model.matchVNumFlag
 - Match room's remove vnum (GA::Obj::WorldModel->...)

F model.updateTitleFlag
 - Update room titles (in 'update' mode) (GA::Obj::WorldModel->...)

F model.updateDescripFlag
 - Update room descriptions (GA::Obj::WorldModel->...)

F model.updateExitFlag
 - Update room exits (GA::Obj::WorldModel->...)

F model.updateSourceFlag
 - Update room's source code file (GA::Obj::WorldModel->...)

F model.updateVNumFlag
 - Update room's remote vnum (GA::Obj::WorldModel->...)

F model.updateRoomCmdFlag
 - Update room's room command list (GA::Obj::WorldModel->...)

F model.updateOrnamentFlag
 - Update room's exit ornaments (GA::Obj::WorldModel->...)

F model.assistedMovesFlag
 - Apply assisted moves (GA::Obj::WorldModel->...)

F model.assistedBreakFlag
 - Break down doors (GA::Obj::WorldModel->...)

F model.assistedPickFlag
 - Pick locks (GA::Obj::WorldModel->...)

F model.assistedUnlockFlag
 - Unlock doors (GA::Obj::WorldModel->...)

F model.assistedOpenFlag
 - Open doors (GA::Obj::WorldModel->...)

F model.assistedCloseFlag
 - Close doors (GA::Obj::WorldModel->...)

F model.assistedLockFlag
 - Lock doors (GA::Obj::WorldModel->...)

F model.protectedMovesFlag
 - Protected moves (GA::Obj::WorldModel->...)

F model.superProtectedMovesFlag
 - Super-protected moves (GA::Obj::WorldModel->...)

F model.setTwinOrnamentFlag
 - Set twin exit's ornament (GA::Obj::WorldModel->...)

F model.countVisitsFlag
 - Count character visits (GA::Obj::WorldModel->...)

F model.allowModelScriptFlag
 - Allow model-wide scripts (GA::Obj::WorldModel->...)

F model.allowRoomScriptFlag
 - Allow individual room scripts (GA::Obj::WorldModel->...)

F model.intelligentExitsFlag
 - Use intelligent exits (GA::Obj::WorldModel->...)

F model.autoCompareFlag
 - Auto-compare new rooms (GA::Obj::WorldModel->...)

F model.followAnchorFlag
 - Create new exit object using follow anchor pattern (GA::Obj::WorldModel->...)

F model.capitalisedRoomTagFlag
 - Display room tags in capitals (GA::Obj::WorldModel->...)

F model.showTooltipsFlag
 - Show tooltips (GA::Obj::WorldModel->...)

F model.explainGetLostFlag
 - Explain when automapper gets lost (GA::Obj::WorldModel->...)

F model.disableUpdateModeFlag
 - Disable automapper update mode (GA::Obj::WorldModel->...)

F model.updateExitTagFlag
 - Update region exit tags (GA::Obj::WorldModel->...)

F model.drawRoomEchoFlag
 - Draw room echos (GA::Obj::WorldModel->...)

F model.trackPosnFlag
 - Track position and scroll window (GA::Obj::WorldModel->...)

S model.trackingSensitivity
 - Tracking sensitivity (GA::Obj::WorldModel->...)

F model.avoidHazardsFlag
 - Avoid hazards when pathfinding (GA::Obj::WorldModel->...)

F model.postProcessingFlag
 - Produce smooth paths when pathfinding (GA::Obj::WorldModel->...)

F model.quickPathFindFlag
 - Go to double-clicked room (GA::Obj::WorldModel->...)

F model.autocompleteExitsFlag
 - Auto-complete uncertain exits (GA::Obj::WorldModel->...)

S model.mudlibPath
 - Mudlib directory path (GA::Obj::WorldModel->...)

S model.mudlibExtension
 - Mudlib file extension (GA::Obj::WorldModel->...)

F model.paintAllRoomsFlag
 - Paint existing rooms (GA::Obj::WorldModel->...)

F model.locateRandomInRegionFlag
 - Locate after random region exit (GA::Obj::WorldModel->...)

F model.locateRandomAnywhereFlag
 - Locate after general region exit (GA::Obj::WorldModel->...)

S model.pathFindStepLimit
 - Path-finding step limit (GA::Obj::WorldModel->...)

L msdp.generic.list
 - List of generic MSDP variables (from GA::Session->msdpGenericValueHash)

S msdp.generic.VAR
 - Value of generic MSDP variable 'VAR'; if 'VAR' contains space characters, write them as underline characters, e.g. 'PAY_FOR_PERKS' (from GA::Session->msdpGenericValueHash)

L msdp.custom.list
 - List of custom MSDP variables (from GA::Session->msdpCustomValueHash)

S msdp.custom.VAR
 - Value of custom MSDP variable 'VAR'; if 'VAR' contains space characters, write them as underline characters (from GA::Session->msdpCustomValueHash)

L mssp.generic.list
 - List of generic MSSP variables 'VAR' (from GA::Profile::World->msspGenericValueHash)

S mssp.generic.VAR
 - Value of generic MSSP variable 'VAR'; if 'VAR' contains space characters, write them as underline characters (from GA::Profile::World->msspGenericValueHash)

L mssp.custom.list
 - List of custom MSSP variables 'VAR' (from GA::Profile::World->msspCustomValueHash)

S mssp.custom.VAR
 - Value of custom MSSP variable 'VAR'; if 'VAR' contains space characters, write them as underline characters (from GA::Profile::World->msspCustomValueHash)

L mxp.entity.list
 - List of MXP entities (custom entities only)

S mxp.entity.VAR
 - Value of MXP entity 'VAR' (both custom and standard entities) (from GA::Session->mxpEntityHash)

L plugin.init
 - List of plugins to load at startup (GA::Client->initPluginList)

L plugin.load
 - List of plugins loaded (from GA::Client->pluginHash)

O plugin.NAME
 - Axmud plugin object called 'NAME' (from GA::Client->pluginHash)

o prof.NAME
 - Profile called 'NAME' (from GA::Session->profHash, or else GA::Client->worldProfHash)

L prof.priority.list
 - Profile priority list in current session (GA::Session->profPriorityList)

F protocol.echo
 - Use ECHO protocol (GA::Client->useEchoFlag)

F protocol.sga
 - Use SGA protocol (GA::Client->useSgaFlag)

F protocol.ttype
 - Use TTYPE protocol (GA::Client->useTTypeFlag)

F protocol.eor
 - Use EOR protocol (GA::Client->useEorFlag)

F protocol.naws
 - Use NAWS protocol (GA::Client->useNawsFlag)

F protocol.newenviron
 - Use NEW-ENVIRON protocol (GA::Client->useNewEnvironFlag)

F protocol.charset
 - Use CHARSET protocol (GA::Client->useCharSetFlag)

S protocol.ttype.mode
 - Which termtypes to send during TTYPE/MTTS (GA::Client->termTypeMode)

S protocol.ttype.name
 - Custom client name to send (GA::Client->customClientName)

S protocol.ttype.version
 - Custom client version to send (GA::Client->customClientVersion)

S protocol.ttype.sent
 - Termttype actually sent (GA::Session->specifiedTType)

F protocol.msdp
 - Use MSDP MUD protocol (GA::Client->useMsdpFlag)

F protocol.mssp
 - Use MSSP MUD protocol (GA::Client->useMsspFlag)

F protocol.mccp
 - Use MCCP MUD protocol (GA::Client->useMccpFlag)

F protocol.msp
 - Use MSP MUD protocol (GA::Client->useMspFlag)

F protocol.mxp
 - Use MXP MUD protocol (GA::Client->useMxpFlag)

F protocol.pueblo
 - Use Pueblo MUD protocol (GA::Client->usePuebloFlag)

F protocol.zmp
 - Use ZMP MUD protocol (GA::Client->useZmpFlag)

F protocol.aard102
 - Use AARDWOLF102 MUD protocol (GA::Client->useAard102Flag)

F protocol.atcp
 - Use ATCP MUD protocol (GA::Client->useAtcpFlag)

F protocol.gmcp
 - Use GMCP MUD protocol (GA::Client->useGmcpFlag)

F protocol.mtts
 - Use MTTS MUD protocol (GA::Client->useMttsFlag)

F protocol.mcp
 - Use MCP MUD protocol (GA::Client->useMcpFlag)

F protocol.msp.multiple
 - Play simultaneous sound triggers (GA::Client->allowMspMultipleFlag)

F protocol.msp.load
 - Download sound files (GA::Client->allowMspLoadSoundFlag)

F protocol.msp.flexible
 - Recognise MSP tags in middle of line (GA::Client->allowMspFlexibleFlag)

F protocol.mxp.font
 - Allow font change (GA::Client->allowMxpFontFlag)

F protocol.mxp.image.allow
 - Allow image display (GA::Client->allowMxpImageFlag)

F protocol.mxp.image.load
 - Allow loading images (GA::Client->allowMxpLoadImageFlag)

F protocol.mxp.image.filter
 - Allow image filters (GA::Client->allowMxpFilterImageFlag)

F protocol.mxp.sound.allow
 - Allow sound (GA::Client->allowMxpSoundFlag)

F protocol.mxp.sound.load
 - Allow loading sound files (GA::Client->allowMxpLoadSoundFlag)

F protocol.mxp.gauge
 - Allow gauges (GA::Client->allowMxpGaugeFlag)

F protocol.mxp.frame
 - Allow frames (GA::Client->allowMxpFrameFlag)

F protocol.mxp.interior
 - Allow interior frames (GA::Client->allowMxpInteriorFlag)

F protocol.mxp.crosslink
 - Allow crosslink operations (GA::Client->allowMxpCrosslinkFlag)

F protocol.mxp.room
 - Locator uses MXP room tags exclusively change (GA::Client->allowMxpRoomFlag)

S protocol.mode.echo
 - Current session's ECHO mode (GA::Session->echoMode)

S protocol.mode.sga
 - Current session's SGA mode (GA::Session->sgaMode)

S protocol.mode.eor
 - Current session's EOR mode (GA::Session->eorMode)

S protocol.mode.naws
 - Current session's NAWS mode (GA::Session->nawsMode)

S protocol.mode.msdp
 - Current session's MSDP mode (GA::Session->msdpMode)

S protocol.mode.mssp
 - Current session's MSSP mode (GA::Session->msspMode)

S protocol.mode.mccp
 - Current session's MCCP mode (GA::Session->mccpMode)

S protocol.mode.msp
 - Current session's MSP mode (GA::Session->mspMode)

S protocol.mode.mxp
 - Current session's MXP mode (GA::Session->mxpMode)

S protocol.mode.pueblo
 - Current session's Pueblo mode (GA::Session->puebloMode)

S protocol.mode.atcp
 - Current session's ATCP mode (GA::Session->atcpMode)

S protocol.mode.gmcp
 - Current session's GMCP mode (GA::Session->gmcpMode)

o race.current
 - Current race profile (GA::Session->currentRace)

o race.NAME
 - Race profile called 'NAME' (from GA::Session->profHash)

F recording.active
 - Recording currently in progress (GA::Session->recordingFlag)

F recording.paused
 - Recording currently paused (GA::Session->recordingPausedFlag)

L recording.current
 - Current recording, a list of strings (GA::Session->recordingList)

F recording.position
 - Current recording position (GA::Session->recordingPosn)

S regex.url
 - Regex matching valid web links (GA::Client->constUrlRegex)

S regex.email
 - Regex matching valid email addresses (GA::Client->constEmailRegex)

L script.init.list
 - List of scripts that start in every session (GA::Client->initScriptOrderList)

S script.init.NAME
 - Run mode for the initial script NAME (from GA::Client->initScriptHash)

O session.obj.NUMBER
 - Session object NUMBER (from GA::Client->sessionHash)

S session.max
 - Maximum sessions allowed (GA::Client->sessionMax)

S session.number
 - Current session number (GA::Session->number)

O session.mainwin
 - Current 'main' window object (GA::Session->mainWin)

O session.tab.default
 - Current session's default tab object (GA::Session->defaultTabObj)

O session.tab.current
 - Current session's current tab object (GA::Session->currentTabObj)

S session.tab.width
 - Default tab width (GA::Session->textViewWidthChars)

S session.tab.height
 - Default tab height in chars (GA::Session->textViewHeightChars)

O session.mapwin
 - Current session's automapper window (GA::Session->mapWin)

O session.map
 - Current session's automapper object (GA::Session->mapObj)

O session.model
 - Current session's world model (GA::Session->worldModelObj)

O session.history
 - Current session's connection history object (GA::Session->connectHistoryObj)

S session.charset
 - Current session's character set (GA::Session->sessionCharSet)

F session.login
 - Current session's login flag (GA::Session->loginFlag)

O session.mission
 - Current session's current mission (GA::Session->currentMission)

S session.queue
 - Number of queued (excess) world commands (from GA::Session->excessCmdList)

F session.quit
 - Flag set if delayed quit IVs set (from GA::Session->delayedQuitTime)

F session.freeze
 - Flag set if task loop frozen (GA::Session->freezeTaskLoopFlag)

S session.host
 - Host address actually used (GA::Session->host)

S session.port
 - Port actually used (GA::Session->port)

S session.protocol
 - Connection protocol used (GA::Session->protocol)

S session.status
 - Session status (GA::Session->status)

S session.packets
 - No. packets received from world during session (GA::Session->packetCount)

S session.time.display
 - Time text last received from world (GA::Session->lastDisplayTime)

S session.time.instruct
 - Time last instruction processed (GA::Session->lastInstructTime)

S session.time.cmd
 - Time last world command processed (GA::Session->lastCmdTime)

S session.time.outbounds
 - Time last out-of-bounds communication processed (GA::Session->lastOutBoundsTime)

S session.redirect.mode
 - Redirect mode (GA::Session->redirectMode)

S session.redirect.string
 - Redirect mode string (GA::Session->redirectString)

L sound.format
 - List of supported audio file formats (from GA::Client->constSoundFormatHash)

F sound.allow
 - Allow sound effects (GA::Client->allowSoundFlag)

F sound.ascii
 - Allow ASCII bells (GA::Client->allowAsciiBellFlag)

L sound.effect.list
 - List of sound effects (from GA::Client->customSoundHash)

L sound.effect.NAME
 - Full file path to sound effect file NAME (from GA::Client->customSoundHash)

- O task.name.NAME
 - Current task with unique name 'NAME', e.g. 'status_task_5' (from GA::Session->currentTaskHash)
- O task.recent.NAME
 - Most recently-created current task with formal name 'NAME', e.g. 'status_task' (from GA::Session->currentTaskNameHash)
- O task.init.NAME
 - Global initial tasklist task called 'NAME' (from GA::Client->initTaskHash)
- O task.custom.NAME
 - Custom task called 'NAME' (from GA::Client->customTaskHash)
- S task.package.NAME
 - Customisable task package name 'NAME' (from GA::Client->taskPackageHash)
- S task.label.LABEL
 - Customisable task label 'LABEL' (from GA::Client->taskLabelHash)
- L task.runlist.first
 - Customisable task run-first runlist (GA::Client->taskRunFirstList)
- L task.runlist.last
 - Customisable task run-last runlist (GA::Client->taskRunLastList)
- O task.current.advance
 - Current Advance task (GA::Session->advanceTask)
- O task.current.attack
 - Current Attack task (GA::Session->attackTask)
- O task.current.chat
 - Lead Chat task (GA::Session->chatTask)
- O task.current.compass
 - Current Compass task (GA::Session->compassTask)
- O task.current.condition
 - Current Condition task (GA::Session->conditionTask)
- O task.current.debugger
 - Current Debugger task (GA::Session->debuggerTask)
- O task.current.divert
 - Current Divert task (GA::Session->divertTask)
- O task.current.inventory
 - Current Inventory task (GA::Session->inventoryTask)
- O task.current.launch
 - Current Launch task (GA::Session->launchTask)
- O task.current.locator
 - Current Locator task (GA::Session->locatorTask)
- O task.current.notepad
 - Current Notepad task (GA::Session->notepadTask)
- O task.current.rawtext
 - Current RawText task (GA::Session->rawTextTask)
- O task.current.rawtoken
 - Current RawToken task (GA::Session->rawTokenTask)
- O task.current.status
 - Current Status task (GA::Session->statusTask)
- O task.current.tasklist
 - Current TaskList task (GA::Session->taskListTask)
- O task.current.watch
 - Current Watch task (GA::Session->watchTask)

- o template.NAME
 - Profile template called 'NAME' (from GA::Session->templateHash)
- o toolbar.NAME
 - Toolbar button object called NAME (from GA::Client->toolbarHash)
- L toolbar.list
 - Ordered of toolbar button object names (GA::Client->toolbarList)
- F tts.flag.allow
 - Allow text-to-speech/TTS (GA::Client->customAllowTTSFlag)
- F tts.flag.smooth
 - Allow TTS smoothing (GA::Client->ttsSmoothFlag)
- L tts.engine.list
 - List of supported TTS engines (GA::Client->constTTSList)
- L tts.obj.list
 - List of TTS configuration objects (from GA::Client->ttsObjHash)
- o tts.obj.NAME
 - TTS configuration object called NAME (from GA::Client->ttsObjHash)
- F tts.enable.receive
 - Convert text received from world (GA::Client->ttsReceiveFlag)
- F tts.flag.login
 - Don't convert text received before login (GA::Client->ttsLoginFlag)
- F tts.enable.system
 - Convert system messages (GA::Client->ttsSystemFlag)
- F tts.enable.error
 - Convert system error messages (GA::Client->ttsSystemErrorFlag)
- F tts.enable.cmd
 - Convert world commands (GA::Client->ttsWorldCmdFlag)
- F tts.enable.dialogue
 - Convert 'dialogue' windows (GA::Client->ttsDialogueFlag)
- F tts.enable.task
 - Convert (some) task window text (GA::Client->ttsTaskFlag)
- F window.main.share
 - Share 'main' windows (GA::Client->shareMainWinFlag)
- S window.main.width
 - Default width for 'main' windows (GA::Client->customMainWinWidth)
- S window.main.height
 - Default height for 'main' windows (GA::Client->customMainWinHeight)
- S window.grid.width
 - Default width for other 'grid' windows (GA::Client->customGridWinWidth)
- S window.grid.height
 - Default height for other 'grid' windows (GA::Client->customGridWinHeight)
- O window.grid.NUMBER
 - 'grid' window NUMBER (from GA::Obj::Desktop->gridWinHash)
- O window.free.NUMBER
 - 'free' window NUMBER (from GA::Obj::Desktop->freeWinHash)

S window.text.size
 - Maximum lines is a textview (GA::Client->customTextBufferSize)

S window.charset.current
 - Current character set (GA::Client->charSet)

L window.charset.list
 - Ordered list of available character sets (GA::Client->charSetList)

S window.mode.tab
 - Session tab mode (GA::Client->sessionTabMode)

F window.mode.xterm
 - Use xterm titles in tabs (GA::Client->xTermTitleFlag)

F window.mode.long
 - Use long world name in tabs (GA::Client->longTabLabelFlag)

F window.mode.simple
 - Allow simple tabs (GA::Client->simpleTabFlag)

F window.mode.toolbar
 - Show toolbar button labels in 'main' and automapper windows
 (GA::Client->toolbarLabelFlag)

F window.mode.irreversible
 - Show icon for irreversible actions in 'edit' windows (GA::Client->irreversibleIconFlag)

F window.mode.urgency
 - Show 'main' window urgency hints (GA::Client->mainWinUrgencyFlag)

F window.mode.tooltip
 - Show tooltips in session's default tab (GA::Client->mainWinTooltipFlag)

F window.confirm.close
 - Prompt user before click-close 'main' window (GA::Client->confirmCloseMainWinFlag)

F window.confirm.tab
 - Prompt user before click-close tab (GA::Client->confirmCloseTabFlag)

F window.keys.scroll
 - Page up/down/home/end keys scrolls window pane (GA::Client->useScrollKeysFlag)

F window.keys.smooth
 - Smooth window pane scrolling (GA::Client->smoothScrollKeysFlag)

F window.keys.split
 - Page up/down/home/end keys enage split screen mode (GA::Client->autoSplitKeysFlag)

F window.keys.complete
 - Tab/cursor up/down keys autocomplete instructions (GA::Client->useCompleteKeysFlag)

F window.keys.switch
 - CTRL+TAB switches window pane tabs (GA::Client->useSwitchKeysFlag)

O winmap.NAME
 - Winmap object called 'NAME' (from GA::Client->winmapHash)

S winmap.default.enabled
 - Default winmap name for 'main' windows when workspace grids enabled
 (GA::Client->defaultEnabledWinmap)

S winmap.default.disabled
 - Default winmap name for 'main' windows when workspace grids disabled
 (GA::Client->defaultDisabledWinmap)

S winmap.default.internal
 - Default winmap name for other 'grid' windows (GA::Client->defaultInternalWinmap)

S workspace.dir
 - Direction of workspace use (GA::Client->initWorkspaceDir)

S workspace.init.count
 - Number of initial workspaces (from GA::Client->initWorkspaceHash)

S workspace.init.NUMBER
 - Default zonemap name for initial workspace NUMBER (from GA::Client->initWorkspaceHash)

F workspace.grid.activate
 - Activate workspace grids (GA::Client->activateGridFlag)

F workspace.grid.permit
 - Permit workspace grids (GA::Obj::Desktop->gridPermitFlag)

S workspace.grid.block
 - Size of gridblocks, in pixels (from GA::Client->gridBlockSize)

S workspace.grid.gap
 - Maximum size of grid gaps, in blocks (GA::Client->gridGapMaxSize)

F workspace.grid.adjust
 - Grid adjustment flag (GA::Client->gridAdjustmentFlag)

F workspace.grid.correct
 - Edge correction flag (GA::Client->gridEdgeCorrectionFlag)

F workspace.grid.reshuffle
 - Grid reshuffle flag (GA::Client->gridReshuffleFlag)

F workspace.grid.invisible
 - Grid invisible window flag (GA::Client->gridInvisWinFlag)

O workspace.obj.NUMBER
 - Workspace object NUMBER (from GA::Obj::Desktop->workspaceHash)

O workspace.grid.NUMBER
 - Workspace grid NUMBER (from GA::Obj::Desktop->gridHash)

o world.current
 - Current world profile (GA::Session->currentWorld)

o world.NAME
 - World profile called 'NAME' (from GA::Client->worldProfHash)

L world.list.favourite
 L world.list.favorite
 - Favourite world list (GA::Client->favouriteWorldList)

L world.list.basic
 - Basic world list (from GA::Client->constBasicWorldHash)

F world.flag.history
 - Store connection history in world profile (GA::Client->connectHistoryFlag)

O zonemap.NAME
 - Zonemap object called 'NAME' (from GA::Client->zonemapHash)

18 GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover

Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you

must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title

Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license

notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  YEAR  YOUR NAME.  
Permission is granted to copy, distribute and/or modify this document  
under the terms of the GNU Free Documentation License, Version 1.3  
or any later version published by the Free Software Foundation;  
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.  
A copy of the license is included in the section entitled "GNU  
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the  
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.