sTAC v3.4

Welcome to sTAC v3.4, an IRC Client for Windows.

Please **read** through these sections before asking questions!

Overview

sTAC **attempts** to provide a user-friendly interface for use with the Internet Relay Chat network. This version is currently under development and has not been fully tested. ie. it contains bugs. Use at your own risk. Read the disclaimer.

sTAC has the following features:

- Coloured text to ease reading.
- A handful of options.
- Uncluttered display.
- Simple aliases.
- Configurable multi-level popup menus.
- DCC Send/Get/Chat/Resume.
- Finger client.
- Remote CTCP tool.
- Simple Event handler.
- Some basic fonts.
- Ident server.
- Simple File-server.
- Many other useful bits and pieces.

The various parts of this client have been designed with the aim of simplifying and speeding an IRC session.

This client implements only a subset of all IRC2 commands. It was designed for users who do not require the sophistication of a full IRC2 client. That said, it does support, to various degrees, more than 50 commands, and also includes other sTAC-specific commands.

This client does NOT support: scripts, command history, and a **large** number of other things.

File Menu

The required setup information can be accessed and set through this menu.

Connect
Setup
Add
Ident Server

Misc Menu

The main options in sTAC can be accessed and set through this menu.

Options	- general options
Aliases	- how to define aliases
Popups	- how to configure popup menus
Fonts	- some basic fonts
Extras	- other bits and pieces

DCC Menu

sTAC supports DCC (Direct Client To Client) Send, Get, and Chat which allows it to make a direct connection to another client, bypassing the IRC network. sTAC also has a DCC Resume capability which allows you to resume a file transfer in the event that it was interrupted.

DCC Send
DCC Get
DCC Resume
DCC Chat
DCC Options

NOTE:
DCC Send and Chat need to use your local host IP address to initiate a connection with a client. If sTAC was unable to get your local host name, you will not be able to use these commands and they will be grayed out in the menu. sTAC will not retry getting your local host name unless you disconnect from IRC and try to reconnect. sTAC will always try to get the local host address whenever you try to connect to IRC if it doesn't already have it.

Windows Menu

The Tile, Cascade, and Arrange Icons menu items work the same way they usually do in windows.

Beneath these, a list of the windows which are currently open is shown.

Help Menu

Contents

Where you are.

About

Where I am.

Misellaneous stuff

sTAC has a few bits and pieces, as well as some quirks, that you should be aware of...

The F1 function key
Remember that you can press F1 in a dialog to get information on it.

User Information
If you are using the mirc.ini that came with sTAC then selecting the UCentral menu item in the user listbox on a channel will popup a user information window. This window allows you to save the nick, name, and email address of the user for future reference.

Channel Information
If you are on a channel, **double-click** with the left mouse-button **in the main window** and a channel information window with current modes and a ban list will appear. You can only change any of the settings if you have Op status. Note that this function sends a "/mode # +b" command to the server to get ban information about the active channel. This means that sTAC has to wait for the server to reply before it can display the ban list. If the server is busy, this may take a while! sTAC acknowledges your double-click by printing the message "Retrieving channel info..." in the status window.

Text Wrap
Text wraps according to the width of the window in which it is printed. If you later enlarge a window that was small, the text already there is not reformatted to the width of the window. Only the new text will wrap to the new width. sTAC tries to wrap lines at whole word boundaries and looks for characters such as fullstops, commas, question marks, etc. at which to wrap a word to the next line. If it can't find any of these, and the word at the end of a line is too long, it doesn't move the whole word to the next line but splits it into two.

Channel List Window
When you use the /list command to get a list of all of the available channels, a new list window opens up which lists the channels in alphabetical order. To join one of the channels, double-click on it. You can also right click to bring up a small popup menu.

Saving options
sTAC saves the main window position and all other settings when you exit the program. Note that the sizes (and not positions) of the finger, dcc get/chat/send, channel list, etc. windows are saved. The next time one of these windows opens up, it will be the same size as the last time you used it.

Multiple Instances
Currently, you cannot run multiple instances from the same EXE. However, as a temporary method you can have two copies of mirc, one mirc.exe and the other mirc2.exe, and you will then be able to connect to two different servers at the same time.

Multiple MIRC.INI files
You can specify the name of an INI file on the command line. This allows you to use different INI settings for sTAC depending on the kind of servers you use. So you could run mirc.exe mirc.ini (which will default to looking in the windows directory) and also run mirc2.exe c:\comms\mirc\mirc2.ini.

3D Dialogs

You can force sTAC to use the ctl3dv2.dll library to give the dialogs a 3D look by specifying the -d switch on the command line when you run sTAC. For example, you would use mirc.exe -d.

The Bouncing Dot
Where oh where can it be? :)

Other things I can't remember...!

The Author

I am **Khaled Mardam-Bey**, residing at 68 Melbury Court, London W8-6NJ, United Kingdom. I can be reached via Email at khaled@mardam.demon.co.uk or xhlec@wmin.ac.uk. You can also see my weird and wonderful homepage at http://www.wmin.ac.uk/~xhlec.

Feel free to send me comments, suggestions, and bug reports. However, I can't guarantee that I'll be able to reply, implement new ideas, or even fix bugs. sTAC was created for my own personal use and it now does more than what I want it to do, so I'm quite happy with it. Nonetheless, I will try to release new updated versions periodically.

You are under no obligation **whatsoever** to pay for this program but if you'd like to make a contribution then please mail me.

I hope you enjoy using sTAC as much as I enjoyed creating it! :)

Disclaimer

Options

Connect on startup
The client tries to connect to the default IRC server automatically when it is run.

Show user addresses
Whenever a user joins/parts/quits/is kicked/etc. from a channel, you can choose to see their address by selecting this option.

Actions are purple
If this is checked, any actions in a channel window appear in purple. Otherwise, all actions appear in black like normal channel text. I've left this as an option because there are quite a few colours being used in a channel window already and adding purple might overload the senses a little bit for some people!

Show quits in channel
Normally if a user on the same channel as you quits IRC, this message is only printed in the status window. If you want the quit message to be printed in the channel window as well then select this option.

Highlight own messages
If this option is selected, anything you type into a window will be highlighted.

Whois on query
Select this to have sTAC do a /whois nickname on any person that sends you a private message. The /whois will be done the first time the query window is opened.

Prefix own messages
If this is selected, your nickname will be prefixed to any messages you type in a channel/query/chat.

Rejoin channel when kicked
If you are kicked from a channel, the client will immediately try to rejoin the same channel.

Cancel away on keypress
If you set yourself as away (using /away <message>) then selecting this option cancels the away option automatically if you type a message to a channel or a query/chat window.

Beep on window message
Whenever anything is written in a window that isn't currently the active window, sTAC will beep. Note that this applies only to Channel, DCC Chat, and Query windows. You can change individual beep settings for each window via the system menu.

Iconify query window
If someone sends you a query, the default is for the query window to open, ready for input. You can select this option to force sTAC to iconify the window preventing it from taking the focus from the window you are currently in.

Beep on message while in buffer
Selecting this option will make sTAC beep if someone speaks on a channel while you are scrolling back through the previous lines in the buffer.

Use dedicated message window

This directs all private messages or notices from other users to one single message window. You will need to use the /msg command to reply. If you want to open a query window to a user, use the /query window.

Alternate Join/Part/Quit format
Checking this option makes sTAC display the join, part, and quit messages in a different, more compact format.

Beep on highlighted word
This works in conjunction with the highlight option (explained below). If a matching word appears and this option is checked then sTAC will beep.

Reconnect on disconnection
If an IRC server disconnects you without you having typed /quit then checking this command will make sTAC reconnect automatically.

Select one of the following options and enter it's corresponding text in the edit box. You can also specify whether the option is active by checking the active checkbox.

Perform commands on logon
You can enter a set of commands in this box which will be performed every time you connect to IRC. To understand how commands work, see the Aliases help section. Note, the commands here are independent of any of the aliases defined in the Alias window.

Highlight lines with these words
On a channel, any lines which contain words matching your list will be highlighted in a dark brown. This eases locating messages directed at you or about you eg. you could place your nickname (or variations of it) in this box. Each word MUST be seperated by only a space, not commas.

Notify presence of these nicks
The client will check to see if the specified nicknames are currently on IRC and will inform you if they have come on or have left. It will pause the specified number of seconds between each check. The minimum number of seconds is 20. You can force notify to check by using the /notify command by itself. You may add/remove users to the notify line at the command line by typing "/notify nick1 nick2 ...". You can turn notify on and off by typing "/notify on" and "/notify off" respectively. You will also be informed if no users in your list are on IRC when you first connect.

Auto-Op these nicks
If you are on a channel and you have channel Op status, any users that match the given nicknames will be automatically given Op status when they join the channel. Since this isn't very safe, you can specify a users address instead of their nickname. You can use wildcards.   You may add/remove users to the auto-op line at the command line by typing "/auto nick1 nick2 ...". You can turn auto on and off by typing "/auto on" and "/auto off" respectively.

Protect these nicks
If a user de-ops a nickname in your protect list, then the user is automatically de-opped. If a user kicks a nickname in your protect list, then that user is also de-opped and kicked. If you do not have Op status, no action is taken. You may add/remove users to the protect line at the command line by typing "/protect nick1 nick2 ...".You can turn protect on and off by typing "/protect on" and "/protect off" respectively.

Ignore these nicks
Messages from the nicknames/addresses in this list will be ignored. (ie. you will not see them). You may add/remove users to the ignore line at the command line by typing "/ignore nick1 nick2". You can turn ignore on and off by typing "/ignore on" and "/ignore off" respectively.

Ctcp finger reply
The message a user recieves when they /ctcp finger you.

Quit message
The message displayed when you quit IRC.

Aliases

sTAC allows you to define simple aliases to speed up your IRC session. The current implementation allows fairly unsophisticated compound terms to be stored. Each unique alias must be entered on a single line. See the following examples to understand how they work. To use these aliases you must know some IRC commands.

Alias Examples

/gb /join #gb

Typing "/gb" is now the same as typing "/join #gb".

/j /join $1

We have now added a parameter string. If we type "/j #gb", then that's the same as typing "/join #gb". The $1 refers to the first parameter in the line that you supply.

/yell /me $2 $1

if you now type, "/yell There! Hello"   the action command will be   "/me Hello There!"
The number after $ specifies the number of the parameter in the string that you entered.

/jj /join $?

the question mark indicates that the client should ask you for that parameter. The parameter you supply will be inserted in the line at that point. So if you type "/jj", a dialog pops up asking you for the channel you want to join. If you enter "#gb", then the final command will be, "/join #gb".

/jj /join #$

the # sign indicates that the parameter you specify should be prefixed with a hash indicating that it is a channel.

/jj /join $?="Enter channel to join:"

This does the same thing but now the dialog will have the "Enter channel to join:" line displayed inside it.

/aw /away $?="Enter away message:" | /say $!

This is similar to the line above except for the addition of the $! parameter. This refers to the text you just typed into the parameter box. ie. the away message. This saves you having to type the same message twice.

/give /me gives $$1 a $$2

The double $$ means that this command will only be executed if the parameter is given. If you give only one parameter in the above command it will not be executed. You can also do $$?1 or $?1 which means try to fill this value with parameter one if it exists. If parameter one doesnt exist, ask for it. In the first case the parameter is necessary for the command to be executed, in the second case it isn't.

```
/multiping /ctcp $* ping
```

The $* means that this command will be executed for ALL parameters individually. So if you typed "/multiping nick1 nick2 nick3 nick4" then each individual nick will be ping'd.

```
/slap /me slaps $1 around with *2
```

The * indicates that everything following (and including) parameter 2 should be appended to the command line. if you type, "/slap Sheepy a large trout", the final line will be "/me slaps Sheepy around with a large trout".

If you had defined the alias as "/slap /me slaps $1 around with $2", then the final command line would have been "/me slaps Sheepy around with a".

```
/laugh /me laughs at $1's joke
```

Anything appended to a $ parameter is appended to the final parameter. So if in the above example we type /laugh funnyguy, the final command would be "/me laughs at funnyguy's joke".

```
/silly /say Hel $+ lo th $+ ere $+ !
```

Parameters are normally seperated by a space. To prevent sTAC from inserting a space before appending the next parameter you can use $+. The above line will send "Hello there!".

```
/p /part #
```

The # sign refers to the channel you are currently on. So if you are on channel #blah, and you type "/p", then the client replaces the # sign with #blah, and the final command is "/part #blah".

```
/op /mode # +o $1
```

To op someone you can now just type   "/op goat" instead of the whole /mode command.

```
/dop /mode # -ooo $1 $2 $3
```

You can now deop three users by typing   "/dop goat mike bongo"

For multiple commands we use a "|" (the shifted character near the bottom left of your keyboard near the shift key). So to write an alias that de-ops, kicks, and bans someone:

```
/dkb /mode # -o $1 | /kick # $1 | /mode # +b $1
```

Notes:
1. If you create an alias that has the same name as an existing IRC command, the alias takes priority over the command (but does not replace it). So if you define an alias as "/join /join #gb", this will work fine.

2. An alias cannot call itself or another alias.

3. Popup menus cannot call aliases defined in the alias window. They are independent!

Popups

sTAC allows you to create **three** different types of popup menu: one for the main window, one for query/chat windows, and one for the listbox where users nicknames on the current channel are listed. To use these you must know some Basic IRC commands.

If you press the right mouse-button, the popup menu will appear and you can select menu-items which you have defined to perform certain tasks, such as Opping a user or joining a channel. In fact, each menu-item is associated with a set of aliases, just like the alias window. However, note that the popup menu-item **cannot** call the commands defined in the alias window. They are independent.

You can create as many levels of popup menus as you want. ie. menus within menus.

To understand how popup menus can be defined, see the following examples.

On a single line, enter:

```
Get Help:/join #irchelp
```

The words before the ":" (colon) are the name of the menu-item. The words after the ":" are the commands that are to be performed. In this case, the menu-item you would see when you click the right mouse-button in the main channel window is "Get Help". The command that would be performed would be "/join #irchelp", as if you had typed it.

The format of the commands follows precisely the same as those in normal aliases. See the aliases help section to understand how to write an alias.

To create a **Sub-menu**, use a "." (a fullstop/period). Thus:

```
Join a Channel
.Get IRC help!:/join #irchelp
.Visit the folks at #friendly:/join #friendly
.Who shall we join?:/join $?
```

In this case, the name of the sub-menu is "Join a Channel". All the commands following it beginning with a "." are part of the submenu that fits under this menu-item.

To create menus within a submenu, you just add more fullstops:

```
Join a Channel
.Fun Stuff
..Type1 Fun
...Fun1:/join #fun1
...Fun2:/join #fun2
..Type2 Fun
...Fun2:/join #fun2
...Fun3:/join #fun3
.Help
..Get IRC help!:/join #irchelp
..Help1:/join #help1
..Help2:/join #help2
.Other Channels
```

```
..Visit the folks at #friendly:/join #friendly
..Wibble Wobble:/join #wibble
.Who shall we join?:/join #$$?="Please enter a channel name:"
```

And another example:

```
whois ?:/whois $?
-
Misc
.Edit Temp:/run notepad.exe temp.txt
.say?: /say $?
.action?:/me $?
Names
.#irchelp: /names #irchelp
.#friendly: /names #friendly
.names ?:/names $?
-
channel list:/list
-
Join a Channel
.Get IRC help!:/join #irchelp
.Visit the folks at #friendly:/join #friendly
.Who shall we join?:/join $?
```

The above menu would probably be suitable for the main channel window.

NOTE: You can seperate menu items by placing a single "-" (minus sign) on a line by itself.

To use the popup menu for the nickname listbox, you need to select a nickname before the menu will pop up. The following example follows the same design as the one above, except that it is oriented towards performing commands on a nickname. Here is a typical listbox popup menu:

```
Who Is?:/whois $1
-
Modes
.Give Ops:/mode # +o $1
.Deop:/mode # -o $1
.Deop, Kick, Ban:/mode # -o $1 | /kick # $1 | /ban $1
-
DCC Send:/dcc send $1
DCC Chat:/dcc chat $1
-
Slap!:/me slaps $1 around a bit with a large trout
Query:/query $1 Hey you! hello? are you there...?
```

Activation
To activate/deactivate popup menus, turn these the Active option on or off.

The popup menu for the **Query/Chat window** works more or less the same as the channel listbox popup menu. The $1 always refers to the nickname of the user with whom you are talking.

Double-Click in...
You can enter a set of commands that will be executed whenever you double-click in the

specified window. This is just a set of commands, not an alias or a popup menu definition! For example, you could use:

```
/query $1 I have been searching for thee wit (sic) me perishing scopes...
```

as the entry for the channel nickame which means that when you double-click on a nickname, the query window opens up and the user receives the above message.

Extras

The options in this menu are more or less after-thoughts... thus the highly descriptive title. Since there are many options that could still be added, they will probably all end up here until they grow to warrant a seperate dialog box all to themselves.

Notification
Certain events cause a number of beeps to be made to alert you of their occurence. Since this might be annoying to some users, it is possible to alter the number of beeps as well as how quickly they are sounded. The second value is in milliseconds. To turn off notification all together, specify zero beeps.

Beep on...
If you are **disconnected** by the server without having typed /quit or selected the disconnect menu item, then sTAC will beep to indicate that you are no longer connected to IRC. You can also select to be beeped if you receive a Query, Notice, or Invite.

Edit Box Size
In single line mode, the text scrolls to the right when you reach the end of the edit box. In multi-line mode, the text starts again at the left and scrolls downwards. It's probably easier to use multi-line mode since you can see what you've typed if the line is long. This mode also allows you to paste several lines of text into the editbox.

NOTE: The status window permanently has a single line edit box.

Colour
You can have sTAC use your windows colours, however it will only display monochrome text. This is because sTAC cannot know which other colours are compatible with the background window colour you've chosen.

Line separator
You can specify a line seperator to be used in the status window. You can use a space to have a blank line. If you leave the box empty, lines in the status window will not be seperated.

Use small fonts...
Selecting this option will make sTAC display text in the Aliases, Popups, and Remote windows using a small font instead of the regular font.

ESCape key minimizes window
To have a window minimize whenever you press the ESCape key select this option.

Scrollback buffer size
This limits the scrollback buffer to the specified number of lines. Note that if a scrollback buffer already contains more than the specified number of lines it **will** be shortened. You can always use the /clear command in a window to clear the scrollback buffer completely. If you are scrolling back through the buffer, lines will not be removed until you return to the bottom of the buffer.

Append this text to the application title bar
This allows you to specify text that will be shown in the sTAC application title bar.

Automatic Logging

You can have sTAC automatically log the channel and query/dcc chat windows whenever they open up.

Log and Buffers directory
The directory in which all log files and buffer saves are stored.

Future Work

This is **UNSUPPORTED** software. I have a life to live!

Feel free to send me comments, suggestions, and bug reports. However, I can't guarantee that I'll be able to reply, implement new ideas, or even fix bugs. sTAC was created for my own personal use and it now does more than what I want it to do, so I'm quite happy with it. Nonetheless, I will try to release new updated versions periodically.

sTAC is not meant to be highly-configurable or comprehensive. It's meant to be quick, simple, and user-friendly, providing the user with the basic functions of an IRC client. I have gone to a lot of trouble to make things transparent... I hope no one notices.

My interpretation of protocols and commands may not be correct. If it appears that sTAC is not performing a function in a standard way, which you feel is important, tell me and I'll see if I can remedy the problem.

Acknowledgements - Thank you, thank you, and thank you...

Writing this client would have been **impossible** without the availability of documentation, source code, and help from many people. In an attempt to understand the workings of winsock and windows routines, source code from many sources was mutilated and recombined in weird and wonderful ways. Several months and 1000's of lines of code later, I find it difficult to know not only who to thank but also how much. So, in no particular order, here goes nothing:

Thanks to...

Winsock help files, RFC1459, IRC Faqs, IRC source codes, and a multitude of other documents, as well as all of the source code for winsock and windows programming in the Public Domain, I couldn't have done it without you. I must have looked at a million zip files. Here is a small list of source codes which I found useful in helping me clarify ideas, understand protocols, and in some cases to become even more confused! These are the few sources I can remember. I also used several different programs to help me trace/experiment with winsock calls as well as other sources in Pascal, C, C++, and Basic. I'm sorry I can't remember all of you, thanks!

Peter R. Tattam, whose WinIRC was the inspiration for this program, specifically it's lack of an /action command! It's quite possible that had WinIRC had an action command, I would never have bothered writing this client.

Nicolas Pioch, for his short IRC primer, portions of which were used to write the IRC commands section.

Dimashq, who kept on asking for new options to be added to the program... remind me to stay away from you the next time I code something ;)

Bibbly, whose accute knowledge of RFC's directed me to the one for IRC, saving me from having to download them all ;)

Stimps, who showed no mercy and uploaded me a slightly large file explaining hows scripts work...

Tjerk, whose comments and help in beta-testing have improved sTAC in many ways...

James G., who answered question... after question... after question...

Bunster, for providing me with an unlimited supply of delicious life-giving queen-size futons...

Viv, may a zillion rose petals land gently on thee.

and ofcourse... the beta-testers and the folks who bothered to send me bug reports and suggestions :)

also to M.Hunnibell for the ident server suggestion and help, M.J.M. for his user interface comments and code, M.Overtoom for the ctl3d dialog code, Mark "Too Slick" Hanson for debugging the DCC routines, and...

all the other wonderful, heart-warming, kind folks who directed me at different people,

places, source codes, and documents, discussed ideas with me, and in general both supported and laughed at my humble efforts. Thanks! :)

Installation

The sTAC package consists of the following files:

sTAC.exe
sTAC.hlp
sTAC.ini
readme.txt
versions.txt

You can put **sTAC.exe** and **sTAC.hlp** together in any directory you want. These are the only required files. I would also suggest that if this is the first time you are running sTAC, you should quit the program, copy the sTAC.ini file to the windows directory, and then re-run sTAC. The **sTAC.ini** file comes with useful examples and pre-written aliases and popup menus that will help you understand how to configure these options.

If you are an old user of sTAC, read the versions.txt file to find out what has been changed/fixed.

To use sTAC you will need a **Winsock.dll** of which several versions exist (both commercial and shareware) and a connection to the internet. If you don't have either of these, ask someone about them!

Some Basic IRC Commands

sTAC does not come with a tutorial and assumes that you are already somewhat familiar with IRC. For those with limited IRC experience, the following list of commands should help you get started. Note that you can join channel #irchelp to get help if you are confused about a command. There are also sTAC-specific commands you can look at.

Once you have connected to an IRC server, you can try some of the following commands:

/JOIN #channel
Join the specified channel.

example: /join #irchelp

This will join you to the #irchelp channel. Once on a channel, anything you type will be seen by all the users on this channel. The #irchelp channel is very useful, so say hello and then ask any questions you want. If the channel you specified doesn't exist, a channel with that name will be created for you.

/PART #channel
Leave a channel.

example: /part #irchelp

/LIST [#string] [-MIN #] [-MAX #]
Lists currently available channels. You can also tell sTAC to show only channels with a minimum and a maximum number of people. If you specify a #string then sTAC will only list channels with that string in their title.

example: /list
example: /list -min 5 -max 20
example /list #love

/ME message
Tells the current channel or query about what you are doing.

/MSG nickname message
Send a private message to this user without opening a query window.

/QUERY nickname message
Open a query window to this user and send them a private message.

/WHOIS nickname
Shows information about someone.

/NICK nickname
Changes your nickname to whatever you like.

/QUIT [reason]
This will disconnect you from IRC and will give the optional message as the reason for your departure. (this message only appears to people who are on the same channels as you).

example: /quit That's all folks!

/AWAY [away message]
Leave a message explaining that you are not currently paying attention to IRC. Whenever someone sends you a MSG or does a WHOIS on you, they automatically see whatever message you set. Using AWAY with no parameters marks you as no longer being away.

example: /away off to get something to eat, back in a moment!

/TOPIC #channel newtopic
Changes the topic for the channel.

example: /topic #friendly Oh what a beautiful day!

/INVITE nickname #channel
Invites another user to a channel.

Channel and User Control

The following commands give you control over both a channel and the users on that channel (assuming you have Op status). It's probably wise not to abuse either of them.

/KICK #channel nickname
Kicks named user off a given channel.

example: /kick #gb Ed

/MODE #channel|nickname [[+|-]modechars [parameters]]
This is a powerful command that gives channel operators control of a channel and the users on it.

Channel modes

ModeChar	Effects on channels
b <person>	ban somebody, <person> in "nick!user@host" form
i	channel is invite-only
l <number>	channel is limited, <number> users allowed max
m	channel is moderated, (only chanops can talk)
n	external /MSGs to channel are not allowed
o <nickname>	makes <nickname> a channel operator
p	channel is private
s	channel is secret
t	topic limited, only chanops may change it
k <key>	set secret key for a channel

User modes

ModeChar	Effects on nicknames
i	makes you invisible to anybody that does not know the exact spelling of your nickname
o	IRC-operator status, can only be set

	by IRC-ops with OPER
s	receive server notices
v	gives a user a voice on a moderated channel

Here a few examples of the MODE command:

To give someone Op status: /mode #channelname +o nickname

Giving someone Op status means giving them control over the channel and the users on it. Give this out sparingly and to people you trust.

To op several people: /mode #channelname +ooo nick1 nick2 nick3

To de-op someone: /mode #channelname -o nickname

To ban someone: /mode #channelname +b nickname (or user address)

example: /mode #animals +b Jiminy
example: /mode #tree +b joe@bloggs.edu

To Unban someone: /mode #channelname -b nickname (or user address)

example: /mode #gb -b Ed

To Make a channel invite only: /mode #channelname +i

You must now **invite a user** for them to be able to join your channel.

There many more commands but this list should help you get started.
To learn more about IRC commands you should download an IRC FAQ.

Good luck!

Fonts

sTAC allows you to select a different font for each type of window. These can be changed at any time both while connected or while offline. Only the five basic system fonts are currently supported.

The default font can be changed for a specific window via its System Menu.

Tools Menu

Some basic tools...

Finger
Timer
Remote

DCC Resume

This allows you to resume DCC transfers that failed to complete.

This will only work with another sTAC client.

If a user tries to send you a file that already exists in your get directory then you will be shown a DCC Get dialog warning that the file exists. The dialog gives you the option to either overwrite, resume, or rename the file.

If you select **overwrite** then the whole file will be downloaded from the beginning and any existing file of the same name will be erased.

If you select **resume** then sTAC will attempt to negotiate a transfer resume to get the remaining part of the file. It will append this to the portion of the file you already have.

The negotiation method is specific to sTAC. It is not standard and will not work with other DCC implementations (especially since most do not have resume capability). The negotiation is automatic, and once the receiving user clicks the resume button, the transfer will commence as normal.

Here is a description of the sTAC DCC Resume Protocol.

Connect

Select this menu item to initiate a connection to IRC. You must have input the required information in the Setup window. This menu item changes to "Cancel connect" while attempting to connect to an IRC server, and it changes to "Disconnect" while you are connected.

Setup

You must enter basic information about yourself in this dialog. The client will not connect if any of the fields are empty.

Real Name and Email Address
The first entry can be either your real name or a witty one liner which will appear in your /whois information. You must also enter a **full** email address eg. khaled@mardam.demon.co.uk. The username part of the email address is used to register with the server.

Nickname and Alternate
As well as entering a regular nickname, you can also enter an alternate nickname so that if the first nickname is in use when you try to log on, the alternate will be tried. If both nicknames are in use, sTAC inserts "/nick" into the edit box so that all you have to do is enter a new nickname and press enter. These values cannot be changed while connected to a server.

Local Host
This is used to register with the server and may be the part of your email address after the @ sign, eg. if my email address is khaled@mardam.demon.co.uk, then I would enter mardam.demon.co.uk here.

IP address
This will normally be filled in by sTAC and is there mainly for your information. sTAC looks up your IP address and stores it in the mirc.ini file for future reference. This way it doesn't have to look it up every time you want to connect.

Always get IP address on connect
This should be selected if you have a dynamic IP address. If you're not sure, leave it on.

NOTE: If sTAC is having trouble getting your IP address then you can enter this value manually and sTAC will assume that it is correct. If this value is wrong you will still be able to log on to IRC but you will not be able to initiate DCC Send/Chat sessions (you will only be able to accept them).

IRC Servers
You can also build up a small list of IRC servers that you use regularly. Type in the IRC server address and the port number (usually 6667) and click on Add Server. If you need to specify a password, you may do so after the port number by appending a colon ":" and then the password. The last server you connected to will be placed at the top of the list. Note that you can use a servers IP address if you know it.

Connect
If you click on connect, sTAC will attempt to connect with the currently selected IRC server. If you are currently connected to a server, you will be disconnected before connecting to the new one.

Ident Server

sTAC can act as an ident server and sends the specified User ID and System as identification. This server will be more useful to some people than others. In general it is better to leave it active as some systems might refuse a connection if there is no reply to an ident request.

User ID can be your account or user name on your system. For most people this will be the User ID portion of their email address (the text before the @ sign).

System identifies your operating system. For all intents and purposes, replying with a value other than UNIX would not be very useful for most people.

Port should usually be 113.

This server will reply to ALL ident queries sent to the specified port ie. it is not limited to replying to an IRC server for sTAC. Note that this server will not work with some internet setups.

Finger

This fingers a persons address to find out more information about them. If a user has their own personal address, this will only work if they run a finger server. Most other addresses eg. school, government, etc. can usually be fingered sucessfully, though they will not necessarily give you any useful information. For some users this can be a way of checking if they have mail.

You can also type   /finger <address> at the command line, eg. /finger xhlec@wmin.ac.uk

Timer

The online timer is displayed in the status window next to your nickname. You can choose to have the timer reset to zero every time you connect so that you can tell how long you are on for that session, or If you select the second option, the timer will be cumulative and will show the total amount of time you have used IRC since the last reset.

Remote

The Remote Tool allows you to specify both how sTAC replies to CTCP queries from other users and how it reacts to certain kinds of IRC events. This tool is the most complex part of sTAC and requires that you understand the format of aliases and IRC commands precisely.

With this tool you will be able to let sTAC respond to almost any remote command you can think of... in fact, any command that you can type can be executed remotely. You can do things like give your friends Op status or invite them to a channel at their request, you can give them access to your files, or you can let them control your sTAC in other ways.

You will also be able to set up sTAC to react to events such as users joining channels, responding to queries with certain words in them, and opping or deopping users automatically.

The Remote window
Special Identifiers
Access Levels and Commands

The following examples describe, with increasing complexity, how the remote tool can be configured.

Example 1: How to change replies to standard CTCP queries...
Example 2: How to Op, Invite, and Unban your friends...
Example 3: How to control sTAC remotely...
Example 4: How to offer files for people to DCC...
Example 5: How to listen for events...

Other notes

The Remote window

The remote window consists of

- the **Users** list where you keep a list of all authorized users and their access levels,
- the **Commands** list where you keep a list of all commands and their access levels,
- the **Events** list where you keep a list of events to which sTAC should react,
- the **Listen** switch which turns the remote on and off, and
- the **Default level** number which controls the number of commands to which unlisted users have access.

sTAC also understands the following commands:

/remote [on | off] which switches the remote on and off.

/dlevel <level> which changes the default user level eg. /dlevel 2

/auser <level> <nick | address> - add a user with the specified access level to the remote list (this replaces any existing user of the same nick/address) eg. /auser 1 khaled

/guser <level> <nick> [type] - does a /who on the user to find out their address and adds it to the user list with the specified level and address type (same as the /ban address types).

/ruser <nick | address> - remove this user from the remote users list eg. /ruser khaled

/ulist [< | >]<number> - lists users in the remote list with the specified access levels. For example:

/ulist <10	lists users with access levels less than or equal to 10
/ulist >5	lists users with access levels larger than or equal to 5
/ulist 4	lists users with access level 4

Access Levels and Commands

Access levels define which users can access which commands.

The **default access level is 1** for users that are not listed in the Users list. All users can access level 1 commands. The higher a user's access level is, the more commands that user can access. You can change the default user level to allow unlisted users to access more commands (see the /dlevel command). However, Users that **are** listed will still have the same access level regardless of the default level setting.

In the **Users** window you can specify a list of users and their access levels in the format:

<access level> : <user nickname/address>

For example:

```
2:*!*jimmy@widgets.com
5:*!*parrot@crackers.inc
19:*!*xhlec@wmin.ac.uk
```

You can specify the users nick or the users nick!user@address. You can also use wildcards.

In the **Commands** window you specify a list of commands and their access levels in the format:

<access level> : <ctcp command> : <alias> : <parameters>

For example:

```
1:MYLEVEL:/notice *1:$nick You are weak! You have at least access level 1
5:MYLEVEL:/notice *1:$nick You are a friend! You have at least access level 5
7:MYLEVEL:/notice *1:$nick I trust you completely! You have at least access level 7
50:MYLEVEL:/notice *1:$nick You are the boss! You have at least access level 50!
```

The command "MYLEVEL" can be executed by anyone who has an access level equal to or greater than it's corresponding access level.

If jimmy does a /ctcp yournick MYLEVEL, the reply will be "You are weak! You have at least access level 1". For parrot the reply will be "You are a friend! You have at least access level 5", and for xhlec the reply is "I trust you completely! You have at least access level 7".

The remote always replies with the **highest level command** that matches the users ctcp query and that is within the users access level.

Limiting Access

Note that the following control prefixes are sensitive to positioning. It is best to set up your definitions starting with the lowerst access levels at the top of the list and increasing numerically towards the bottom of the list.

You can **prevent** a user from accessing any levels by prefixing their level with an **equal sign**:

```
Users:
=150:bannednick!user@not.nice.inc

Commands:
150:*:/notice *1:$nick Go away, you are banned from using this remote.
```

bannednick now has access **only** to command 150 and will always get this reply. Notice that the ctcp command is a * which means that this will match **any** ctcp command that this user sends.

Likewise, you can **limit commands to users with a certain access level** by prefixing a command with a **plus sign**, for example:

```
Users:
3:*!*rocky@crackers.inc
10:*!*nicky@crackers.inc
17:*!*wacky@crackers.inc

Commands:
+10:OPME:/notice *1:$nick Hi nicky! Only users with level 10 can access this command!
```

You can also **prevent users with higher access levels from accessing lower access level commands** by using a **equal sign** in the following way:

```
Users:
17:*!*nicky@crackers.inc

Commands:
3:HELPME:/notice *1:$nick Help is unavailable!
4:HELPUS:/notice *1:$nick Here is your help...
10:HELPME:=
```

This would prevent users with access levels higher than 10 from accessing matching commands with levels lower than or equal to 10. Thus, nicky cannot access the level 3 HELPME but she can access the HELPUS command.

You can **prevent a command from reacting to a command sent by you** by using the the exclamation mark in the following way:

```
Users:
17:*!*nicky@crackers.inc

Commands:
3:HELPME:/notice *1:$nick Help for level 3 is unavailable!
10!:HELPME:/notice *1:$nick Help for level 10 is unavailable!
```

If nicky has this definition in her remote setup and sends herself a /ctcp nicky HELPME then even though she has access level 17, the level 10 command will be skipped and she will get the reply from the level 3 command.

How to change replies to standard CTCP queries...

Here are simple examples that show how you can customize your normal ctcp replies for finger, ping, time, etc... You do not need to modify your remote Users list, just copy the following lines to your Commands list, and turn on the Listen option in the remote window:

```
1:PING:/notice *1:$nick Hee hee, that tickles! :>
1:FINGER:/notice *1:$nick If you want to know something about me... ask!
1:TIME:/notice *1:$nick Time to get a real watch! :>
1:VERSION:/notice *1:$nick I'm using sTAC v3.2! :>
1:USERINFO:/notice *1:$nick You have reached a non-working number, please try again...
```

Description:

1. The number 1 at the beginning of each line specifies the access level required to execute this command. All users have default access level 1 if they are not specified in the users list, so all users can execute level 1 commands.

2. The word following the number 1 eg. PING is the ctcp query that the user has sent you. It is not case-sensitive.

3. The /notice *1 is the alias command. You should already know how to write aliases if you are reading this section. If not, read the aliases help section first!

4. The $nick is the nickname of the user who sent you the ctcp query. Other variables you can use are $address (which is the address of the user), $parms (the parameters that the user sent), and $me (your own nickname).

5. The :> at the end of a line indicates that the normal reply to this ctcp query should **also** be sent. If you do not add the :> then only your reply is sent.

How to Op, Invite, and Unban your friends...

This simple example shows how users can ask your remote to execute an IRC command for them.

Users:
2:friendA
5:friendB
6:friendC
9:friendD

Commands:
1:WHOAMI:/notice *1:$nick You are $nick your address is $address and you sent parameters $parms
1:OPME:/notice *1:$nick Sorry, you do not have the required access level for this command
2:INVITE:/invite $1 $2:$nick $parm1
5:UNBAN:/mode $1 -b $2:$parms
7:OPME:/mode $1 +o $2:$parm1 $nick

Description:

1. If any of your friends send you a "/ctcp yournick WHOAMI", the remote will reply with their nick, address, and any parameters they sent.

2. All of your friends can send you a "/ctcp yournick INVITE #channel" and the remote will invite them to the channel they specified (assuming you are on that channel).

3. friendB, friendC, and friendD can send you a "/ctcp yournick UNBAN #channel friendX" and if you have Op status on #channel then the remote will unban them.

4. friendD can send you a "/ctcp yournick OPME #channel" and if you have Op status on #channel then the remote will give friendD Op status. All other users will get the level 1 OPME reply.

How to control sTAC remotely...

```
Users:
10:yournick!*@youraddress

Commands:
10:PART:/part $2 | /notice $1 *3:$nick $parm1 I have left channel $parm1
10:QUIT:/notice *1 | /quit:$nick Okay boss, I'm quitting... see you later!
10:SEND:/dcc send $1 $2:$nick $parms
10:DO
```

Description:

1. If you send a "/ctcp yournick PART #channel", the remote will part the specified channel and tell you that it has done so.

2. If you send a "/ctcp yournick QUIT", the remote will quit IRC and tell you that it has done so.

3. With the SEND command, you can ask the remote to send you a file. So if you send a "/ctcp yournick SEND info.zip", the remote will dcc send you this file.

4. DO is the most powerful command in the remote. It executes any IRC command you send to it. This allows you to perform commands remotely without having to define them first. For example:

```
/ctcp yournick DO /join:#mirc
/ctcp yournick DO /quit:See you later folks...
/ctcp yournick DO /mode:#mirc +o yourothernick
/ctcp yournick DO /dcc send:yourothernick info.txt
```

WARNING: This command gives authorized users complete control of sTAC...! You should not let **anyone** have access to this command unless you trust them!

How to offer files for people to DCC...

This example shows how the remote can reply differently for levels of users. Level 1 users cannot request files from you while level 2 users can. Only a level 3 user can get the special file. The level 3 user can also get files #1 and #2.

```
Users:
1:jimmy!*@widgets.com
2:parrot!*@crackers.inc
3:*!xhlec@wmin.ac.uk

Commands:
1:CMDS:/notice *1:$nick Existing commands are : cmds, mylevel, mycmds, xdcc
1:MYCMDS:/notice *1:$nick Commands available: cmds, mylevel, mycmds
1:MYLEVEL:/notice *1:$nick You have access level 1

2:MYCMDS:/notice *1:$nick Commands available: cmds, mylevel, mycmds, xdcc
2:XDCC LIST:/notice *1:$nick Files offered: #1 readme.txt(4K) ; #2 utils.zip(132K)
2:XDCC SEND #1:/dcc send $1 $2:$nick c:\files\readme.txt
2:XDCC SEND #2:/dcc send $1 $2:$nick c:\files\utils.zip
2:XDCC SEND:/notice *1:$nick You must specify file #1 or #2
2:XDCC:/notice *1:$nick Try XDCC LIST
2:MYLEVEL:/notice *1:$nick You have access level 2

3:XDCC LIST:/notice *1:$nick Hello $nick !! Files offered: #1 readme.txt(4K) ; #2
utils.zip(132K); #3 special.zip
3:XDCC SEND #3:/dcc send $1 $2:$nick c:\files\special.zip
3:XDCC SEND:/notice *1:$nick You must specify file #1, #2, or #3
3:MYLEVEL:/notice *1:$nick You have access level 3
```

Special Identifiers

In the parameters supplied to an alias command you can use the following identifiers to refer to certain values. Don't worry if you don't understand how these are used right now, they'll be explained in the examples.

$me refers to your own nickname

$nick and $address refer to the nickname and address of the user sending the ctcp query

$parms refers to the **whole** parameter line that the user sends with the ctcp query.

If you want to refer to particular parameters within the user supplied parameter line, you can use:

$parm#   where # is a number   eg. $parm1   or $parm2

To refer to all parameters from a certain position:

$parmN*   where N is the parameter position. eg. $parm3*

You will see later that it is important to specify a parameter number in certain situations. For example, if you define a command that requires only 1 parameter and the user sends 2 parameters then this could affect the way the command works.

Other notes...

Another example of a multiple command is:

```
5:XDCC SEND #1:/notice $1 *3 | /dcc send $1 $2:$nick somefile.zip I acknowledge your XDCC SEND #1
```

As you can see, things can get a bit messy! It is wise to test out every command you define to make sure it does what you want and **nothing else**. For example:

```
5:OPME:/mode $1 +o $2:$parms $nick
```

With this definition, if a user supplies more than one parameter then $2 will refer not to the users nickname but to the second parameter that the user supplied (which is in $parms). The remote will then Op a different user from the one it was intended to Op. You should redefine the command as:

```
5:OPME:/mode $1 +o $2:$parm1 $nick
```

This makes sure that only the first parameter that the user supplies is used in the final command executed by the remote.

Redirection

You can also tell the remote to execute several commands by using the :> switch. This is useful mainly because it saves you from having to retype similar commands. For example:

```
5:HELP1:/notice *1:$nick This is help1 :> HELP2
5:HELP2:/notice *1:$nick and help2 :> HELP3 :> HELP4
5:HELP3:/notice *1:$nick and help3
5:HELP4:/notice *1:$nick and help4
```

Be careful you don't do this:

```
5:HELP1:/notice *1:$nick This is help1 :> HELP2
5:HELP2:/notice *1:$nick and help2 :> HELP1
```

As this will result in an infinite loop. The remote does NOT check for this. You've been warned!

The Shared Parameter Line

Remember, the parameter line is shared by any aliases in a definition, so:

```
1:TEST:/notice *1:$nick hello | /notice *1:$nick goodbye
```

is wrong!

```
1:TEST:/notice $1 $2 | /notice $1 $3:$nick hello goodbye
```

is correct. Unfortunately, this format makes it difficult to specify which parameters you want for the first and second aliases... some combinations would be messy and might even be impossible. An easy way to get around this problem is to use a redirection:

```
1:TEST:/notice *1:$nick hello there! :> TEST2
1:TEST2:/notice *1:$nick How have you been?
```

Pass through commands

You can allow only certain commands to be executed by doing this:

```
5:PING
5:VERSION
5:*:!
```

The remote will now only allow replies to PING and VERSION. All other **level 5** commands will be ignored. The exclamation mark ! at the end of the line tells the remote to halt any further processing of level 5 commands.

Commenting out commands and events

You can comment out definitions by prefixing them with a ; (semi-colon) or REM.

```
5:PING1
rem 5:PING1
;5:*:PING1
```

Ordering of definitions
Remember that many of the prefixes and controls are sensitive to numerical order of the definitions. The safest thing is to order your definitions starting with the lowest access levels first and increasing numerically down the list.

DCC Send

DCC Send allows you to send a file to a user by specifiying their nickname and the name of the file you want to send. Once you have input the nickname and the filename, select "Send" and sTAC will send information to the nickname you specified telling them that you want to send a file. The user then has to accept your send request, at which point the file transfer should begin.

You can also select **multiple files** to send at one time but only up to a maximum of nine.

If you check **minimize** then the dcc send window(s) will be minimized automatically.

The **Packet Size** is the number of bytes that sTAC will send to another client in one packet. The minimum is 512, the maximum is 4096. Each DCC Send session can have a different packet size... just change it before you start the DCC Send. sTAC can receive packets up to 4096 bytes.

NOTE: There is an experimental DCC Fast Send option which you can try out.

DCC Chat

DCC Chat allows you to talk privately by connecting directly to another client, bypassing the IRC network. Enter the nickname of the user and select "OK" and sTAC will send information to the nickname you specified telling them that you want to DCC Chat. The user then has to accept your chat request, at which point you can talk with them normally as if you were in a query window on IRC. If a user replies to a /dcc chat request by initiating another dcc chat with you then sTAC treats this as an acceptance of it's own request.

If a user sends you a chat request, a dialog will popup asking you whether you want to talk to them. You can then accept or decline.

DCC Get

The DCC Get dialog only appears in response to a send request from another user. If you choose to accept the file that the user wants to send you, sTAC will tell the sender to begin the file transfer, at which point you should begin receiving the file.

DCC Options

Show Get Dialog/Auto-get file
By default, a send request must be accepted by you before transfer begins. However, if you select the auto-get file option then sTAC will automatically accept a send request and begin receiving a file. It will also **minimize** the receive window if you select the minimize option. You can also have sTAC close the get window automatically after a tranfser is complete by selecting the Auto-close get window   option.

On transfer completion...
sTAC will perform the selected options once a transfer has been completed. sTAC will indicate in the status window whether the transfer was a success or a failure. The Notify with beep depends on the settings in the Extras window.

Show progress as percent
For DCC Get, this depends on whether the sending client provided the size of the file when it initiated the send session. If no file size was sent, then percentage cannot be displayed and only the bytes received are shown.

Show Chat Dialog/Auto-accept chat
By default, a chat request must first be accepted by you before chatting begins. However, if you select the auto-accept option then sTAC will automatically open a DCC chat window and accept the chat request. It will also **minimize** the chat window if you select the minimize option.

Get/Chat Dialog Time Out
When a user sends you a Send or Chat request, a dialog pops up and waits for you to accept or decline. The dialog will wait the specified number of seconds before disappearing.

Send/Get Transfer Time Out
During a transfer sTAC will wait the specified number of seconds for a response from the other client before closing the connection.

Fileserver Time Out
sTAC will close the fileserver window if a user has been idle for the specified number of seconds.

Max. DCC Sends
This limits the number of simultaneous **remote** DCC Sends. ie. it does not apply to you manually initiating a DCC Send but it does apply to users who request a DCC Send remotely.

Max. Fileservers
This limits the number of users that can be served simultaneously.

DCC Get Directory
The directory where received files are stored.

DCC Resume Protocol

In it's current form the protocol is very simple.

User1 is sending the file.
User2 is receiving the file.

To initiate a DCC Send, User1 sends:

PRIVMSG User2 :DCC SEND filename ipaddress port filesize

Normally, if User2 accepts the DCC Send request, User2 connects to the address and port number given by User1 and the file transfer begins.

If User2 chooses to resume a file transfer of an existing file, the following negotiation takes place:

User2 sends:

PRIVMSG User1 :DCC RESUME filename port position

filename = the filename sent by User1.
port = the port number sent by User1.
position = the current size of the file that User2 has.

User1 then responds:

PRIVMSG User2 :DCC ACCEPT filename port position

This is simply replying with the same information that User2 sent as acknowledgement.

At this point User2 connects to User1 address and port and the transfer begins from the specified position.

Add

sTAC allows you to specify another INI file from which the Aliases, Popups, and Remote Users, Commands, and Events settings will be loaded. All of your other settings from your mirc.ini file will be loaded as usual except for these. Note that if you then edit any of these settings they will be NOT be saved to your mirc.ini file but to the INI file you specified here.

If you want to load in the Aliases, Popups, and Remote Users, Commands, and Events settings from another INI file while sTAC is running you should use:

/add [-apuce] filename.ini

where the switches a, p, u, c, and e indicate which sections should be loaded. If you do not specify any switches, all five sections are loaded. You can also use:

/save filename.ini

which will save all five sections into the specified INI file. Please note that doing a /save does **not** make the specified INI the default where future settings will be saved. In all cases, your original mirc.ini file be untouched unless you do a /save mirc.ini.

How to listen for events...

The events window allows you to specify how sTAC reacts to certain types of events. The set of events currently available is limited to the most basic types, each of which will be given simple examples to illustrate how they work.

The event handler listens to events caused by users specified in the Users List (which it shares with the Commands list). Each event has an access level which is matched to a users access level. The default level works the same as in the commands list. You can also define normal command defintions in this window.

The examples cover each type of event:

The ON TEXT event listens to messages from users either on certain channels or in private. It can also look for specific words in messages.

The ON JOIN/PART events react to users joining or parting a channel.

The ON KICK event occurrs when a user is kicked from a channel.

The ON OP/DEOP events occur when a user is opped or deopped.

And a few Other Events which react to various situations.

ON TEXT

Keys:

?	= Any private message
#	= Any channel
#mirc	= Channel #mirc
*	= Both private and channel messages

Where **text** is:

*	= any text
=text	= if user said only this word
text*	= if user started line with this word
*text	= if user ended line with this word
text	= if user said this word anywhere
text	= if user said this word anywhere

To listen for messages on channels...

```
1:ON TEXT:hello*:#:/msg *1:$nick Hello back!
```

This command listens for the message "hello" from any user on any channel and replies with the message "Hello Back!".

```
1:ON TEXT:hello*:#sTAC:/msg *1:$chan Welcome to #mIRC!
```

This command listens for the message "hello" from any user on channel #MIRC and replies with the message "Welcome to #sTAC!".

```
5:ON TEXT:goodbye:#sTAC:/msg *1:$chan Goodbye $nick!
```

This command listens for the message "goodbye" from any user with access level 5 on channel #MIRC and replies with the message "Goodbye NickName!"

To listen for private messages...

```
1:ON TEXT:=help:?:/msg *1:$nick Please use /ctcp $me help for more information...
```

This command listens for a private message with the word "help" in it from any user and replies with the message " Please use /ctcp mynickname help for more information..."

```
1:ON TEXT:*:?:/msg *1:$nick I'm not here! Go away!
```

This command listens for any private message from any user and replies with the message "I'm not here! Go away!"

To listen for any messages...

```
1:ON TEXT:*:*:/msg *1:$nick Not Listening...
```

This command listens for any message both private or public from any user and replies with the message "Not Listening..."

The **ON NOTICE** event works exactly the same way as the ON TEXT event.

NOTE: You cannot test out this event by typing text to yourself! It is initiated by someone else saying something in a channel or in a private message.

ON JOIN/PART

You can have sTAC perform certain actions whenever a user joins a channel...

```
2:ON JOIN:#:/msg *1:$chan Welcome $nick!
```

When a user with access level 2 joins any channel, you say "Welcome nickname" on the channel.

```
10:ON JOIN:#sTAC:/kick $1 $2 | /msg *2:$chan $nick And OUT of $chan $nick goes! :)
```

When a user with access level 10 joins channel #sTAC, you immediately kick them out and send the above message to them.

```
14:ON PART:#:/msg *1:$chan Yahoo! $nick is gone! ;)
```

When a user with access level 14 parts any channel you tell everyone how happy you are :)

ON KICK

```
2:ON KICK:#:/kick $1 $2 | /invite $3 $4 | /msg $2 *5:$chan $nick $knick $chan That person is my friend!
```

When a certain user is kicked out of any channel you kick the person who kicked them, send them a message, and invite the person who was kicked back to the channel.

```
2:ON KICK:#sTAC:/notice *1:$chan Hey! Wake up! $nick was just kicked with the message " $parms "
```

When a certain user is kicked from a channel you send yourself a notice telling you what just happened.

ON OP/DEOP

These two events are concerned with operator status. The event is performed for each user that is opped/deopped:

```
100:ON OP:#:/mode $1 -o $2 | /msg *3:$chan $opnick $nick Don't Op that person!
```

When a user with access level 100 is opped, you immediately deop them and send a mesage to the person that opped them.

```
9:ON DEOP:#mirc:/mode $1 +o $2 | /msg *3:$chan $opnick $nick Hey! That person is my friend!
```

When a user with access level 9 is deopped, you immediately op them and send a mesage to the person that deopped them.

```
1:ON SERVEROP:#:/mode $1 -o $3 | /notice *2:$chan $me $opnick was opped by $nick!
```

When a user is opped by a server I deop the user and send myself a notice. The $nick refers to the server that did the opping.

Note that **these events work on nicknames and not addresses**. This is because the IRC server only sends the address of the user doing the opping/deopping and not of the users affected ie. it only sends the **nicknames** of the users being opped/deopped.

Other Events

```
3:ON NOTIFY:/msg *1:$me Hey, wake up! $nick just joined IRC!
```

When a user joins IRC and you have their nick in your notify list (and the notify is turned on) then this will react to the notification.

```
2:ON INVITE:#sTAC:/join $1 | /describe *1:$chan appears in a puff of smoke!
```

When a user invites you to channel #sTAC, you automatically join the channel and send the above action to it.

```
2:ON NICK:/describe *1:$newnick thinks $nick was a nicer nickname!
```

When a user changes their nickname, you tell them you thought their last nickname was nicer.

```
2:ON QUIT:/notice *1:$me Hey! Wake up! $nick just quit with the message " $parms "
```

When a user quits IRC you alert yourself with a notice.

```
2:ON TOPIC:#:/describe *1:$chan admires $nick's new topic!
```

When a user changes the channel topic...

DCC Fast Send

DCC Fast Send is a simple and experimental extension to a normal dcc send. The aim was to see if it was possible to write a dcc send algorithm that would a) increase transfer speed, and b) remain compatible with the standard dcc get routine.

Basically, dcc fast send attempts to maintain a steady flow of packets by sending slightly ahead of acknowledgements. It does not require the receiving client to implement a special DCC Get algorithm. ie. it will/should work with any client.

By varying the packetsize (using /dcc packetsize <size>) and turning on the fast dcc send option, it is possible to achieve a fair increase in transfer speed. The packetsize is crucial to finding a good balance for maintaining a steady packet flow. 4096 bytes is **not** necessarily the best choice. It is probably better to use something like 2048 bytes.

As an example, sending to a unix client with 4096 byte packets and the fast send option turned **off,** a transfer speed of 0.9k/sec was achieved, while with the fast send option turned **on,** 1.5k/sec was achieved, almost double the speed. Sending from one sTAC to another sTAC can also give a comparable increase in speed.

However, how well it works really depends on both the senders and receivers internet setup and connection speed.

You can turn DCC Fast Send on and off using: /fsend [on|off]

Note:
1.This option has been included just to see if it really does work for most people. It's possible that it might not increase transfer speed for you and in fact, it might even make things slower.
2.This option is always turned **off** whenever you start sTAC. ie. the setting is not saved. Again, I did this because I'm still not sure if it is robust/practical/etc. If you find it works for you then you can always add "/fsend on" to the Options->Perform section to turn it on automatically whenever you connect to a server.

sTAC-Specific Commands

The following commands are either sTAC-specific or slightly modified standard commands...

/ recalls the previous command entered in the current window. sTAC can remember **only** the last command that you typed for that window. If you type **/!** this recalls the last command typed in any window.

/add is used to add sections from another INI file.

/amsg and **/ame** send the specifed message or action to all channels which you are currently on.

/auser and related commands are listed in the remote section.

/ban bans someone from the current channel using their address. To do this, it first does a /who on the user, which gives it the user's address, and then it does a /mode # -o <user address>. There are four levels of ban, the format is:

/ban [#channel] <nickname> [type]

Where the number ranges from 1 to 4, least to most severe respectively. If you do not specify a ban type, then sTAC uses the whole nick!user@address to do the ban. If you are banning an IP address then a wild card replaces the last number of the IP address. If you are on a channel then the #channel specification is not necessary.

/beep beeps a number of times with a delay, the format is: /beep <number> <delay>

/channel pops up the channel central window (only works in a channel)

/clear clears the entire scrollback buffer of the current window.

/dcc send can take multiple file names, the format is:

/dcc send <nickname> <file1> <file2> <file3> ... <fileN>

This will initiate multiple dcc send sessions to the specified user.

/exit forces sTAC to closedown and exit.

/finger does a finger on a users address.

/flood is a crude flood control method. The format is:

/flood <numberoflines> <seconds> <pausetime>

This translates to: if sTAC has sent a certain number of lines to the server within a specified number of seconds then prevent sTAC from sending anything more to the server for the specified number of seconds. Lines which are prevented from being sent are **lost**.

/flood 10 5 3

If sTAC has sent 10 lines in the last five seconds, prevent any further sending to the server for 3 seconds.

/fsend [on | off] allows you to turn dcc fast send on or off.

/fserve initiates a fileserver session to another user. See the FileServer section.

/help brings up the Basic IRC Commands section in the sTAC help file.

/log allows you switch logging on and off for a window by specifying an on or off paramater.

/omsg and **/onotice** send the specified message to all channel ops on the current channel. You must be a channel operator to use these commands.

/play allows you to send text files to a window. It will only play to a window which is currently open. ie. you cannot play to a channel without first joining it. If you want /play to send actual irc commands then use the -c switch. The format is:

/play [-c] <filename> [delay]

The delay is in milliseconds. If you play files too quickly to a server you will probably be disconnected for flooding. The default setting is 1000 ie. 1 second. Empty lines between text are treated as a delay.

/play c:\text\mypoem.txt 2000

/raw sends any parameters you supply directly to the server. You **must** know the correct RAW format of the command you are sending. Useful for sending commands which sTAC hasn't implemented yet.

/raw PRIVMSG nickname :Helloooo there!

/run allows you to run the specified program with parameters.

/ftp /run c:\comms\ftp\ftp.exe sunsite.unc.edu

This runs the ftp program with the parameter sunsite.unc.edu.

/edit /run notepad.exe $?

This asks you for a parameter and runs notepad using the parameter as the filename.

/say lets you define an alias that writes directly to a channel as if you were saying something. So "/say Hello there" would be the same as just typing "Hello there". This is useful in an alias when you want to ask the same question (or send the same information) again and again.

/info /say Please note that the games server is currently down and will be offline for a few hours...

/server is implemented in the following way: /server irc.server.co.uk 6667 password

If you type /server with no parameters, sTAC will connect to the last server you used. If you use the server command while still connected, you will be disconnected with your normal quit message and will then connect to the specified server.

/timer activates the specified timer (of which there are five) to perform the specified command at a specified interval. The format is:

```
/timer[N] <repetitions> <interval in seconds> <commands>
```

```
/timer1 0 20 /ame is AWAY!
```

Timer1 will repeat an all channel action every 20 seconds until you stop the timer.

```
/timer5 10 60 /msg #games For more info on the latest games do /msg GaMeBoT info
```

Timer5 will repeat this message to channel #games every sixty seconds and stop after 10 times.

You can use the above methods to call an alias you have defined in the alias window. If you supply the timer with multiple commands seperated by | (see aliases) then you cannot call aliases you have defined. for example:

```
/timer4 100 30 /names #irchelp | /names #mirc
```

To see a list of active timers type /timers. To see the setting for timer1 type /timer1. To deactivate timer1 type /timer1 off. If you are activating a new timer you do not need to specify the timer number, just use:

```
/timer 10 20 /ame I'm not here!
```

And sTAC will allocate the first free timer it finds to this command.

/uwho pops up a user window showing information about the specified user. It is the same information you would get if you did a "/whois nickname". You can also build up a list of nicknames and their associated real name and machine address. The format is: /uwho nickname

Other features...

Please read the following sections, they **are** important!

The File Server

The **/fserve** command opens up a fileserver session using a DCC Chat to the specified user. You must specify a homedirectory. The user will be limited to accessing only files and directories within this homedirectory. The format is:

/fserve <nickname> <maxgets> <homedirectory> <welcome text file>

The maxgets is the maximum number of **simultaneous** dcc gets that the user can have during a fileserver session. The welcome text file is text that is sent to the user when they first connect. For example:

```
/fserve goat 5 c:\users\level1 level1.txt
```

This will initiate a filserver session to user goat with his homedirectory as c:\users\level1 and will send goat the text in the level1.txt file (presumably informing him that he is a level1 user and what files he can access etc.). The user can only have 5 simultaneous gets.

In each directory, you can place a **dirinfo.srv** file which describes that directory. Everytime the user does a CD to change into a directory, sTAC will look for this file and if it finds it, the text in it will be sent to the user.

The commands availabe to a user once connected are:

cd - change directory.

dir [-b|k] [-#] [/w] - lists the name and size of each file in the current directory. The /w switch forces a wide listing. The [-b|k] selects bytes or k's. The [-#] specifies the number of files on each line in a horizontal listing.

ls [-b|k] [-#] - lists the name of each file in the currenty directory using a wide listing.

get <filename> - asks the fileserver to DCC Send the specified file.

read <filename.txt> - reads the specified text file. Note that only files ending in .txt can be read.

help - lists the available commands.

exit or **bye** - terminates the connection.

These commands have been greatly limited in the hope that this will prevent a security breach.

NOTE:
1.It will improve performance a lot if you make sure that your directories are not too large. If a directory has a large number of files try to split them up into subdirectories.
2.If a user is idle too long the fileserver will automatically close the connection. You can set the idle time out in the DCC Options dialog.
3.A user is limited to opening a **single** fileserver session at any one time. If sTAC initiates a fileserver session to a user and that user doesnt respond then the fileserver session will have to time-out and close before that user can ask for another session.

System menu

If you click the **system menu** button in the top left hand corner of a window (ie. the button you usually double-click to close a window), it will popup the usual system menu but with a few added functions (these vary depending on the type of window):

Window: You can tell sTAC to **remember or forget the position and size** of the status and channel windows. If they are remembered, the next time a window opens up, it will do so in the saved position. If you choose the **reset** menu item, the window will be moved back to it's previously saved position. Note that if a windows saved position lies outside the size of the main window then it will open in a default position and size. To force a window to open up in default positions assigned by windows, select **forget.**

Buffer: You can **clear** the text in the current window buffer or you can **save** the text to a file. The filename is automatically taken from the name of the window.

Font: For the channel, query, and dcc chat windows, you can select fonts that are different from the default fonts in the font setup dialog. The font settings for each window will be remembered across sessions. For the other windows, this changes their default font.

Logging: If you select this, the text to the window will be logged to a file. This setting stays on across sessions until you switch it off. This function is only available for the status, channel, and query windows. The filename is automatically taken from the name of the window.

Beeping: This appears only in Channel, DCC Chat, and Query windows. If selected then sTAC will beep any time a message is sent to the window if it isn't active. This setting is remembered across sessions for each window.

Text Copy and Paste

To copy text from a window, you mark the text as usual with the mouse by pressing the left mouse-button and dragging it. The moment you release the left mouse-button, the text will be copied into the clipboard.

To paste text you can then do the usual Shift-Insert key combination to paste the text any where you want.

The limitation: You can only copy the currently displayed text. To copy text from another page, you must scroll up/down to it and then copy it. If you want to store most of the text you see on a channel, you might want to use the logfile/buffer options in the System Menu.

The explanation: the use of colour in sTAC means that a simple text box cannot be used since text boxes can display only plain text (and they also have other limitations). However, text boxes also have built in cut/copy/paste routines which unfortunately are unavailable to a graphic window. This means that I had to code the mark/copy routine myself. I'm not sure which ran out first, my patience or my programming ability :-)