MTS (mIRC Theme Standard) 1.3 Help

(Based on: MTS DRAFT 8.4 - VERSION 1.3)

Contents

CHAPTER 1 - THE MTS CONCEPT

Why a theme standard? How is this different from other "standards" people have tried to come up with in the mIRC community?

A theme standard is needed in the mIRC community. There are many themes out there for various scripts, and no way to simply use them in another script. If a themer creates a theme using a standard, then it can be loaded in all scripts supporting that standard- allowing all scripts and users to benefit from the talents of themers.

This standard was created from the input of many scripters and themers, many who have already planned to, and have implemented earlier versions of this standard.

Most scripters are aware that there are basically two different ways to implement a theme system in mIRC, although there are a few variations of each type.

The first type of theme is the "scripted" theme system, where basically a script file is /loaded by a theme alias, and this new script file contains all of the code to handle the text processing for the script. The script simply calls aliases in the theme file to handle each event.

Pros: This is very flexible, since everything is scripted.

Cons: It is more difficult for a themer to make a scripted theme because they must know at least a bit of mIRC scripting. It is very difficult to write scripts that modify these themes, such as theme editors. The method used for these aliases also differs wildly from script to script.

The second type of theme is the "keyword" theme system. This is implemented by having a text file or ini file that is read into the script in some manner.

The theme system then replaces keywords in the theme lines with correct values whenever an event occurs, and displays that.

Pros: Simple and easy to edit for themers. Theme editors can be made easily for these types of themes.

Cons: You are limited in your flexibility, since there is no actual scripting in the theme files. You're also limited to what the script allows in a theme, and you cannot (usually) do multi-line displays or other fancy themes.

MTS is a combination of both types, for maximum flexibility. It is what you might expect if you started with a text theme system, and then added support for scripted aliases on top.

Remember however that MTS is a standard for theme files, not for a theme engine. What this means is that when you see a bug in a theme or engine, it is not a bug in MTS, rather it is a bug in the theme or theme engine.

CHAPTER 2 - HOW MTS WORKS

A basic run down of what happens during an event that an MTS script wants to format or "theme"-

1) The script catches the event in mIRC, using a script event or raw.

2) The script checks the theme file to see what the theme wishes to do for this event. If the theme has a simple line of text for this event-

a) The script replaces special codes (variables) in the line of text, such as nickname, channel, colors, etc.

b) The script then displays the line of text to the screen.

If the theme instead refers to an alias name-

a) The script sets a number of mIRC %::variables to values such as nickname, channel, colors, etc.

b) The script sets %:echo to the command it wishes to use for theme output, such as "/echo -t $chan"

c) The script calls the theme alias, which formats and displays output using these variables.

.

CHAPTER 3 - BASIC FORMAT FOR AN MTS THEME FILE

All MTS theme files should have an extension of .MTS. MTS files are simple text files, where each line contains one event, raw, setting, or other piece of theme-related information.

You may have blank lines, and any line beginning with a semicolon (;) will be treated as a comment and ignored.

Before any settings, the first line in your MTS file must be "[mts]". This is related to scheme usage. (Chapter 9) Even though scheme support is optional and not part of every theme, all themes must have the [mts] line to mark the main MTS data section of the file.

Themes may refer to external files such as images or a .MRC file. These files must be in the same directory as the theme .mts file. For example, ultra.MTS refers to ultra.MRC and BK.PNG. These files would be in the same directory as the ultra.mts file.

IT IS HIGHLY RECOMMENDED THAT .MTS FILES BE PUT IN THEIR OWN SUBDIRECTORY SO

THAT SUPPLEMENTAL THEME FILES DO NOT CONFLICT WITH OTHER THEMES.

A script may load a theme and process in any way it desires, as long as it achieves the effects detailed in this document. Some behaviors or events may not apply to a given script and can be ignored. A script may also voluntarily ignore certain features, such as images, if desired, but this may diminish the value of some themes.

The decision was made to use plain text files instead of .ini files for a number of reasons:

1. Text files have no 64k limit, as ini files do.

2. INI files can not use mIRC's text formatting codes - they are stripped out.

3. Reading in an INI file in mIRC is often more complex than a text file because of later processing that must be done on MTS theme lines.

4. Text files are easier to read both programming-wise and visually.

CHAPTER 4 - FORMAT OF EVENTS AND RAWS IN MTS THEME FILES

Each event or raw has one line in an .MTS file. The line starts with a code that signifies the event or raw. The remainder of the line is the text displayed for that event, but will contain special variable <codes> that will get replaced when the actual event occurs. For example-

TextChan [<nick>] <text>

This is a basic theme line for text in a channel window. When it is displayed, the script will replace <nick> with the nickname of the user, and <text> with what they said.

Some (or all) events may instead refer to aliases or lines of code. For example-

TextChan !Script theme.chantext

This tells the script that, when a text event happens for a channel, to call the /theme.chantext alias (in the theme's .mrc file, specified by the Script line - see chapter 6). When this alias is called, it will then handle the echoing of the text, using the %:echo variable (see chapter 5), as well as theme variables that have already been set by the calling script. For example, %::nick will be the nickname, and %::text will be what the text was from the event. Notice how these correspond with <nick> and <text> above- this is true for all theme variables.

A special variable, %:echo, contains the command the alias should use to display it's text. A short example alias that could be part of a theme's script file:

```
alias theme.chantext {
%:echo ( $+ $lower(%::nick) $+ ) %::text
}
```

These aliases are stored in a separate .MRC file. (this file is mentioned on the SCRIPT line in the .MTS file- see chapter 6 for details.) When the script sees the "!Script" prefix, it should take the remaining text and run it as if it were an alias name, but it should $eval() it first, so that it runs as an actual line of mIRC code. (see chapter 9 for details.) In other words, something such as "!Script %:echo %::text" should be allowed to work.

Note that "!Script" by itself should not be an error- it should simply "do nothing", essentially preventing any output.

This covers the basics of how a script should handle a theme- Check the line, and either replace <variables> and display it, or set %::variables and call the listed alias. Note that when replacing <variables>, you should also replace <lt> and <gt> with < and > so a theme can use those characters. (see chapter 9 for details.)

CHAPTER 5 - VARIABLES USED IN EVENTS AND RAWS

The script should set the variables listed here before calling a theme alias. Note that all variables, other than %:echo and %:comments, have a corresponding <variable> that should be replaced when processing normal theme lines. Some variables may not apply to a given event or raw, of course. Some variables are listed multiple times when they have different uses in different situations.

All-purpose variables: (these must be set in all events and raws)

%:echo Command being used to display text. Use in place of /echo.

%::me Your nickname.

%::server The server you are currently connected to. (if connected)

%::port The port you are currently connected to. (if connected)

%::pre Set to the value of Prefix. (see Prefix option.)

%::c1 Base color 1. (see BaseColors option)

%::c2 Base color 2. (see BaseColors option)

%::c3 Base color 3. (see BaseColors option)

%::c4 Base color 4. (see BaseColors option)

%::timestamp The current timestamp, using the script's Timestamp line format. Used by <timestamp> in theme files.

Variables used in events:

%:comments Used by all events. These comments are for a script to add any additional text to a line, and should therefore be appended to the end of any displayed line. A normal text line (one not using !Script) does not need to do anything with this- the script should append it automatically when used. For this reason there is no corresponding <comments> code.

%::text The text or message from the event. (if any)

For CTCP, CTCPSELF, CTCPREPLY, CTCPREPLYSELF- Additional text for the CTCP, beyond the command.

For NOTIFY, UNOTIFY- Notify note, if any.

%::parentext For KICK, KICKSELF, QUIT, PART, NOTIFY, UNOTIFY- This is the same as %::text, but surrounded with parenthesis. If there was no message, this is blank- not "()". The format for this can be changed by a script- see ParenText, chapter 6.

%::nick Nickname of user triggering event. In cases such as you sending a private message, this is who you sent it to.

%::address Address in ident@host format of %::nick, if available.

%::chan Channel event occured in.

%::cmode Current mode of %::nick on %::chan, such as @ or +.

%::cnick Current color of %::nick on %::chan, as a number.

%::target Equivalent to $target, useful for op notices and certain

other cases.

%::knick For KICK, KICKSELF- Nickname of user who was kicked.

%::kaddress For KICK, KICKSELF- Address in ident@host format of %::knick,

if available.

%::newnick For NICK, NICKSELF- New nickname for user.

%::modes For MODE, USERMODE- Channel modes or usermodes being set.

%::ctcp For CTCP, CTCPSELF, CTCPREPLY, CTCPREPLYSELF- CTCP or

CTCPREPLYcommand. (single word) The remainder of the CTCP, if any, is in %::text.

Variables used in raws:

(Note that if a raw only uses ONE of %::value, %::nick, or %::chan, then the script is allowed to place the SAME VALUE in all three, for simplicity. A theme should not rely on this behaviour.)

%::text The full text from the event, either $2- or $3-. (if $2

is placed in %::nick or %::chan, %::text should be $3-)

%::numeric Number of the raw reply being triggered.

%::value A single value of interest from a raw, usually $2.

%::nick Nickname raw applies to. (usually from $2)

%::chan Channel raw applies to. (usually from $2)

%::fromserver The server that is sending the raw line to the user- this

is determined using $nick in a mIRC raw event. This will

NOT always match %::server.

%::users For RAW.251, RAW.255, RAW.265, RAW.266- User count.

%::modes For RAW.221, RAW.324- Channel/user modes currently set.

%::address For RAW.302, RAW.352- Address of nickname. (USERHOST or WHO)

Variables for WHOIS, WHOWAS, and WHO events and raws:

(Note that there is no WHO event, just RAW.352, but it is listed as WHO below for clarity. A script is only required to use these variables in WHOIS/WHOWAS/WHO events and raws- they do not need to work in other events and raws.)

%::away WHOIS- Away message, if any.

WHO- H for Here, G for Gone. (away)

%::nick WHOIS, WHOWAS, WHO- Nickname of user.

%::address WHOIS, WHOWAS, WHO- Address of user in ident@host format.

%::chan WHOIS, WHOWAS- Channels user is on, if any.

WHO- A single channel user is on.

%::cmode WHO- Current mode of user on specified channel.

%::realname WHOIS, WHOWAS, WHO- The "full name" field.

%::isoper WHOIS, WHO- Is this user an IRCop? Set to "is" or "is not".

%::operline WHOIS- The text of the "is an ircop" line sent by the server, not including the nickname, as different users may have different levels of status described in this line.

%::isregd WHOIS- Is this nickname registered? Set to "is" or "is not".

%::wserver WHOIS, WHOWAS, WHO- IRC server the user is on.

%::serverinfo WHOIS, WHOWAS- "Info" about the IRC server the user is on.

(usually just useless text)

%::idletime WHOIS- How long the user has been idle, in seconds.

%::signontime WHOIS- When the user signed on, in $asctime() format.

%::value WHO- Number of hops user is away.

%::text WHOIS, WHOWAS- Any extra notes from a whois not covered elsewhere.

WHO- Entire /who line.

Variables for DNS events:

(Other than %::nick, a script is only required to use these variables in DNS events- they do not need to work in other events and raws.)

%::nick Nickname of user being DNS'd, if appropriate.

%::address The address being looked up. This will be equal to either

%::naddress or %::iaddress. (not %::nick)

%::naddress Named address, if available.

%::iaddress IP address, if available.

%::raddress Resolved address, if available.

List of all variables without duplicates-

%:echo %:comments

%::address %::fromserver %::newnick %::serverinfo

%::away %::iaddress %::nick %::signontime

%::c1 %::idletime %::numeric %::target

%::c2 %::isoper %::operline %::text

%::c3 %::isregd %::port %::users

%::c4 %::kaddress %::parentext %::value

%::chan %::knick %::pre %::wserver

%::cmode %::me %::raddress

%::cnick %::modes %::realname

%::ctcp %::naddress %::server

CHAPTER 6 - MTS FILE FORMAT - SETTINGS AND INFORMATION

.MTS files contain more than just events and raws- they contain other lines with information and settings. These settings are covered in this chapter. Remember that all entries in an .MTS file take up one line. Actual events and raws are in the next chapter. Also note that the setting names are not case

sensitive.

Note that MTSVERSION and NAME are the ONLY required fields. The [mts] line is also required. Everything else is optional, and a script should use default values or no values for missing items.

Informational items in an MTS file:

[mts] This must appear before any MTS data. It should appear on a line by itself. (this is to ease MTS loading by allowing use of the mIRC /loadbuf -t switch.)

Name This is the name of the theme, and is required.

Author The name or nickname of the theme's author.

Email E-mail address of the author.

Website Website for the theme or author.

Description A brief description of the theme.

Version The version of the theme. A new version of the same theme should have a value that is greater than (>) the previous version's value.

MTSVersion The version of MTS the theme is designed for. A theme will work on a later version of MTS but not an earlier one. This field is required and should be in the form "n.nn".

Color options in an MTS file:

These items allow a theme to change mIRC and theme colors.

Colors These are the mIRC /color settings to use; the order is the same as in the mirc.ini file. The format is 28 numbers, separated by commas. If not present, use the default mIRC colors. See chapter 9 for the order of colors and defaults.

RGBColors These are the RGB values to use for all 16 colors. Each color is three numbers (R,G,B) separated by commas; there are 16 such sets, separated by spaces. In other words, R,G,B R,G,B R,G,B etc. (where R,G,B are numbers) If this line is not present, you should reset all colors to their default mIRC RGB values (can be done with /color -r in mIRC)

Alternatively, a named color can be specified instead of the R,G,B combination. A listing of supported named colors can be found in Chapter 14. It is up to the theme engine to support this

BaseColors This is four colors used in the theme itself. These will be used for %::c1 through %::c4 and <c1> through <c4>. The format is four numbers, separated by commas. It is recommended that these four colors be in a certain order. (see below)

BaseColors help keep a theme "modular"- if you use them throughout a theme, you can then change the colors of your entire theme just by changing one or two color lines. (more details in chapter 9 - schemes)

Scripts may also choose to use the colors in BaseColors for their own specialized echos and displays. For that reason, the order of the four BaseColors is recommended to be as close to the following as possible-

Text color, Nickname color, Text highlight color, Bracket color

These are merely recommendations- a theme can use these for anything it wishes. The reason for the recommendation is so a theme contains reasonable colors that a script can use for it's own purposes as well. All four colors should ideally produce readable text on the selected background color.

Nicklist options in an MTS file:

The first set of items are single numbers representing the color to use for nicknames in the nicklist. Each line either corresponds to a channel mode (OP, etc.) or a special status. (such as IRCOP) Not all scripts will use all colors (or even use nicklist colors at all) and remember that themes do not need to

include all colors. A script should ideally not use any colors that are not present- in other words, if a user is both an OP and an IRCOP, but the theme does not define an IRCOP color, use the OP color.

CLineOwner Mode . on certain servers (channel owner)

CLineOP Mode @ (op)

CLineHOP Mode % on some servers (half-op or helper)

CLineVoice Mode + (voice)

CLineRegular No special mode.

CLineMe Your nickname.

CLineFriend A friend. (notify list, userlist, protected, auto-op, etc.)

CLineEnemy An enemy. (banned, ignored, blacklisted, etc.)

CLineIrcOP IRCops.

Note that the color numbers MUST BE DOUBLE DIGITS. Many themers do not do this, or forget, so it is a good idea to have error checking on these lines.

For scripts that can use /aline for custom windows (such as multiserver scripts), MTS provides for you to be able to change the mode character and color for each type of user. Keep in mind that CLineCharRegular

probably wont use a character, but if the theme wants one, then you should accomodate for it. The syntax is the exact color and mode character to be prefixed before the above CLine colors. Note that, like the other

CLine lines, you should not include the color control code itself. (Ctrl+K)

CLineCharOwner 12.

CLineCharOP 11@

CLineCharHOP 10%

CLineCharVoice 09+

CLineCharRegular

Note that the color numbers MUST BE DOUBLE DIGITS. Many themers do not do this, or forget, so it is a good idea to have error checking on these lines as well.

Font options in an MTS file:

These options define the fonts to use for windows. A script may ignore these but the theme may not look correct. Some fonts apply to multiple windows or types of windows. Each font line is a font name, followed by a comma and a font size.

The size should be a positive or negative number. To make a font bold, put ,B after the font size.

You can specify multiple fonts to be used if the others aren't available, by adding additional font names separated by the ; character, example:

FontDefault IBMPC; Terminal; Lucida Console, 12, B

This would tell the theme engine that in order of priority, IBMPC is the preferred font, and if it isn't available, then Terminal will be used, and if that isn't available, Lucida Console will be used. Keep in mind that at the time of this writing, there is not a built-in way in mIRC to determine whether a font exists.

FontDefault Font used for status window and any unspecified windows. This is basically a "default" font that can be used for any window. (for example, script windows or notify/url/finger windows)

FontChan Font for channel windows.

FontQuery Font for query, chat, and dedicated message windows.

Image options in an MTS file:

These options define image files to use for backgrounds and other areas of mIRC. A script is free to ignore some or all of these. The first word of each line defines the way to apply the background- center, fill, normal, stretch, tile, or photo. This is followed by the filename of the image. The

ImageButtons item does not have a style. The ImageToolbar and ImageSwitchbar should include a style. (although mIRC currently only supports the fill style for these, this may change in the future.)

ImageStatus Background for status window.

(you may use this for other windows if desired)

ImageChan Background for channel windows.

ImageQuery Background for query, chat, and dedicated message windows.

ImageMirc Background for main mIRC window.

ImageToolbar Image to use for toolbar.

ImageButtons Image to use for toolbar buttons.

ImageSwitchbar Image to use for switchbar.

Miscellaneous options in an MTS file:

Prefix This value is simply used for %::pre and <pre>, so that a common prefix does not need to be rewritten on every event.

ParenText This special value is used to surround part, quit, and kick messages with parenthesis. If not present, a script should use "(<text>)" for this. Whenever a part, quit, or kick occurs, <text> will contain the normal text, and <parentext> will use this. However, <parentext> will be blank if there is no message. This usage prevents events from showing empty parenthesis if there is no message.

Timestamp Set to ON or OFF. Can be removed from a theme file so as to not affect the timestamp setting, or a script can ignore this setting in a theme.

TimestampFormat mIRC's /timestamp format is set to this. Note that a script should reset this on startup, because if it contains colors then mIRC will remove them on startup.

A themer can also use the special <timestamp> variable to put a timestamp anywhere in a line they want. The theme engine should always set the %::timestamp variable when

$theme.text is called.

Script This item contains the filename of a mIRC script to load as part of the theme. This file should contain all aliases used in !Script lines. This file should be loaded using /load -rs, as it is a script file, not an alias file.

Scheme1 These lines contain names of color schemes that the theme

Scheme2 contains. See chapter 9 for details on this optional feature.

etc...

CHAPTER 7 - MTS FILE FORMAT - EVENTS AND RAWS

All possible theme events, and all common raws, are listed below. A script is not required to use all events, but most should be used for full effect. Note that if an event or raw is missing, a script should use some sort of default display. A missing event or raw does NOT mean to hide that event or raw.

Whenever an event or raw refers to a variable in %::var format, remember that the same value will be in <var> format when processing normal theme lines.

Note that only some %::vars are mentioned- others like %::nick are assumed.

Text IRC events:

TextChan Standard text in a channel.

TextChanSelf Your text in a channel.

ActionChan Action performed in a channel.

ActionChanSelf You perform an action in a channel.

NoticeChan You recieve a channel or op notice. Note that %::target

will contain @#channel for an op notice.

Notice You recieve a private notice.

NoticeSelf You send a private notice.

NoticeSelfChan You send a channel or op notice. The script should put

either #channel or @#channel into %::target to clarify.

TextQuery Private message in a query or chat window.

TextQuerySelf You send a message in a query or chat window.

ActionQuery Action performed in a query or chat window.

ActionQuerySelf You perform an action in a query or chat window.

TextMsg You recieve a private message, displayed in active or

dedicated messages window.

TextMsgSelf You send a private message, displayed in active or dedicated

messages window.

Basic IRC events:

Mode Channel mode change. %::modes has full mode string.

ModeUser You change usermodes. %::modes has full mode string.

Join Another user joins a channel.

JoinSelf You join a channel.

Part Another user parts a channel. %::text has part message, if any. %::parentext will contain the message with parenthesis, if any. (See ParenText, Chapter 6)

Kick Another user is kicked. %::text has kick message.

KickSelf You are kicked. %::text has kick message. Both this and Kick should place the message, if any, into %::parentext with parenthesis. (See ParenText, Chapter 6)

Quit A user quits IRC. %::text has quit message, if any. This event will be called once for each place it appears. (usually, once for each channel the user was in) %::parentext will contain the message with parenthesis, if any. (See ParenText, Chapter 6)

Topic Topic changed in a channel. %::text has new topic.

Nick Your or another user changes nickname. This event will be called once for each place it appears. (usually, once for each channel the user was in)

NickSelf This appears in status when you change nickname. The NICK event will be used for each channel, also.

Invite You are invited to another channel.

ServerError The server reports an error. This is equivalent to the ERROR event in mIRC. %::text has full error string.

Rejoin You attempt to rejoin a channel.

Ctcp You recieve a CTCP. %::ctcp contains the CTCP command (one word) and %::text contains any additional CTCP text.

CtcpChan You recieve a CTCP on a channel. %::ctcp contains the CTCP command (one word) and %::text contains any additional CTCP text. %::chan will contain the channel it was received on.

CtcpSelf You send a CTCP. %::nick will contain whom it was sent to.

%::ctcp and %::text work like CTCP.

CtcpChanSelf You send a CTCP to a chan. %::chan will contain where it was sent to. %::ctcp and %::text work like CTCP.

CtcpReply You recieve a CTCP reply. %::ctcp contains the CTCP reply (one word) and %::text contains any additional reply text. %::chan will contain the channel it was received on, if any.

CtcpReplySelf You send a CTCP reply. Not all scripts use this when sending

replies. %::chan or %::nick (but not both) will contain whom

it was sent to. %::ctcp and %::text work like CTCPREPLY.

Notify A user on your notify list is on IRC. %::text contains any notify note.

UNotify A user on your notify list has left IRC. %::text contains any notify note.

Wallop A wallop, contained in %::text.

NoticeServer A server notice, contained in %::text. %::nick will contain the server that sent the notice, which may differ from %::server.

Non-IRC events:

These events are not directly related to an IRC connection.

DNS Displayed when you start a /dns request. Either %::nick or %::address (but not both) will contain what you are /dns'ing, as appropriate.

DNSError A /dns request has failed. One of the following will be true-

1) Only %::nick is filled. You tried to /dns a user that was not found.

2) Only %::address is filled. You tried to /dns an address and failed.

3) Both are filled. You tried to /dns a user and failed, but the user was online.

If %::address is filled, either %::iaddress or %::naddress will also be filled, but this isn't usually used.

DNSResolve A /dns request has been resolved. %::nick will contain a nickname if appropriate. %::address, %::naddress, %::iaddress, and %::raddress will all be filled.

Echo Used for anything a script wants to display. %::text contains the actual text to display.

EchoTarget Used for anything related to or displayed in a channel.

%::target contains the target window name, %::text contains the actual text.

Error Used for any error messages a script displays. %::text contains the text to display.

Load Displayed (or run) when the theme is loaded. (after any script is loaded and any settings changed.)

Unload Displayed (or run) right before the theme is unloaded.

Whois and Whowas raws and events:

/whois (and /whowas) replies are handled a little differently in MTS. Each raw is called normally as expected- and if this is enough for a script (displaying the /whois line-by-line) then nothing more is needed. Some themes, however, like to wait until the entire whois reply is collected and then display it, for formatting purposes. In this case, leave the /whois raws blank, and add Whois and Whowas events. These will be called after the data for a full /whois or /whowas is collected. Since a /whois may reply on multiple nicknames, this event will either be called right before the end of whois raw (RAW.318) or right before the start of the next whois. (RAW.311) Most themes will use either the line-by-line raw method, or the Whois method- but a theme is allowed to use both if it needs to for any reason.

If a theme does not have a Whois or Whowas event, then a script should use default /whois or /whowas behavior if the raws are also not present.

Otherwise, hide output even if a specific /whois or /whowas raw is not found.

Note that any unrecognized raws seen between RAW.311 and RAW.318 should be treated as part of a /whois, and stored in or appended to %::text. This isn't a requirement, but suggested for ideal /whois operation. (if not done, %::text will be blank in the WHOIS or WHOWAS event.)

RAW.311 Start of whois- %::nick, %::address, %::realname.

RAW.314 Start of whowas- %::nick, %::address, %::realname.

RAW.319 Channels- %::chan. This raw may occasionally appear multiple times per nick. %::chan will contain -all-channels in the final WHOIS event.

RAW.312 Server- %::wserver, %::serverinfo. (in a whowas, %::serverinfo often contains signoff time)

RAW.301 Away- %::text.

RAW.307 Registered nick- %::isregd.

RAW.313 IRCop- %::isoper, %::operline.

RAW.317 Idle/signon- %::idletime, %::signontime.

RAW.318 End of whois- %::nick will contain the nickname or nickname(s) that were /whois'ed.

RAW.369 End of whowas- %::nick will contain the nickname or nickname(s) that were /whowas'ed.

Whois Display whois- Called right before 318. (or 311 in a multi-user whois)

Whowas Display whowas- Called right before 369. (or 314 in a multi-user whowas)

THIS IS NOT A COMPREHENSIVE LIST OF RAWS. ANY raw can be listed in a theme.

Simply use RAW.nnn, where nnn is the three digit raw number. If a raw is not found, RAW.OTHER should be used, if present. If not, the script can display the raw however it pleases. If you wish to hide the output from a raw, you should just write "!Script" for the event, with nothing to run. This will essentially "do nothing".

Any raw not listed here should have $2 in %::nick and %::chan, $3 in %::value, and $3- in %::text if $2 is a nickname or channel, or $2 in %::value and $2- in %::text otherwise.

Note that if a raw only uses ONE of %::value, %::nick, or %::chan, then the script is allowed to place the SAME VALUE in all three, for simplicity. A theme should not rely on this behavior.

RAW.001 Welcome to IRC.

RAW.002 Server and version- %::server (this is always set anyways) and %::value.

RAW.003 Server created on- %::value.

RAW.005 Protocols supported by server- %::text.

RAW.221 Current usermode- %::nick, %::modes.

RAW.250 Total connections- %::value.

RAW.251 Users, invisible, servers- %::users, %::text, %::value.

RAW.252 Operators- %::value.

RAW.253 Unknown connections- %::value.

RAW.254 Number of channels- %::value.

RAW.255 Local clients, servers- %::users, %::value.

RAW.265 Local users, max- %::users, %::value.

RAW.266 Global users, max- %::users, %::value.

RAW.302 Userhost- %::nick, %::address, %::value. Called once for each user in the userhost. (value contains + for here, - for away,

and also contains a * for an ircop.)

RAW.315 End of /who- %::value contains exactly what was /who'd.

RAW.324 Channel modes- %::chan, %::modes.

RAW.332 Channel topic- %::chan, %::text.

RAW.333 Topic set by and when- %::chan, %::nick, %::text.

RAW.341 User was invited- %::nick, %::chan.

RAW.352 /who data- %::nick, %::address, %::cmode, %::away, %::chan, %::wserver, %::realname, %::value, %::text, %::isoper (nick, address, cmode, chan, wserver, realname, and isoper are exactly like a /whois. away is H for here and G for gone. value is number of hops. text is the entire line, so the script can parse it manually if desired.)

RAW.353 Channel names list- %::chan, %::text.

RAW.366 End of names list- %::chan.

RAW.372 Message of the day- %::text.

RAW.375 Begin message of the day.

RAW.376 End message of the day.

RAW.391 Time/date at server- %::text.

RAW.401 No such nickname- %::nick.

RAW.403 No such channel- %::chan.

RAW.404 Unable to send message- %::chan.

RAW.421 Invalid command- %::value.

RAW.433 Nickname in use- %::nick.

RAW.471 Channel full- %::chan.

RAW.473 Channel invite only- %::chan.

RAW.474 Banned from channel- %::chan.

RAW.475 Channel key required- %::chan.

RAW.482 Not opped- %::chan.

RAW.Other Used for any raw not found in the theme.

CHAPTER 8 - MTS FILE FORMAT - SOUNDS (OPTIONAL)

This chapter describes a method of adding sound events to a theme.

Support for this in a script is entirely optional. Scripts are encouraged to have their own sound events not listed here.

Sounds can be defined for events by simply making another line in the theme in the form of:

SndEventName filename.wav

"EventName" should match a MTS event or a mIRC event or command, if possible. An event that matches an IRC event affecting you should include the "Self" suffix. Some more commonly used sound events-

SndNotice

SndJoin

SndJoinSelf

SndPart

SndPartSelf

SndKickSelf

SndOpSelf

SndDeopSelf

SndBanSelf

SndInvite

SndNotify

SndUNotify

SndError

SndStart

SndConnect

SndDisconnect

SndOpen

SndDCC

SndDialog

SndAway

SndBack

SndPager

Ideally, the theme engine should determine the type of file the theme is trying to play (whether mid, mp3, or wav), and use the appropriate command. MTS does not standarize on a sound format. The simplest way to implement sounds is to add a

line in /theme.text that checks for SndEventName and if it exists, play the sound. This will not catch many of the more common events, however, such as SndOpen or SndConnect.

This is up to scripter preference.

CHAPTER 9 - MTS FILE FORMAT - SCHEMES (OPTIONAL)

This chapter describes a method of supporting multiple color sets in a theme, hereafter referred to as "schemes". Support for this in a script is entirely optional. Themes do not need to contain any schemes.

The concept of schemes is to allow a theme to have a few things that change for each scheme, and let the rest remain the same across all schemes. Usually, the items changing will be Colors, RGBColors, and similar items, but this is not required. For example, a theme could contain a "red" scheme, a "blue" scheme, and a "green" scheme, which would contain different Colors lines; however, everything else would remain the same.

Schemes prevent having to create entire new themes just to change minor things such as colors.

For each scheme included in a theme, you need to add a SchemeN line to the main theme section, containing a description or name for that scheme. Then, add a [schemeN] section to the theme containing all items that are changing in that scheme. Here is a short example-

[mts]

Name This is an example theme.

MTSVersion 0.97

; (normal theme lines would go here)

; We include a default colors line for scripts that

; don't support schemes.

Colors 1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,1,2

; These are our schemes

Scheme1 Red

Scheme2 Green

[scheme1]

; Colors for the 'Red' scheme

Colors 8,7,6,5,4,3,2,1,8,7,6,5,4,3,2,1,8,7,6,5,4,3,2,1,8,7,8,8

[scheme2]

; Colors for the 'Green' scheme

Colors 9,1,6,5,4,7,3,2,3,5,1,3,5,9,3,8,4,5,2,7,3,2,4,1,0,5,9,9

; We also change the prefix for this scheme

Prefix [*]

; (end of file)

When a script (that supported schemes) were to load this theme, it would first load everything in the [mts] section. Then, it would grab all of the scheme names from the SchemeN lines, and prompt the user for the scheme to load. Then, it would load the associated [schemeN] section, overriding anything in the original theme.

Note how the theme includes a Colors line in the main section. This is so that a script that doesn't support schemes will still load the theme properly.

CHAPTER 10 - SCRIPTING AND THEMING NOTES

Some important notes that apply to the script processing the theme files-

1) If using $replace to replace text, always replace <text> last, as it might contain other <codes>. <chan>, <ctcp>, <away>, and <realname> should also be replaced as late as possible, as they could also contain <codes>.

Note that there are ways to process theme lines that do not have this limitation, but $replace is the simplest to code.

2) To allow the use of < and > in a theme line, a script must replace <lt> with < and <gt> with >. If using $replace, you should do this last- after everything else. It's possible for text already replaced to contain these values, but this is an unavoidable side effect- worse side effects could occur if replaced earlier.

3) When setting %::variables, you should probably use /set -u so the variables unset automatically. Be careful with /whois variables, as you will need to have them properly set for the final WHOIS event as well. (you may wish to store them in a more permanent location until that event)

4) When setting %::value, %::nick, and %::chan in a raw event, you can set all three to the same value if you're not sure which one applies for a given raw, as the important value is usually in $2. You SHOULD NOT do this for any raw that uses more than one of these variables. A theme should not rely on a script doing this.

5) <variables> are written with those characters to better support future expansion and avoid conflicts that could be caused by other methods of writing variables.

6) A script is free to define it's own <variables>, events, settings, etc.

Other scripts should ignore lines in a theme file that it does not recognize. If possible, a script should also remove <variables> it doesn't recognize, but this is not required behavior.

7) Remember to reset /timestamp when mIRC starts, as otherwise any control codes will be removed.

8) When reading a !Script line, be sure to $eval() it, not just run the command as-is. This allows for single-line scripting without having to create separate aliases. For example, the following line should work, and will work properly if you simply use $eval() on the line when running it.

TextChan !Script %:echo ($+ $lower(%::nick) $+) %::text

9) When loading a theme, remember to use /loadbuf or some other method that only gets the [mts] section. Even if you don't actually support schemes, this will make sure you don't accidentally load settings from one.

10) If your script supports schemes, make sure to load the scheme lines (overriding any parts of the theme) before handing SCRIPT and LOAD lines, just in case they are changed by the scheme.

11) If your script uses %:comments, you should simply add them to the end of a normal text line. There is no <comments> code to replace. !Script should handle %:comments on it's own.

12) Regular expressions are very powerful. pai has come up with some really great example code on www.mircscripts.org/mts.php for speeding up your engines, so make sure to check it out.

13) If the channel or nickname returned from the server in an event is *, then the respective %::chan or %::nick should be set to $null.

14) In an event that sends text to multiple windows (such as Nick and Quit events), remember that the theme ENGINE is what does the looping, not the theme. The theme is only for formatting the text.

Order of items in a Colors line-

Background, Action text, CTCP text, Highlight text, Info text, Info2 text,

Invite text, Join text, Kick text, Mode text, Nick text, Normal text,

Notice text, Notify text, Other text, Own text, Part text, Quit text,

Topic text, Wallops text, Whois text, Editbox, Editbox text,

Listbox, Listbox text, Inactive, Grayed text, Title text

Default mIRC colors for a Colors line-

0,6,4,5,2,3,3,3,3,3,3,1,5,7,6,1,3,2,3,5,1,0,1,0,1,15,6,0

Default mIRC RGB values for an RGBColors line-

255,255,255 0,0,0 0,0,128 0,144,0 255,0,0 128,0,0 160,0,160 255,128,0

255,255,0 0,255,0 0,144,144 0,255,255 0,0,255 255,0,255 128,128,128

208,208,208

Important notes for theme files-

1) A theme alias should never use /halt. Use /return instead.

2) A theme alias should never modify (or unset) %::variables.

3) A theme file should not contain the mIRC ON LOAD and ON UNLOAD events. Use the MTS LOAD and UNLOAD events instead.

4) A theme that contains schemes should still include all settings (that it uses) in the main [mts] section, in case a script that doesn't support schemes tries to load it.

5) Remember to begin your theme with the [mts] line. This is required so that scripts can ease theme (and scheme) loading. (even if your theme or script does not support schemes, all themes must still use this line for compatibility.)

6) Do not worry about %:comments for normal text lines, but when you !Script an event, you should display it. This doesn't apply to raws.

7) Color codes should ALWAYS BE DOUBLE DIGIT. Although, most people still aren't used to doing this, so a theme engine should IDEALLY try and compensate for a theme that doesn't have double digit color codes and adjust accordingly.

cold from www.mircscripts.org wrote this alias to do conversions:

; $'repcolor(string, n1, n2[, n3, n4[, ..., nN]])

; - replaces colors which code is n1 to n2, then n3 to n4 and successively, but not if the code that's ought to be replaced has a background color

; - examples: $'repcolor(_1Hi! _2How _01are _12you?, 1, 15) returns "_15Hi! _2How _15are _12you?"

; supposing %blah is "_15,1Hi! _14How _15are _14you?",

; $'repcolor(%blah, 15, 4) returns "_15,1Hi! _14How _4are _14you?"

; $'repcolor(_4many _5replaced _04colors _06here!, 4,12, 6,13, 5,7) returns "_12many _7replaced _12colors _13here!"

alias 'repcolor {

```
var %i = 0, %s = $1, %c = $2-, %p = $0 - 1

while (%i < %p) {

inc %i 2

tokenize 32 $gettok(%c, $+($calc(%i -1),-,%i), 32)

if ($0 < 2) { break }

if ($1 !isnum 0-99) || ($2 !isnum 0-99) { return }

var %x, %y = $regsub(%s, /_( $+ $1)( $+ $iif(? iswm $1,[^0-9,[^) $+ $chr(44) $+ ])/g, $+(_,$2,\2), %x)

if (? iswm $1) { %y = $regsub(%x, /_( $+ $base($1,10,10,2) $+ )([^0-9 $+ $chr(44) $+ ])/g, $+(_, $base($2,10,10,2), \2), %x) }

%s = %x

}

return %x

}
```

8) !Script can not be used on ParenText, Prefix, or Timestamp (doesn't make sense to do this)

9) Themes should NOT USE /echo. They should only use %:echo.

10) <variables> are not to be used on !Script lines - use the correct %::variable instead.

11) An item in a theme file with no value, such as TextChan by itself, should do nothing.

This especially applies to schemes - a scheme should have the capability to override (or remove) a setting in the main theme.

CHAPTER 11 - STANDARD MTS ALIAS NAMES

The following are required aliases for the theme engine. These are standardized so that addons can easily detect and use an MTS-enabled script.

/theme.text <event>

For calling the theme engine to do the theme text (variables already set)

$mtsversion

Returns the MTS version supported by the theme engine. Can be used by an addon to see if the script supports MTS.

$theme.setting(<setting>)

Returns a line from the theme without any processing.

The following are suggested aliases, that a script will probably want to have.

/theme.load

Loads a theme

/theme.scheme

Changes the scheme used in the theme

/theme.unload

Unloads a theme, returning to default display

CHAPTER 12 - MTS LOADING EVENT ORDER

To correctly load a theme, the various load events must be triggered in a specific order, so as to not cause conflicts in the various theme engines.

1) If a theme is currently loaded, it must have the MTS Unload event triggered, then the Script (if applicable) unloaded from mIRC's memory, and then the theme file must be unloaded.

2) The new theme .MTS file should be loaded up.

3) The selected scheme should be overlaid on top of the base .MTS file.

4) All theme engine specific processing should be done - for example, some theme engines choose to "pre-compile" the MTS event lines into native mIRC coding, so as to require less processing later at the actual event trigger.

5) Once all pre-load theme processing is complete (some theme engines may not have very much), the theme engine should load up the theme's Script file (if applicable).

6) The theme's MTS Load event should be triggered (so if the theme wishes to generate background images, etc, it may, before step 7)

7) Theme colors and fonts should be applied.

8) Theme background images should be applied.

CHAPTER 13 - THEME DISTRIBUTION

Up to now, themes have been distributed in a variety of formats, ranging from sending just the .mts file, to sending .zip files, and to .rar files, and various other formats.

This has caused a fair amount of headache for the theme engine writers, as well as users, because there aren't any official suggestions for theme distribution, so everyone has either ignored this aspect of themes, or tried to implement every possible distribution possibility.

Up until recently, there wasn't a very good way to work with .zip files in mIRC, because there weren't any DLLs available that would work with them. There is one now, that can decompress zip files, named mUnzip, by Kamek. It can be obtained from www.mircscripts.org.

Another option that isn't very often explored by scripters, is packing up themes (not compressing) through the use of scripting. These file packs can then be compressed through the use of ngzipn.dll, and unpacked with the same file. In case you are not aware, gzip can only compress a single file, hence the need for the previous file-archive step.

Doing this through scripting can be a more elegant solution because it gives you the capability to also pack up a theme for distribution, all through scripting, which few, or no theme engines support.

One possible solution for scripted file-packing/unpacking and compression with ngzipn.dll is a small addon named vPak. It can be obtained at www.mircscripts.org. A few theme engines for unpacking already support it, but theme editors could really benefit from it by giving the themer the option to package up his/her theme for distribution.

No matter which compression option you choose, files should be packed in a certain manner.

1) No directories should prefix the theme files. Meaning, if your theme is named Ultra, the distribution file should NOT contain the "Ultra" directory to extract to - this should be left up to the theme engine to determine where the theme should be extracted to, and what the directory should be called.

2) Themes should not contain any subdirectories of their own.

Given these guidelines, compressed files should at least give theme engine scripters less of a headache, and if someone implements a way to zip up themes via a DLL, then themers will also be happier because it will make it easier to distribute themes they make.

CHAPTER 14 - NAMED COLOR LIST

MTS 1.3 engines must be able to convert these colors to the appropriate RGB values in order to be fully compliant.

Please note:

"Aqua" and "Cyan" produce the same color: 0,255,255

"Fuchsia" and "Magenta" produce the same color: 255,0,255

These are the standard HTML color codes. They have been modified only to be converted from RRGGBB hex format to RR,GG,BB integer format.

Aliceblue 240,248,255

Antiquewhite 250,235,215

Aqua 0,255,255

Aquamarine 127,255,212

Azure 240,255,255

Beige 245,245,220

Bisque 255,228,196

Black 0,0,0

Blanchedalmond 255,235,205

Blue 0,0,255

Blueviolet 138,43,226

Brown 165,42,42

Burlywood 222,184,135

Cadetblue 95,158,160

Chartreuse 127,255,0

Chocolate 210,105,30

Coral 255,127,80

Cornflowerblue 100,149,237

Cornsilk 255,248,220

Crimson 220,20,60

Cyan 0,255,255

Darkblue 0,0,139

Darkcyan 0,139,139

Darkgoldenrod 184,134,11

Darkgray 169,169,169

Darkgreen 0,100,0

Darkkhaki 189,183,107

Darkmagenta 139,0,139

Darkolivegreen 85,107,47

Darkorange 255,140,0

Darkorchid 153,50,204

Darkred 139,0,0

Darksalmon 233,150,122

Darkseagreen 143,188,143

Darkslateblue 72,61,139

Darkslategray 47,79,79

Darkturquoise 0,206,209

Darkviolet 148,0,211

Deeppink 255,20,147

Deepskyblue 0,191,255

Dimgray 105,105,105

Dodgerblue 30,144,255

Firebrick 178,34,34

Floralwhite 255,250,240

Forestgreen 34,139,34

Fuchsia 255,0,255

Gainsboro 220,220,220

Ghostwhite 248,248,255

Gold 255,215,0

Goldenrod 218,165,32

Gray 128,128,128

Green 0,128,0

Greenyellow 173,255,47

Honeydew 240,255,240

Hotpink 255,105,180

Indianred 205,92,92

Indigo 75,0,130

Ivory 255,255,240

Khaki 240,230,140

Lavender 230,230,250

Lavenderblush 255,240,245

Lawngreen 124,252,0

Lemonchiffon 255,250,205

Lightblue 173,216,230

Lightcoral 240,128,128

Lightcyan 224,255,255

Lightgoldenrodyellow 250,250,210

Lightgreen 144,238,144

Lightgrey 211,211,211

Lightpink 255,182,193

Lightsalmon 255,160,122

Lightseagreen 32,178,170

Lightskyblue 135,206,250

Lightslategray 119,136,153

Lightsteelblue 176,196,222

Lightyellow 255,255,224

Lime 0,255,0

Limegreen 50,205,50

Linen 250,240,230

Magenta 255,0,255

Maroon 128,0,0

Mediumauqamarine 102,205,170

Mediumblue 0,0,205

Mediumorchid 186,85,211

Mediumpurple 147,112,216

Mediumseagreen 60,179,113

Mediumslateblue 123,104,238

Mediumspringgreen 0,250,154

Mediumturquoise 72,209,204

Mediumvioletred 199,21,133

Midnightblue 25,25,112

Mintcream 245,255,250

Mistyrose 255,228,225

Moccasin 255,228,181

Navajowhite 255,222,173

Navy 0,0,128

Oldlace 253,245,230

Olive 128,128,0

Olivedrab 104,142,35

Orange 255,165,0

Orangered 255,69,0

Orchid 218,112,214

Palegoldenrod 238,232,170

Palegreen 152,251,152

Paleturquoise 175,238,238

Palevioletred 216,112,147

Papayawhip 255,239,213

Peachpuff 255,218,185

Peru 205,133,63

Pink 255,192,203

Plum 221,160,221

Powderblue 176,224,230

Purple 128,0,128

Red 255,0,0

Rosybrown 188,143,143

Royalblue 65,105,225

Saddlebrown 139,69,19

Salmon 250,128,114

Sandybrown 244,164,96

Seagreen 46,139,87

Seashell 255,245,238

Sienna 160,82,45

Silver 192,192,192

Skyblue 135,206,235

Slateblue 106,90,205

Slategray 112,128,144

Snow 255,250,250

Springgreen 0,255,127

Steelblue 70,130,180

Tan 210,180,140

Teal 0,128,128

Thistle 216,191,216

Tomato 255,99,71

Turquoise 64,224,208

Violet 238,130,238

Wheat 245,222,179

White 255,255,255

Whitesmoke 245,245,245

Yellow 255,255,0

YellowGreen 154,205,50

CHAPTER 15 - FUTURE EXPANSION

1) List support - Not sure what everyone thinks about this:

List events will provide support for echoing data in an easily readable display.

ListBegin Beginning of list - %::text

ListText Line of data - %::text

ListSep Line seperator - %::text

ListEnd End of list - %::text

The issues with this idea that have been brought up are:

1. Why not use !Script?

2. This introduces a bit more complexity into theme engines, that isn't needed due to the !Script support.

2) A theme distribution part of MTS is still needed - right now, the majority of people that use MTS themes are more advanced users than most, that know how to unzip files to a certain location, etc...

If a distribution standard was in place, it would be simpler for everyone.

The community needs a simple .DLL that can compress .zip files.

Take a look at every program out there that has any type of support for themes/skins - they all have a "standard" that is used for distribution, that when followed, makes it very easy on the end users using the product.

A few suggestions have been made in draft 8.3 for this. Unless a DLL is made that can support compressing and uncompressing .zip files in the near future, a scripted file packing/unpacking and ngzipn.dll compressing/decompressing solution should be picked as the standard, because this is an extremely important part of any theme/skin standard. Currently vPak, as previously mentioned, fits this role perfectly, but this is only due to a lack of options - nothing else has been released for mIRC in this category.

Once we iron out this detail, MTS 2.0 can be released, as this will be a large step in the standard.

3) Should support for IRCX be added? Not too many scripts even support IRCX servers, much less theme engines... and IRCX servers don't seem to be getting a lot more popular either.

CHANGE LOG

5-14-2002: Draft 8.4 changes (Variant)

---

1. Fixed various wording issues

2. Forgot to mention in draft 8.3 that support for multiple fonts was added. Check chapter 6 for details.

3. Added list of named colors this is now chapter 14. Chapter 15 is the Future Expansion chapter now.

5. Other notes on this draft:

I have left in the Theme Distribution chapter unchanged from Draft 8.3, because it seems that the general concensus is that MTS themes need a standard for this sort of thing... but not very many people are talking about it or suggesting any other options than the ones I have proposed there... .zip, or a script-pack + .gz setup. We really need feedback on this guys.

The advantages to a script-pack + .gz setup are that gzipping the theme pack isn't required... it is just simply convenient to be able to compress a theme.

Does anyone feel that the multiple font support could be much better? In adding it, I took a general concensus of what people had asked for in this addition and added it, keeping the capability for themes (and engines) to be backward compatible as much as possible... does it need to be changed?

5-2-2002: Draft 8.3 changes (Variant)

---

1. Added various wording in places, mainly explaining why plain text vs.

INI files are used.

2. Added in various new items relating to mIRC 6.0+, mainly in the colors.. Inactive, Grayed text, Title text.

3. Removed Example MTS theme - there are many themes available now at www.mircscripts.org.

4. Added Chapter 12 - MTS event loading order.

5. Added Chapter 13 - Theme Distribution.

6. Modified chapter 14 a bit, with some questions posed for the community.

9-24-2001: Draft 8.2 changes (Variant)

---

1. Fixed numerous typos, SELFNICK - NICKSELF, SELFKICK, etc. Also there were a couple instances of "send" left, they should have been "self."

2. Added quite a few more notes to chapter 11 , both for scripts, and for themes. Please read it to make sure you haven't made any of these common mistakes.

3. Added <timestamp> and %::timestamp. Timestamp format is now on the appropriately named TimestampFormat line, and now Timestamp is simply ON or OFF.

4. MTSVersion is now 1.1 due to the <timestamp> addition.

8-20-2001: Draft 8.1 changes (Variant)

---

1. Typo <addres>s fixed in ultra.mts example theme

2. Standardized on "Self" instead of "Send" - no point in having both.. CtcpSelf now instead of CtcpSend, etc.

8-17-2001: Draft 8 changes (Variant)

---

1. Fixed typos, <b1> to <c1> etc...

2. Few other small typos

3. Added some wording in some places.

4. Changed FontStatus to FontDefault

5. MTSVersion is now 1.0

This draft is largely unchanged from draft 7, but it needed a few small tweaks here and there to be called "final."

8-16-2001: Draft 7 changes (pai)

---

1. Some reformatting changes, not as extreme as blue-elf but should be a bit more readable now.

2. Changed "ChanText" to "TextChan". Similar changes for all Text and Action events. All of the events have similar naming schemes now.

3. Changed %::comments to %:comments. Single-colon variables are those that do not have a corresponding <variable> code.

4. Moved sound support to it's own chapter, clarified it as optional, added some recommended events.

5. Readded scheme support as it's own chapter, clarified it as optional.

6. Themes must contain an [mts] line to allow scheme support to function properly. (even scripts/themes not using schemes should include this line.)

7. Clarified BaseColors so themes have a better idea of which colors can or should go there. Sample theme now uses BaseColors.

8. Changed () addition to usage of ParenText and <parentext>. See %::parentext and chapter 6 for details.

9. Some other minor corrections.

10. MTSVersion is now 0.97. This draft will probably become the final draft except for grammatical fixes. MTSVersion 1.0 is still pending.

8-15-2001: Draft 6 changes (Variant)

Too many changes to mention all.

1. Naming for all events has been updated to be more organized

2. .mts files are now recommended to be in the theme's directory with all the other supplemental files.

3. Scheme support removed - complicated the standard too much.

4. .mts files can now be loaded from anywhere, but if they have supplemental files, the .mts file must be in the same directory as the supplemental files. It is HIGHLY recommended that supplemental files for one theme not be in the same directory as supplemental files from another theme, due to possible naming conflicts.

5. Sound support has been added.

6. MTSVersion is now 0.96 since this draft is the final draft except for grammatical fixes. MTSVersion 1.0 is pending.

8-8-2001: Draft 5 changes (Variant)

---

1. Changed ** to ---- everywhere (its a bit more professional)

2. In chapter 3, it used to say that supplemental theme files could go in the same directory as the .mts file, which is not a good idea because it could cause clutter. The standard is the same as it was in draft 2 now - supplemental files must be in the theme's subdirectory.

3. Edited chapter 4 wording, it sounded like the theme's aliases should be specifying %:echo and the other theme vars, which is incorrect.

4. Removed chapter 10 (example mts script) because it doesn't conform with the standard alias names that were defined in previous drafts. Example MTS engines can be addons, and should not be included in the standard, because we do not suggest how scripters should/could implement their engines. This is a standard for theme files. Chapter 10 is now suggested alias names.

5. Added $mtsversion required alias that returns the version of MTS a theme engine supports, if the script supports MTS.

8-7-2001: Draft 4.7 changes (pai)

---

1. Added SCRIPT item. (accidentally missing from draft 4)

2. Fixed draft 4 changes date.

3. Added example MTS script chapter.

4. Added a few notes to chapter 9, scripting notes.

5. Corrected a couple chapter headings.

6. Added note for %::nick in SNOTICE event.

7. Added note that %::value, %::nick, and %::chan can be set to the same thing in a raw event for simplicity, except in raws that use more than one of them.

8. Added order and defaults for COLORS.n lines to chapter 9.

9. Added defaults for RGBCOLORS.n lines to chapter 9.

10. Added note to RGBCOLORS.n entry regarding defaults.

11. Changed tabs to spaces for better formatting.

12. Removed a reference to %::isoperb and %::isregdb.

13. Removed "!Parse"- !Script can handle both cases.

14. MTSVERSION is still 0.95

8-6-2001: Draft 4.5 changes (tabo)

---

1. A theme alias should never modify or UNSET %::variables.

2. %::me must be set for all events

3. Events: Added %::target

4. Raws: Added %::fromserver

5. Whois: Added %::operline (in some servers, not all the operators will have the same oper line, so this is needed)

6. Added the "!Parse" prefix to the events in the .MTS file. It will be used instead of "!Script" for small snippets in the text theme. As discussed with Variant, having !Script doing 2 different things can get confusing.

(!Parse or !Eval ?)

7. Added something new to consider in Chapter 9

8. MTSVERSION is now 0.95

8-6-2001: Draft 4 changes (pai)

---

1. Reorganized and rewrote most text. Added a lot of explanations.

2. Using "!Script " instead of "!Script:". (minor syntax change)

3. Recommending $eval() for !Script lines. (to allow one-liners)

4. Scripting notes added

5. Using <var> instead of &var;. Added <gt> and <lt>.

6. DNS <variables> and WHO/WHOIS/WHOWAS <variables> documented fully.

7. %::echo changed back to %:echo.

8. WEBSITE item added.

9. <cnick> added.

10. Four generic colors- <c1> <c2> <c3> <c4> and corresponding BASECOLORS option added.

11. Added color schemes and renamed color-related items. Added NUMSCHEMES and SCHEME.n.

12. RGBs are now all in one line.

13. Removed FONT.CHAT, FONT.DQWINDOW. (use FONT.QUERY)

14. Added CNICK.IRCOP.

15. Removed CNICK.*CHAR. Multiserver scripts can add these as script-specific options if desired.

16. Added SNOTICE, WALLOP.

17. Added CHANNOTICE, SELFCHANNOTICE.

18. Detailed WHOIS support instructions, for using both RAW.nnn and WHOIS events.

19. Remvoed %::isoperb and %::isregdb. A theme script can just check using ==.

(no need for redundant variables)

20. WHOIS now uses %::away for away message and uses %::text for "miscellaneous" text.

21. Added %::serverinfo for WHOIS.

22. WHO now uses a lot of %::vars. (no new ones- just ones that were defined for WHOIS)

23. Temporarily removed scripting examples.

24. MTSVERSION is now 0.94

8-5-2001: Draft 3 changes (pai)

---

1. Changed %:echo to %::echo, so that all MTS variables use %::.

2. Clarified usage of many %::vars

3. Added %::value (for raws)

4. Removed or renamed some %::vars that were related to connection raws

5. Added a few %::vars to whois, and changed %::channels to %::chan

(redundant)

6. Changed &var to &var; instead, and added suggestion to $replace & with &.

7. Changed -u0 to -u in scripting examples.

8. Added %::me, mostly for use with SELF* events.

9. Clarified directory structure section- subdirectories not needed for themes without extra files.

10. Renamed NORMNICK to NICK, NORMKICK to KICK, NORMJOIN to JOIN.

11. Added MSGTEXT, SELFMSGTEXT.

12. Replaced ACTION, SELFACTION with CHANACTION, SELFCHANACTION and QUERYACTION, SELFQUERYACTION.

13. Renamed ERROR to SERVERERROR (why? see 14.)

14. Added ECHO, CHANECHO, QUERYECHO, ERROR.

15. Renamed CLINE.* to CNICK.*, added OWNER, plus CNICK.ENEMY, .ME, .FRIEND, and .MISC.

16. Renamed Font* to FONT.* for consistency, renamed FontChannel to FONT.CHAN.

17. Renamed *Background to IMAGE.* for consistency, ChannelBackground to IMAGE.CHAN.

(Using IMAGE because toolbar/switchbar isn't really a background. See 18.)

18. Added IMAGE.TOOLBAR, IMAGE.BUTTONS, and IMAGE.SWITCHBAR.

19. Added TIMESTAMP.

20. Added UNLOAD.

21. Added NOTIFY, UNOTIFY.

22. Updated all scripting examples to match changes

23. MTSVERSION is now 0.92

8-3-2001: Draft 2 changes

---

1. Changed %::isircop and %::reggednick to %::isoper and %::isregd

2. Fixed &var& typos - should be all &vars.

3. Updated small parts in the example ultra theme that used unknown &variable names - they should have been using &text.

4. Added section on suggested use of multi-line returns in chapter 4.

5. Removed :)'s :P

6. Added FontChat and FontDQWindow.

7. Added CLINE. lines into the default ultra theme. Only multiserver scripts can use the CLINE.*CHAR lines though, since they don't use mIRC's IRC connection and can /aline to their @windows.

8. Added VERSION and LOAD lines into the MTS standard, to have the theme version, and LOAD to echo something about your theme being loaded, or to use !Script: to process an alias in your theme's .mrc file.

9. MTSVERSION is now 0.91

CREDITS

Current credits include: Deleted, cold, Kamek, b1ind, Ecronika, sax, Eric^^, Dark_Greg, tabo, Oxigun, blue-elf, Psionic, pai, Variant

The current credits have grown far beyond this small list. Thanks to all the scripters from www.mircscripts.org, and special thanks to fubar for creating the MTS section of mircscripts.org and really helping MTS to become a reality!

Help file format (.hlp and .chm) compiled by OryNider (http://pubory.uv.ro , ory_001@yahoo.co.uk).

This file is hosted on www.mirc.net!